8
Polynomial Interpolation

This chapter is addressed to the approximation of a function which is known through its nodal values.

Precisely, given $m+1$ pairs (x_i, y_i), the problem consists of finding a function $\Phi = \Phi(x)$ such that $\Phi(x_i) = y_i$ for $i = 0, \ldots, m$, y_i being some given values, and say that Φ *interpolates* $\{y_i\}$ at the nodes $\{x_i\}$. We speak about *polynomial interpolation* if Φ is an algebraic polynomial, *trigonometric approximation* if Φ is a trigonometric polynomial or *piecewise polynomial interpolation* (or *spline interpolation*) if Φ is only locally a polynomial.

The numbers y_i may represent the values attained at the nodes x_i by a function f that is known in closed form, as well as experimental data. In the former case, the approximation process aims at replacing f with a simpler function to deal with, in particular in view of its numerical integration or derivation. In the latter case, the primary goal of approximation is to provide a compact representation of the available data, whose number is often quite large.

Polynomial interpolation is addressed in Sections 8.1 and 8.2, while piecewise polynomial interpolation is introduced in Sections 8.3, 8.4 and 8.5. Finally, univariate and parametric splines are addressed in Sections 8.6 and 8.7. Interpolation processes based on trigonometric or algebraic orthogonal polynomials will be considered in Chapter 10.

8.1 Polynomial Interpolation

Let us consider $n+1$ pairs (x_i, y_i). The problem is to find a polynomial $\Pi_m \in \mathbb{P}_m$, called an *interpolating polynomial*, such that

$$\Pi_m(x_i) = a_m x_i^m + \ldots + a_1 x_i + a_0 = y_i \quad i = 0, \ldots, n. \tag{8.1}$$

The points x_i are called *interpolation nodes*. If $n \neq m$ the problem is over or under-determined and will be addressed in Section 10.7.1. If $n = m$, the following result holds.

Theorem 8.1 *Given $n+1$ distinct points $x_0, \ldots, x_n$ and $n+1$ corresponding values $y_0, \ldots, y_n$, there exists a unique polynomial $\Pi_n \in \mathbb{P}_n$ such that $\Pi_n(x_i) = y_i$ for $i = 0, \ldots, n$.*

Proof. To prove existence, let us use a constructive approach, providing an expression for Π_n. Denoting by $\{l_i\}_{i=0}^n$ a basis for $\mathbb{P}_n$, then Π_n admits a representation on such a basis of the form $\Pi_n(x) = \sum_{i=0}^n b_i l_i(x)$ with the property that

$$\Pi_n(x_i) = \sum_{j=0}^{n} b_j l_j(x_i) = y_i, \quad i = 0, \ldots, n. \tag{8.2}$$

If we define

$$l_i \in \mathbb{P}_n : \quad l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \quad i = 0, \ldots, n, \tag{8.3}$$

then $l_i(x_j) = \delta_{ij}$ and we immediately get from (8.2) that $b_i = y_i$.
The polynomials $\{l_i,\ i = 0, \ldots, n\}$ form a basis for $\mathbb{P}_n$ (see Exercise 1). As a consequence, the interpolating polynomial exists and has the following form (called *Lagrange* form)

$$\Pi_n(x) = \sum_{i=0}^{n} y_i l_i(x). \tag{8.4}$$

To prove uniqueness, suppose that another interpolating polynomial Ψ_m of degree $m \leq n$ exists, such that $\Psi_m(x_i) = y_i$ for $i = 0, ..., n$. Then, the difference polynomial $\Pi_n - \Psi_m$ vanishes at $n+1$ distinct points x_i and thus coincides with the null polynomial. Therefore, $\Psi_m = \Pi_n$.

An alternative approach to prove existence and uniqueness of Π_n is provided in Exercise 2. ◇

It can be checked that (see Exercise 3)

$$\Pi_n(x) = \sum_{i=0}^{n} \frac{\omega_{n+1}(x)}{(x - x_i)\omega'_{n+1}(x_i)} y_i \tag{8.5}$$

where ω_{n+1} is the *nodal polynomial* of degree $n+1$ defined as

$$\omega_{n+1}(x) = \prod_{i=0}^{n}(x - x_i). \tag{8.6}$$

Formula (8.4) is called the *Lagrange form* of the interpolating polynomial, while the polynomials $l_i(x)$ are the *characteristic polynomials*. In Figure 8.1 we show the characteristic polynomials $l_2(x)$, $l_3(x)$ and $l_4(x)$, in the case of degree $n = 6$, on the interval [-1,1] where equally spaced nodes are taken, including the end points.

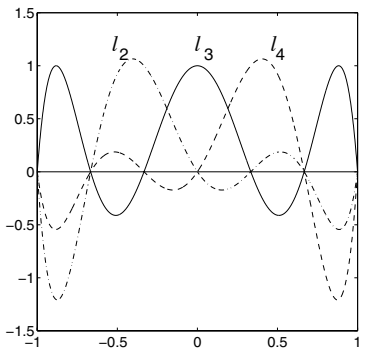

FIGURE 8.1. Lagrange characteristic polynomials

Notice that $|l_i(x)|$ can be greater than 1 within the interpolation interval.

If $y_i = f(x_i)$ for $i = 0, \ldots, n$, f being a given function, the interpolating polynomial $\Pi_n(x)$ will be denoted by $\Pi_n f(x)$.

8.1.1 The Interpolation Error

In this section we estimate the interpolation error that is made when replacing a given function f with its interpolating polynomial $\Pi_n f$ at the nodes $x_0, x_1, \ldots, x_n$ (for further results, we refer the reader to [Wen66], [Dav63]).

Theorem 8.2 *Let $x_0, x_1, \ldots, x_n$ be $n+1$ distinct nodes and let x be a point belonging to the domain of a given function f. Assume that $f \in C^{n+1}(I_x)$, where I_x is the smallest interval containing the nodes $x_0, x_1, \ldots, x_n$ and x. Then the interpolation error at the point x is given by*

$$E_n(x) = f(x) - \Pi_n f(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x), \tag{8.7}$$

where $\xi \in I_x$ and ω_{n+1} is the nodal polynomial of degree $n+1$.

Proof. The result is obviously true if x coincides with any of the interpolation nodes. Otherwise, define, for any $t \in I_x$, the function $G(t) = E_n(t) - \omega_{n+1}(t)E_n(x)/\omega_{n+1}(x)$. Since $f \in C^{(n+1)}(I_x)$ and ω_{n+1} is a polynomial, then $G \in C^{(n+1)}(I_x)$ and it has $n+2$ distinct zeros in I_x, since

$$G(x_i) = E_n(x_i) - \omega_{n+1}(x_i)E_n(x)/\omega_{n+1}(x) = 0, \quad i = 0, \dots, n$$
$$G(x) = E_n(x) - \omega_{n+1}(x)E_n(x)/\omega_{n+1}(x) = 0.$$

Then, thanks to the mean value theorem, G' has $n+1$ distinct zeros and, by recursion, $G^{(j)}$ admits $n+2-j$ distinct zeros. As a consequence, $G^{(n+1)}$ has a unique zero, which we denote by ξ. On the other hand, since $E_n^{(n+1)}(t) = f^{(n+1)}(t)$ and $\omega_{n+1}^{(n+1)}(x) = (n+1)!$ we get

$$G^{(n+1)}(t) = f^{(n+1)}(t) - \frac{(n+1)!}{\omega_{n+1}(x)}E_n(x),$$

which, evaluated at $t = \xi$, gives the desired expression for $E_n(x)$. ◇

8.1.2 Drawbacks of Polynomial Interpolation on Equally Spaced Nodes and Runge's Counterexample

In this section we analyze the behavior of the interpolation error (8.7) as n tends to infinity. For this purpose, for any function $f \in C^0([a,b])$, define its *maximum norm*

$$\|f\|_\infty = \max_{x\in[a,b]} |f(x)|. \tag{8.8}$$

Then, let us introduce a lower triangular matrix X of infinite size, called the *interpolation matrix* on $[a,b]$, whose entries x_{ij}, for $i,j = 0,1,\dots$, represent points of $[a,b]$, with the assumption that on each row the entries are all distinct.

Thus, for any $n \geq 0$, the $n+1$-th row of X contains $n+1$ distinct values that we can identify as nodes, so that, for a given function f, we can uniquely define an interpolating polynomial $\Pi_n f$ of degree n at those nodes (any polynomial $\Pi_n f$ depends on X, as well as on f).

Having fixed f and an interpolation matrix X, let us define the interpolation error

$$E_{n,\infty}(\mathrm{X}) = \|f - \Pi_n f\|_\infty, \quad n = 0, 1, \dots \tag{8.9}$$

Next, denote by $p_n^* \in \mathbb{P}_n$ the *best approximation polynomial*, for which

$$E_n^* = \|f - p_n^*\|_\infty \leq \|f - q_n\|_\infty \qquad \forall q_n \in \mathbb{P}_n.$$

The following comparison result holds (for the proof, see [Riv74]).

Property 8.1 *Let $f \in C^0([a,b])$ and* X *be an interpolation matrix on* $[a,b]$. *Then*

$$E_{n,\infty}(\mathrm{X}) \leq E_n^* \left(1 + \Lambda_n(\mathrm{X})\right), \qquad n = 0, 1, \ldots \tag{8.10}$$

where $\Lambda_n(\mathrm{X})$ *denotes the Lebesgue constant of* X, *defined as*

$$\Lambda_n(\mathrm{X}) = \left\| \sum_{j=0}^{n} |l_j^{(n)}| \right\|_\infty, \tag{8.11}$$

and where $l_j^{(n)} \in \mathbb{P}_n$ *is the* j*-th characteristic polynomial associated with the* $n+1$*-th row of* X, *that is, satisfying* $l_j^{(n)}(x_{nk}) = \delta_{jk}$, $j, k = 0, 1, \ldots$

Since E_n^* does not depend on X, all the information concerning the effects of X on $E_{n,\infty}(\mathrm{X})$ must be looked for in $\Lambda_n(\mathrm{X})$. Although there exists an interpolation matrix X^* such that $\Lambda_n(\mathrm{X})$ is minimized, it is not in general a simple task to determine its entries explicitly. We shall see in Section 10.3, that the zeros of the Chebyshev polynomials provide on the interval $[-1, 1]$ an interpolation matrix with a very small value of the Lebesgue constant.

On the other hand, for any possible choice of X, there exists a constant $C > 0$ such that (see [Erd61])

$$\Lambda_n(\mathrm{X}) > \frac{2}{\pi} \log(n+1) - C, \qquad n = 0, 1, \ldots$$

This property shows that $\Lambda_n(\mathrm{X}) \to \infty$ as $n \to \infty$. This fact has important consequences: in particular, it can be proved (see [Fab14]) that, given an interpolation matrix X on an interval $[a,b]$, there always exists a continuous function f in $[a,b]$, such that $\Pi_n f$ does not converge uniformly (that is, in the maximum norm) to f. Thus, polynomial interpolation does not allow for approximating *any* continuous function, as demonstrated by the following example.

Example 8.1 (Runge's counterexample) Suppose we approximate the following function

$$f(x) = \frac{1}{1+x^2}, \qquad -5 \leq x \leq 5 \tag{8.12}$$

using Lagrange interpolation on equally spaced nodes. It can be checked that some points x exist within the interpolation interval such that

$$\lim_{n \to \infty} |f(x) - \Pi_n f(x)| \neq 0.$$

In particular, Lagrange interpolation diverges for $|x| > 3.63\ldots$. This phenomenon is particularly evident in the neighborhood of the end points of the interpolation interval, as shown in Figure 8.2, and is due to the choice of equally spaced nodes. We shall see in Chapter 10 that resorting to suitably chosen nodes will allow for uniform convergence of the interpolating polynomial to the function f to hold. •

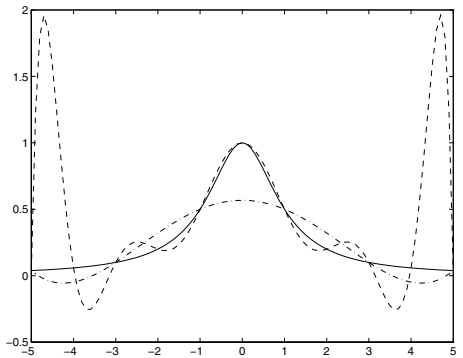

FIGURE 8.2. Lagrange interpolation on equally spaced nodes for the function $f(x) = 1/(1+x^2)$: the interpolating polynomials $\Pi_5 f$ and $\Pi_{10} f$ are shown in dotted and dashed line, respectively

8.1.3 Stability of Polynomial Interpolation

Let us consider a set of function values $\left\{\widetilde{f}(x_i)\right\}$ which is a perturbation of the data $f(x_i)$ relative to the nodes x_i, with $i = 0, \ldots, n$, in an interval $[a, b]$. The perturbation may be due, for instance, to the effect of rounding errors, or may be caused by an error in the experimental measure of the data.

Denoting by $\Pi_n \widetilde{f}$ the interpolating polynomial on the set of values $\widetilde{f}(x_i)$, we have

$$\begin{aligned} \|\Pi_n f - \Pi_n \widetilde{f}\|_\infty &= \max_{a \le x \le b} \left| \sum_{j=0}^{n} (f(x_j) - \widetilde{f}(x_j)) l_j(x) \right| \\ &\le \Lambda_n(\mathrm{X}) \max_{i=0,\ldots,n} |f(x_i) - \widetilde{f}(x_i)|. \end{aligned}$$

As a consequence, small changes on the data give rise to small changes on the interpolating polynomial only if the Lebesgue constant is small. This constant plays the role of the *condition number* for the interpolation problem.

As previously noticed, Λ_n grows as $n \to \infty$ and in particular, in the case of Lagrange interpolation on equally spaced nodes, it can be proved that (see [Nat65])

$$\Lambda_n(\mathrm{X}) \simeq \frac{2^{n+1}}{en \log n}$$

where $e \simeq 2.7183$ is the naeperian number. This shows that, for n large, this form of interpolation can become unstable. Notice also that so far we have completely neglected the errors generated by the interpolation process in constructing $\Pi_n f$. However, it can be shown that the effect of such errors is generally negligible (see [Atk89]).

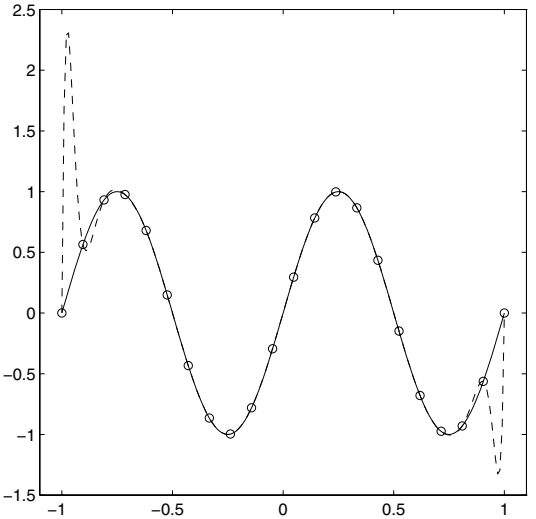

FIGURE 8.3. Instability of Lagrange interpolation. In solid line $\Pi_{21} f$, on unperturbed data, in dashed line $\Pi_{21} \widetilde{f}$, on perturbed data, for Example 8.2

Example 8.2 On the interval $[-1, 1]$ let us interpolate the function $f(x) = \sin(2\pi x)$ at 22 equally spaced nodes x_i. Next, we generate a perturbed set of values $\widetilde{f}(x_i)$ of the function evaluations $f(x_i) = \sin(2\pi x_i)$ with $\max_{i=0,\dots,21} |f(x_i) - \widetilde{f}(x_i)| \simeq 9.5 \cdot 10^{-4}$. In Figure 8.3 we compare the polynomials $\Pi_{21} f$ and $\Pi_{21} \widetilde{f}$: notice how the difference between the two interpolating polynomials, around the end points of the interpolation interval, is quite larger than the impressed perturbation (actually, $\|\Pi_{21} f - \Pi_{21} \widetilde{f}\|_\infty \simeq 2.1635$ and $\Lambda_{21} \simeq 24000$). •

8.2 Newton Form of the Interpolating Polynomial

The Lagrange form (8.4) of the interpolating polynomial is not the most convenient from a practical standpoint. In this section we introduce an alternative form characterized by a cheaper computational cost. Our goal is the following:

given $n + 1$ pairs $\{x_i, y_i\}$, $i = 0, \dots, n$, we want to represent Π_n (with $\Pi_n(x_i) = y_i$ for $i = 0, \dots, n$) as the sum of Π_{n-1} (with $\Pi_{n-1}(x_i) = y_i$ for $i = 0, \dots, n-1$) and a polynomial of degree n which depends on the nodes x_i and on only one unknown coefficient. We thus set

$$\Pi_n(x) = \Pi_{n-1}(x) + q_n(x), \tag{8.13}$$

where $q_n \in \mathbb{P}_n$. Since $q_n(x_i) = \Pi_n(x_i) - \Pi_{n-1}(x_i) = 0$ for $i = 0, \dots, n-1$, it must necessarily be that

$$q_n(x) = a_n(x - x_0) \dots (x - x_{n-1}) = a_n \omega_n(x).$$

To determine the unknown coefficient a_n, suppose that $y_i = f(x_i)$, $i = 0, \ldots, n$, where f is a suitable function, not necessarily known in explicit form. Since $\Pi_n f(x_n) = f(x_n)$, from (8.13) it follows that

$$a_n = \frac{f(x_n) - \Pi_{n-1} f(x_n)}{\omega_n(x_n)}. \tag{8.14}$$

The coefficient a_n is called n-th the *Newton divided difference* and is generally denoted by

$$a_n = f[x_0, x_1, \ldots, x_n] \tag{8.15}$$

for $n \geq 1$. As a consequence, (8.13) becomes

$$\Pi_n f(x) = \Pi_{n-1} f(x) + \omega_n(x) f[x_0, x_1, \ldots, x_n]. \tag{8.16}$$

If we let $y_0 = f(x_0) = f[x_0]$ and $\omega_0 = 1$, by recursion on n we can obtain from (8.16) the following formula

$$\Pi_n f(x) = \sum_{k=0}^{n} \omega_k(x) f[x_0, \ldots, x_k]. \tag{8.17}$$

Uniqueness of the interpolating polynomial ensures that the above expression yields the same interpolating polynomial generated by the Lagrange form. Form (8.17) is commonly known as the *Newton divided difference formula* for the interpolating polynomial.

Program 65 provides an implementation of Newton's formula. The input vectors `x` and `y` contain the interpolation nodes and the corresponding functional evaluations of f, respectively, while vector `z` contains the abscissae where the polynomial $\Pi_n f$ is to be evaluated. This polynomial is stored in the output vector `f`.

Program 65 - interpol : Lagrange polynomial using Newton's formula

```
function [f] = interpol (x,y,z)
[m n] = size(y);
for j = 1:m
  a (:,1) = y (j,:)';
  for i = 2:n
    a (i:n,i) = ( a(i:n,i-1)-a(i-1,i-1) )./(x(i:n)-x(i-1))';
  end
  f(j,:) = a(n,n).*(z-x(n-1)) + a(n-1,n-1);
  for i = 2:n-1
    f(j,:) = f(j,:).*(z-x(n-i))+a(n-i,n-i);
  end
end
```

8.2.1 Some Properties of Newton Divided Differences

The n-th divided difference $f[x_0, \dots, x_n] = a_n$ can be further characterized by noticing that it is the coefficient of x^n in $\Pi_n f$. Isolating such a coefficient from (8.5) and equating it with the corresponding coefficient in the Newton formula (8.17), we end up with the following explicit representation

$$f[x_0, \dots, x_n] = \sum_{i=0}^{n} \frac{f(x_i)}{\omega'_{n+1}(x_i)}. \tag{8.18}$$

This formula has remarkable consequences:

1. the value attained by the divided difference is invariant with respect to permutations of the indexes of the nodes. This instance can be profitably employed when stability problems suggest exchanging the indexes (for example, if x is the point where the polynomial must be computed, it is convenient to introduce a permutation of the indexes such that $|x - x_k| \leq |x - x_{k-1}|$ with $k = 1, \dots, n$);

2. if $f = \alpha g + \beta h$ for some $\alpha, \beta \in \mathbb{R}$, then

$$f[x_0, \dots, x_n] = \alpha g[x_0, \dots, x_n] + \beta h[x_0, \dots, x_n];$$

3. if $f = gh$, the following formula (called the Leibniz formula) holds (see [Die93])

$$f[x_0, \dots, x_n] = \sum_{j=0}^{n} g[x_0, \dots, x_j] h[x_j, \dots, x_n];$$

4. an algebraic manipulation of (8.18) (see Exercise 7) yields the following *recursive formula* for computing divided differences

$$f[x_0, \dots, x_n] = \frac{f[x_1, \dots, x_n] - f[x_0, \dots, x_{n-1}]}{x_n - x_0}, \quad n \geq 1. \tag{8.19}$$

Program 66 implements the recursive formula (8.19). The evaluations of f at the interpolation nodes `x` are stored in vector `y`, while the output matrix `d` (lower triangular) contains the divided differences, which are stored in the following form

$$\begin{array}{c|ccccc} x_0 & f[x_0] & & & & \\ x_1 & f[x_1] & f[x_0, x_1] & & & \\ x_2 & f[x_2] & f[x_1, x_2] & f[x_0, x_1, x_2] & & \\ \vdots & \vdots & & \vdots & \ddots & \\ x_n & f[x_n] & f[x_{n-1}, x_n] & f[x_{n-2}, x_{n-1}, x_n] & \dots & f[x_0, \dots, x_n] \end{array}$$

The coefficients involved in the Newton formula are the diagonal entries of the matrix.

Program 66 - dividif : Newton divided differences

```
function [d]=dividif(x,y)
[n,m]=size(y);
if n == 1, n = m; end
n = n-1;    d = zeros (n+1,n+1);   d (:,1) = y';
for j = 2:n+1
  for i = j:n+1
    d (i,j) = ( d (i-1,j-1)-d (i,j-1))/(x (i-j+1)-x (i));
  end
end
```

Using (8.19), $n(n+1)$ sums and $n(n+1)/2$ divisions are needed to generate the whole matrix. If a new evaluation of f were available at a new node x_{n+1}, only the calculation of a new row of the matrix would be required ($f[x_n, x_{n+1}]$, $\ldots$, $f[x_0, x_1, \ldots, x_{n+1}]$). Thus, in order to construct $\Pi_{n+1}f$ from $\Pi_n f$, it suffices to add to $\Pi_n f$ the term $a_{n+1}\omega_{n+1}(x)$, with a computational cost of $(n+1)$ divisions and $2(n+1)$ sums. For the sake of notational simplicity, we write below $D^r f_i = f[x_i, x_{i+1}, \ldots, x_r]$.

Example 8.3 In Table 8.1 we show the divided differences on the interval (0,2) for the function $f(x) = 1+\sin(3x)$. The values of f and the corresponding divided differences have been computed using 16 significant figures, although only the first 5 figures are reported. If the value of f were available at node $x = 0.2$, updating the divided difference table would require only to computing the entries denoted by italics in Table 8.1. •

x_i	$f(x_i)$	$f[x_i, x_{i-1}]$	$D^2 f_i$	$D^3 f_i$	$D^4 f_i$	$D^5 f_i$	$D^6 f_i$
0	1.0000						
0.2	*1.5646*	*2.82*					
0.4	1.9320	*1.83*	*-2.46*				
0.8	1.6755	-0.64	*-4.13*	*-2.08*			
1.2	0.5575	-2.79	-2.69	*1.43*	*2.93*		
1.6	0.0038	-1.38	1.76	3.71	*1.62*	*-0.81*	
2.0	0.7206	1.79	3.97	1.83	-1.17	*-1.55*	*-0.36*

TABLE 8.1. Divided differences for the function $f(x) = 1 + \sin(3x)$ in the case in which the evaluation of f at $x = 0.2$ is also available. The newly computed values are denoted by italics

Notice that $f[x_0, \ldots, x_n] = 0$ for any $f \in \mathbb{P}_{n-1}$. This property, however, is not always verified numerically, since the computation of divided differences might be highly affected by rounding errors.

Example 8.4 Consider again the divided differences for the function $f(x) = 1 + \sin(3x)$ on the interval $(0, 0.0002)$. The function behaves like $1 + 3x$ in a sufficiently small neighbourhood of 0, so that we expect to find smaller numbers as the order of divided differences increases. However, the results obtained running Program 66, and shown in Table 8.2 in exponential notation up to the first 4 significant figures (although 16 digits have been employed in the calculations), exhibit a substantially different pattern. The small rounding errors introduced in the computation of divided differences of low order have dramatically propagated on the higher order divided differences. •

x_i	$f(x_i)$	$f[x_i, x_{i-1}]$	$D^2 f_i$	$D^3 f_i$	$D^4 f_i$	$D^5 f_i$
0	1.0000					
4.0e-5	1.0001	3.000				
8.0e-5	1.0002	3.000	-5.39e-4			
1.2e-4	1.0004	3.000	-1.08e-3	-4.50		
1.6e-4	1.0005	3.000	-1.62e-3	-4.49	1.80e+1	
2.0e-4	1.0006	3.000	-2.15e-3	-4.49	-7.23	$\boxed{-1.2e+5}$

TABLE 8.2. Divided differences for the function $f(x) = 1+\sin(3x)$ on the interval (0,0.0002). Notice the completely wrong value in the last column (it should be approximately equal to 0), due to the propagation of rounding errors throughout the algorithm

8.2.2 The Interpolation Error Using Divided Differences

Consider the nodes $x_0, \ldots, x_n$ and let $\Pi_n f$ be the interpolating polynomial of f on such nodes. Now let x be a node distinct from the previous ones; letting $x_{n+1} = x$, we denote by $\Pi_{n+1} f$ the interpolating polynomial of f at the nodes x_k, $k = 0, \ldots, n+1$. Using the Newton divided differences formula, we get

$$\Pi_{n+1} f(t) = \Pi_n f(t) + (t - x_0) \ldots (t - x_n) f[x_0, \ldots, x_n, t].$$

Since $\Pi_{n+1} f(x) = f(x)$, we obtain the following formula for the interpolation error at $t = x$

$$\begin{aligned} E_n(x) &= f(x) - \Pi_n f(x) = \Pi_{n+1} f(x) - \Pi_n f(x) \\ &= (x - x_0) \ldots (x - x_n) f[x_0, \ldots, x_n, x] \\ &= \omega_{n+1}(x) f[x_0, \ldots, x_n, x]. \end{aligned} \tag{8.20}$$

Assuming $f \in C^{(n+1)}(I_x)$ and comparing (8.20) with (8.7), yields

$$f[x_0, \ldots, x_n, x] = \frac{f^{(n+1)}(\xi)}{(n+1)!} \tag{8.21}$$

for a suitable $\xi \in I_x$. Since (8.21) resembles the remainder of the Taylor series expansion of f, the Newton formula (8.17) for the interpolating polynomial is often regarded as being a truncated expansion around x_0 provided that $|x_n - x_0|$ is not too big.

8.3 Piecewise Lagrange Interpolation

In Section 8.1.1 we have outlined the fact that, for equally spaced interpolating nodes, uniform convergence of $\Pi_n f$ to f is not guaranteed as $n \to \infty$. On the other hand, using equally spaced nodes is clearly computationally convenient and, moreover, Lagrange interpolation of low degree is sufficiently accurate, provided sufficiently small interpolation intervals are considered.

Therefore, it is natural to introduce a partition $\mathcal{T}_h$ of $[a, b]$ into K subintervals $I_j = [x_j, x_{j+1}]$ of length h_j, with $h = \max_{0 \le j \le K-1} h_j$, such that $[a, b] = \cup_{j=0}^{K-1} I_j$ and then to employ Lagrange interpolation on each I_j using $n+1$ equally spaced nodes $\left\{x_j^{(i)},\ 0 \le i \le n\right\}$ with a small n.

For $k \ge 1$, we introduce on $\mathcal{T}_h$ the piecewise polynomial space

$$X_h^k = \left\{v \in C^0([a, b]) :\ v|_{I_j} \in \mathbb{P}_k(I_j)\, \forall I_j \in \mathcal{T}_h\right\} \tag{8.22}$$

which is the space of the continuous functions over $[a, b]$ whose restrictions on each I_j are polynomials of degree $\le k$. Then, for any continuous function f in $[a, b]$, the *piecewise interpolation polynomial* $\Pi_h^k f$ coincides on each I_j with the interpolating polynomial of $f_{|I_j}$ at the $n+1$ nodes $\left\{x_j^{(i)},\ 0 \le i \le n\right\}$. As a consequence, if $f \in C^{k+1}([a, b])$, using (8.7) within each interval we obtain the following error estimate

$$\|f - \Pi_h^k f\|_\infty \le C h^{k+1}\, \|f^{(k+1)}\|_\infty. \tag{8.23}$$

Note that a small interpolation error can be obtained even for low k provided that h is sufficiently "small".

Example 8.5 Let us go back to the function of Runge's counterexample. Now, piecewise polynomials of degree $k = 1$ and $k = 2$ are employed. We check experimentally for the behavior of the error as h decreases. In Table 8.3 we show the absolute errors measured in the maximum norm over the interval $[-5, 5]$ and the corresponding estimates of the convergence order p with respect to h. Except when using an excessively small number of subintervals, the results confirm the theoretical estimate (8.23), that is $p = k + 1$. •

h	$\|f-\Pi_1^h\|_\infty$	p	$\|f-\Pi_2^h\|_\infty$	p
5	0.4153		0.0835	
2.5	0.1787	1.216	0.0971	-0.217
1.25	0.0631	1.501	0.0477	1.024
0.625	0.0535	0.237	0.0082	2.537
0.3125	0.0206	1.374	0.0010	3.038
0.15625	0.0058	1.819	1.3828e-04	2.856
0.078125	0.0015	1.954	1.7715e-05	2.964

TABLE 8.3. Interpolation error for Lagrange piecewise interpolation of degree $k = 1$ and $k = 2$, in the case of Runge's function (8.12); p denotes the trend of the exponent of h. Notice that, as $h \to 0$, $p \to k + 1$, as predicted by (8.23)

Besides estimate (8.23), convergence results in integral norms exist (see [QV94], [EEHJ96]). For this purpose, we introduce the following space

$$\mathrm{L}^2(a,b) = \left\{ f : (a,b) \to \mathbb{R},\ \int_a^b |f(x)|^2 dx < +\infty \right\}, \tag{8.24}$$

with

$$\|f\|_{\mathrm{L}^2(a,b)} = \left(\int_a^b |f(x)|^2 dx \right)^{1/2}. \tag{8.25}$$

Formula (8.25) defines a norm for $\mathrm{L}^2(a,b)$. (We recall that norms and seminorms of functions can be defined in a manner similar to what was done in Definition 1.17 in the case of vectors). We warn the reader that the integral of the function $|f|^2$ in (8.24) has to be intended in the Lebesgue sense (see, e.g., [Rud83]). In particular, f needs not be continuous everywhere.

Theorem 8.3 *Let $0 \le m \le k+1$, with $k \ge 1$ and assume that $f^{(m)} \in \mathrm{L}^2(a,b)$ for $0 \le m \le k+1$; then there exists a positive constant C, independent of h, such that*

$$\|(f - \Pi_h^k f)^{(m)}\|_{\mathrm{L}^2(a,b)} \le C h^{k+1-m} \|f^{(k+1)}\|_{\mathrm{L}^2(a,b)}. \tag{8.26}$$

In particular, for $k = 1$, and $m = 0$ or $m = 1$, we obtain

$$\begin{aligned} &\|f - \Pi_h^1 f\|_{\mathrm{L}^2(a,b)} \le C_1 h^2 \|f''\|_{\mathrm{L}^2(a,b)}, \\ &\|(f - \Pi_h^1 f)'\|_{\mathrm{L}^2(a,b)} \le C_2 h \|f''\|_{\mathrm{L}^2(a,b)}, \end{aligned} \tag{8.27}$$

for two suitable positive constants C_1 and C_2.

Proof. We only prove (8.27) and refer to [QV94], Chapter 3 for the proof of (8.26) in the general case.

Define $e = f - \Pi_h^1 f$. Since $e(x_j) = 0$ for all $j = 0, \ldots, K$, Rolle's theorem infers the existence of $\xi_j \in (x_j, x_{j+1})$, for $j = 0, \ldots, K-1$ such that $e'(\xi_j) = 0$.

Since $\Pi_h^1 f$ is a linear function on each I_j, for $x \in I_j$ we obtain

$$e'(x) = \int_{\xi_j}^{x} e''(s)ds = \int_{\xi_j}^{x} f''(s)ds,$$

whence

$$|e'(x)| \leq \int_{x_j}^{x_{j+1}} |f''(s)|ds, \qquad \text{for } x \in [x_j, x_{j+1}]. \tag{8.28}$$

We recall the *Cauchy-Schwarz inequality*

$$\left| \int_\alpha^\beta u(x)v(x)dx \right| \leq \left(\int_\alpha^\beta u^2(x)dx \right)^{1/2} \left(\int_\alpha^\beta v^2(x)dx \right)^{1/2} \tag{8.29}$$

which holds if $u, v \in \mathrm{L}^2(\alpha, \beta)$. If we apply this inequality to (8.28) we obtain

$$\begin{aligned} |e'(x)| &\leq \left(\int_{x_j}^{x_{j+1}} 1^2 dx \right)^{1/2} \left(\int_{x_j}^{x_{j+1}} |f''(s)|^2 ds \right)^{1/2} \\ &\leq h^{1/2} \left(\int_{x_j}^{x_{j+1}} |f''(s)|^2 ds \right)^{1/2}. \end{aligned} \tag{8.30}$$

To find a bound for $|e(x)|$, we notice that

$$e(x) = \int_{x_j}^{x} e'(s)ds,$$

so that, applying (8.30), we get

$$|e(x)| \leq \int_{x_j}^{x_{j+1}} |e'(s)|ds \leq h^{3/2} \left(\int_{x_j}^{x_{j+1}} |f''(s)|^2 ds \right)^{1/2}. \tag{8.31}$$

Then

$$\int_{x_j}^{x_{j+1}} |e'(x)|^2 dx \leq h^2 \int_{x_j}^{x_{j+1}} |f''(s)|^2 ds \quad \text{and} \quad \int_{x_j}^{x_{j+1}} |e(x)|^2 dx \leq h^4 \int_{x_j}^{x_{j+1}} |f''(s)|^2 ds,$$

from which, summing over the index j from 0 to $K-1$ and taking the square root of both sides, we obtain

$$\left(\int_a^b |e'(x)|^2 dx \right)^{1/2} \leq h \left(\int_a^b |f''(x)|^2 dx \right)^{1/2},$$

and

$$\left(\int_a^b |e(x)|^2 dx \right)^{1/2} \leq h^2 \left(\int_a^b |f''(x)|^2 dx \right)^{1/2},$$

which is the desired estimate (8.27), with $C_1 = C_2 = 1$. ◇

8.4 Hermite-Birkoff Interpolation

Lagrange polynomial interpolation can be generalized to the case in which also the values of the derivatives of a function f are available at some (or all) of the nodes x_i.

Let us then suppose that $(x_i, f^{(k)}(x_i))$ are given data, with $i = 0, \dots, n$, $k = 0, \dots, m_i$ and $m_i \in \mathbb{N}$. Letting $N = \sum_{i=0}^{n}(m_i + 1)$, it can be proved (see [Dav63]) that, if the nodes $\{x_i\}$ are distinct, there exists a unique polynomial $H_{N-1} \in \mathbb{P}_{N-1}$, called the *Hermite interpolation polynomial*, such that

$$H_{N-1}^{(k)}(x_i) = y_i^{(k)}, \quad i = 0, \dots, n \quad k = 0, \dots, m_i,$$

of the form

$$H_{N-1}(x) = \sum_{i=0}^{n}\sum_{k=0}^{m_i} y_i^{(k)} L_{ik}(x) \tag{8.32}$$

where $y_i^{(k)} = f^{(k)}(x_i)$, $i = 0, \dots, n$, $k = 0, \dots, m_i$.
The functions $L_{ik} \in \mathbb{P}_{N-1}$ are called the *Hermite characteristic polynomials* and are defined through the relations

$$\frac{d^p}{dx^p}(L_{ik})(x_j) = \begin{cases} 1 & \text{if } i = j \text{ and } k = p, \\ 0 & \text{otherwise.} \end{cases}$$

Defining the polynomials

$$l_{ij}(x) = \frac{(x - x_i)^j}{j!} \prod_{\substack{k=0 \\ k \neq i}}^{n} \left(\frac{x - x_k}{x_i - x_k} \right)^{m_k+1}, \ i = 0, \dots, n, \ j = 0, \dots, m_i,$$

and letting $L_{im_i}(x) = l_{im_i}(x)$ for $i = 0, \dots, n$, we have the following recursive formula for the polynomials L_{ij}

$$L_{ij}(x) = l_{ij}(x) - \sum_{k=j+1}^{m_i} l_{ij}^{(k)}(x_i) L_{ik}(x) \qquad j = m_i - 1, m_i - 2, \dots, 0.$$

As for the interpolation error, the following estimate holds

$$f(x) - H_{N-1}(x) = \frac{f^{(N)}(\xi)}{N!} \Omega_N(x) \quad \forall x \in \mathbb{R}$$

where $\xi \in \mathrm{I}(x; x_0, \dots, x_n)$ and Ω_N is the polynomial of degree N defined by

$$\Omega_N(x) = (x - x_0)^{m_0+1}(x - x_1)^{m_1+1} \dots (x - x_n)^{m_n+1}. \tag{8.33}$$

Example 8.6 (osculatory interpolation) Let us set $m_i = 1$ for $i = 0, \ldots, n$. In this case $N = 2n + 2$ and the interpolating Hermite polynomial is called the *osculating polynomial*, and it is given by

$$H_{N-1}(x) = \sum_{i=0}^{n} \Big(y_i A_i(x) + y_i^{(1)} B_i(x) \Big)$$

where $A_i(x) = (1 - 2(x - x_i) l_i'(x_i)) l_i(x)^2$ and $B_i(x) = (x - x_i) l_i(x)^2$, for $i = 0, \ldots, n$, with

$$l_i'(x_i) = \sum_{k=0, k \neq i}^{n} \frac{1}{x_i - x_k}, \qquad i = 0, \ldots, n.$$

As a comparison, we use Programs 65 and 67 to compute the Lagrange and Hermite interpolating polynomials of the function $f(x) = \sin(4\pi x)$ on the interval $[0, 1]$ taking four equally spaced nodes ($n = 3$). Figure 8.4 shows the superposed graphs of the function f (dashed line) and of the two polynomials $\Pi_n f$ (dotted line) and H_{N-1} (solid line). •

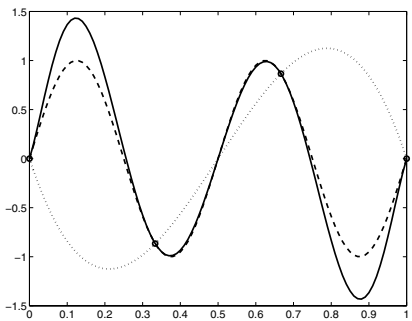

FIGURE 8.4. Lagrange and Hermite interpolation for the function $f(x) = \sin(4\pi x)$ on the interval $[0, 1]$

Program 67 computes the values of the osculating polynomial at the abscissae contained in the vector `z`. The input vectors `x`, `y` and `dy` contain the interpolation nodes and the corresponding function evaluations of f and f', respectively.

Program 67 - hermpol : Osculating polynomial

```
function [herm] = hermite(x,y,dy,z)
n = max(size(x)); m = max(size(z)); herm = [];
for j = 1:m
  xx = z(j); hxv = 0;
  for i = 1:n,
    den = 1; num = 1; xn = x(i); derLi = 0;
    for k = 1:n,
```

```
      if k ~= i, num = num*(xx-x(k)); arg = xn-x(k);
        den = den*arg; derLi = derLi+1/arg;
      end
    end
    Lix2 = (num/den)^2; p = (1-2*(xx-xn)*derLi)*Lix2;
    q = (xx-xn)*Lix2; hxv = hxv+(y(i)*p+dy(i)*q);
   end
   herm = [herm, hxv];
end
```

8.5 Extension to the Two-Dimensional Case

In this section we briefly address the extension of the previous concepts to the two-dimensional case, referring to [SL89], [CHQZ88], [QV94] for more details. We denote by Ω a bounded domain in $\mathbb{R}^2$ and by $\mathbf{x} = (x, y)$ the coordinate vector of a point in Ω.

8.5.1 Polynomial Interpolation

A particularly simple situation occurs when $\Omega = [a, b] \times [c, d]$, i.e., the interpolation domain Ω is the tensor product of two intervals. In such a case, introducing the nodes $a = x_0 < x_1 < \ldots < x_n = b$ and $c = y_0 < y_1 < \ldots < y_m = d$, the interpolating polynomial $\Pi_{n,m} f$ can be written as $\Pi_{n,m} f(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{m} \alpha_{ij} l_i(x) l_j(y)$, where $l_i \in \mathbb{P}_n$, $i = 0, \ldots, n$, and $l_j \in \mathbb{P}_m$, $j = 0, \ldots, m$, are the characteristic one-dimensional Lagrange polynomials with respect to the x and y variables respectively, and where $\alpha_{ij} = f(x_i, y_j)$.

The drawbacks of one-dimensional Lagrange interpolation are inherited by the two-dimensional case, as confirmed by the example in Figure 8.5.

Remark 8.1 (The general case) If Ω is not a rectangular domain or if the interpolation nodes are not uniformly distributed over a Cartesian grid, the interpolation problem is difficult to solve, and, generally speaking, it is preferable to resort to a least-squares solution (see Section 10.7). We also point out that in d dimensions (with $d \geq 2$) the problem of finding an interpolating polynomial of degree n with respect to each space variable on $n + 1$ distinct nodes might be ill-posed.

Consider, for example, a polynomial of degree 1 with respect to x and y of the form $p(x, y) = a_3 xy + a_2 x + a_1 y + a_0$ to interpolate a function f at the nodes $(-1, 0)$, $(0, -1)$, $(1, 0)$ and $(0, 1)$. Although the nodes are distinct, the problem (which is nonlinear) does not in general admit a unique solution; actually, imposing the interpolation constraints, we end up with a system that is satisfied by any value of the coefficient a_3. ■

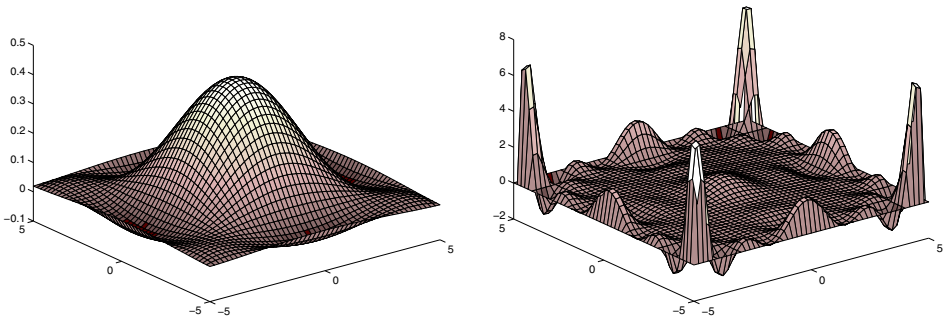

FIGURE 8.5. Runge's counterexample extended to the two-dimensional case: interpolating polynomial on a 6×6 nodes grid (left) and on a 11×11 nodes grid (right). Notice the change in the vertical scale between the two plots

8.5.2 Piecewise Polynomial Interpolation

In the multidimensional case, the higher flexibility of piecewise interpolation allows for easy handling of domains of complex shape. Let us suppose that Ω is a polygon in $\mathbb{R}^2$. Then, Ω can be partitioned into K nonoverlapping triangles (or *elements*) T, which define the so called *triangulation* of the domain which will be denoted by $\mathcal{T}_h$. Clearly, $\overline{\Omega} = \bigcup_{T \in \mathcal{T}_h} T$. Suppose that the maximum length of the edges of the triangles is less than a positive number h. As shown in Figure 8.6 (left), not any arbitrary triangulation is allowed. Precisely, the admissible ones are those for which any pair of non disjoint triangles may have a vertex or an edge in common.

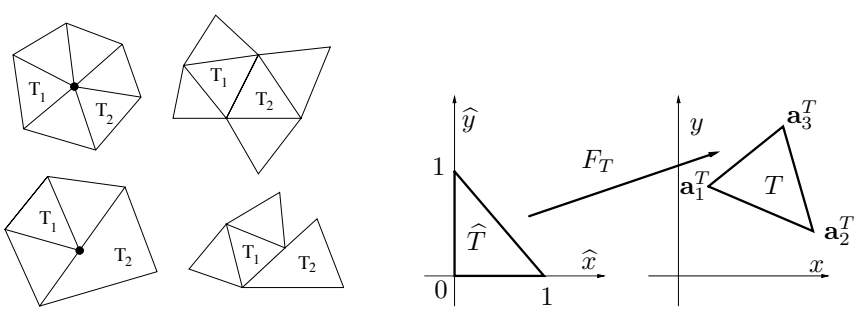

FIGURE 8.6. The left side picture shows admissible (above) and non admissible (below) triangulations while the right side picture shows the affine map from the reference triangle $\hat{T}$ to the generic element $T \in \mathcal{T}_h$

Any element $T \in \mathcal{T}_h$, of area equal to $|T|$, is the image through the affine map $\mathbf{x} = F_T(\hat{\mathbf{x}}) = \mathrm{B}_T\hat{\mathbf{x}} + \mathbf{b}_T$ of the *reference triangle* $\widehat{T}$, of vertices (0,0),

(1,0) and (0,1) in the $\hat{\mathbf{x}} = (\hat{x}, \hat{y})$ plane (see Figure 8.6, right), where the invertible matrix B_T and the right-hand side $\mathbf{b}_T$ are given respectively by

$$\mathrm{B}_T = \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix}, \quad \mathbf{b}_T = (x_1, y_1)^T, \tag{8.34}$$

while the coordinates of the vertices of T are denoted by $\mathbf{a}_T^{(l)} = (x_l, y_l)^T$ for $l = 1, 2, 3$.

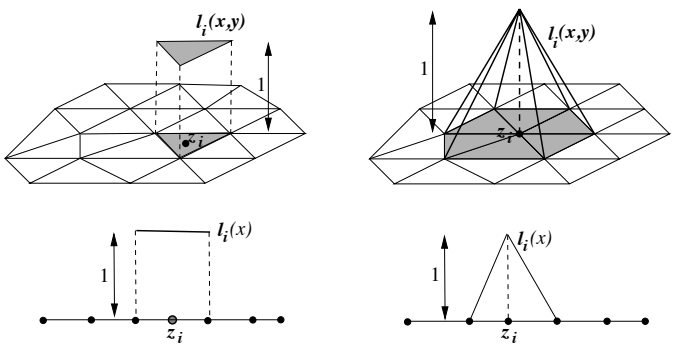

FIGURE 8.7. Characteristic piecewise Lagrange polynomial, in one and two space dimensions. Left, $k = 0$; right, $k = 1$

The affine map (8.34) is of remarkable importance in practical computations, since, once a basis has been generated for representing the piecewise polynomial interpolant on $\hat{T}$, it is possible, applying the change of coordinates $\mathbf{x} = F_T(\hat{\mathbf{x}})$, to reconstruct the polynomial on each element T of $\mathcal{T}_h$. We are thus interested in devising local basis functions, which can be fully described over each triangle without needing any information from adjacent triangles.
For this purpose, let us introduce on $\mathcal{T}_h$ the set $\mathcal{Z}$ of the *piecewise interpolation nodes* $\mathbf{z}_i = (x_i, y_i)^T$, for $i = 1, \ldots, N$, and denote by $\mathbb{P}_k(\Omega)$, $k \geq 0$, the space of algebraic polynomials of degree $\leq k$ in the space variables x, y

$$\mathbb{P}_k(\Omega) = \left\{ p(x,y) = \sum_{\substack{i,j=0 \\ i+j \leq k}}^{k} a_{ij} x^i y^j,\ x, y \in \Omega \right\}. \tag{8.35}$$

Finally, for $k \geq 0$, let $\mathbb{P}_k^c(\Omega)$ be the space of piecewise polynomials of degree $\leq k$, such that, for any $p \in \mathbb{P}_k^c(\Omega)$, $p|_T \in \mathbb{P}_k(T)$ for any $T \in \mathcal{T}_h$. An elementary basis for $\mathbb{P}_k^c(\Omega)$ consists of the *Lagrange characteristic polynomials* $l_i = l_i(x, y)$, such that $l_i \in \mathbb{P}_k^c(\Omega)$ and

$$l_i(\mathbf{z}_j) = \delta_{ij}, \qquad i, j = 1, \ldots, N, \tag{8.36}$$

where δ_{ij} is the Kronecker symbol. We show in Figure 8.7 the functions l_i for $k = 0$, 1, together with their corresponding one-dimensional counterparts. In the case $k = 0$, the interpolation nodes are collocated at the *centers of gravity* of the triangles, while in the case $k = 1$ the nodes coincide with the *vertices* of the triangles. This choice, that we are going to maintain henceforth, is not the only one possible. The midpoints of the edges of the triangles could be used as well, giving rise to a discontinuous piecewise polynomial over Ω.

For $k \geq 0$, the *Lagrange piecewise interpolating polynomial* of f, $\Pi_h^k f \in \mathbb{P}_k^c(\Omega)$, is defined as

$$\Pi_h^k f(x, y) = \sum_{i=1}^{N} f(\mathbf{z}_i) l_i(x, y). \tag{8.37}$$

Notice that $\Pi_h^0 f$ is a piecewise constant function, while $\Pi_h^1 f$ is a linear function over each triangle, continuous at the vertices, and thus globally continuous.

For any $T \in \mathcal{T}_h$, we shall denote by $\Pi_T^k f$ the restriction of the piecewise interpolating polynomial of f over the element T. By definition, $\Pi_T^k f \in \mathbb{P}_k(T)$; noticing that $d_k = \dim\mathbb{P}_k(T) = (k + 1)(k + 2)/2$, we can therefore write

$$\Pi_T^k f(x, y) = \sum_{m=0}^{d_k - 1} f(\tilde{\mathbf{z}}_T^{(m)}) l_{m,T}(x, y), \qquad \forall T \in \mathcal{T}_h. \tag{8.38}$$

In (8.38), we have denoted by $\tilde{\mathbf{z}}_T^{(m)}$, for $m = 0, \ldots, d_k - 1$, the piecewise interpolation nodes on T and by $l_{m,T}(x, y)$ the restriction to T of the Lagrange characteristic polynomial having index i in (8.37) which corresponds in the list of the "global" nodes $\mathbf{z}_i$ to that of the "local" node $\tilde{\mathbf{z}}_T^{(m)}$.

Keeping on with this notation, we have $l_{j,T}(\mathbf{x}) = \hat{l}_j \circ F_T^{-1}(\mathbf{x})$, where $\hat{l}_j = \hat{l}_j(\hat{\mathbf{x}})$ is, for $j = 0, \ldots, d_k - 1$, the j-th Lagrange basis function for $\mathbb{P}_k(\hat{T})$ generated on the reference element $\hat{T}$. We notice that if $k = 0$ then $d_0 = 1$, that is, only one local interpolation node exists (coinciding with the center of gravity of the triangle T), while if $k = 1$ then $d_1 = 3$, that is, three local interpolation nodes exist, coinciding with the vertices of T. In Figure 8.8 we draw the local interpolation nodes on $\hat{T}$ for $k = 0$, 1 and 2.

As for the interpolation error estimate, denoting for any $T \in \mathcal{T}_h$ by h_T the maximum length of the edges of T, the following result holds (see for the proof, [CL91], Theorem 16.1, pp. 125-126 and [QV94], Remark 3.4.2, pp. 89-90)

$$\|f - \Pi_T^k f\|_{\infty,T} \leq C h_T^{k+1} \|f^{(k+1)}\|_{\infty,T}, \qquad k \geq 0, \tag{8.39}$$

where for every $g \in C^0(T)$, $\|g\|_{\infty,T} = \max_{\mathbf{x}\in T} |g(\mathbf{x})|$. In (8.39), C is a positive constant independent of h_T and f.

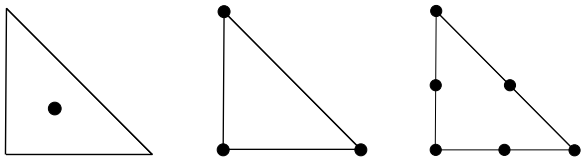

FIGURE 8.8. Local interpolation nodes on $\hat{T}$; left, $k = 0$, center $k = 1$, right, $k = 2$

Let us assume that the triangulation $\mathcal{T}_h$ is *regular*, i.e., there exists a positive constant σ such that

$$\max_{T \in \mathcal{T}_h} \frac{h_T}{\rho_T} \leq \sigma,$$

where $\forall T \in \mathcal{T}_h$, ρ_T is the diameter of the inscribed circle to T, Then, it is possible to derive from (8.39) the following interpolation error estimate over the whole domain Ω

$$\|f - \Pi_h^k f\|_{\infty,\Omega} \leq C h^{k+1} \|f^{(k+1)}\|_{\infty,\Omega}, \qquad k \geq 0, \qquad \forall f \in C^{k+1}(\Omega). \tag{8.40}$$

The theory of piecewise interpolation is a basic tool of the *finite element method*, a computational technique that is widely used in the numerical approximation of partial differential equations (see Chapter 12 for the one-dimensional case and [QV94] for a complete presentation of the method).

Example 8.7 We compare the convergence of the piecewise polynomial interpolation of degree 0, 1 and 2, on the function $f(x, y) = e^{-(x^2+y^2)}$ on $\Omega = (-1, 1)^2$. We show in Table 8.4 the error $\mathcal{E}_k = \|f - \Pi_h^k f\|_{\infty,\Omega}$, for $k = 0, 1, 2$, and the order of convergence p_k as a function of the mesh size $h = 2/N$ for $N = 2, \ldots, 32$. Clearly, linear convergence is observed for interpolation of degree 0 while the order of convergence is quadratic with respect to h for interpolation of degree 1 and cubic for interpolation of degree 2. •

h	$\mathcal{E}_0$	p_0	$\mathcal{E}_1$	p_1	$\mathcal{E}_2$	p_2
1	0.4384		0.2387		0.016	
$\frac{1}{2}$	0.2931	0.5809	0.1037	1.2028	$1.6678 \cdot 10^{-3}$	3.2639
$\frac{1}{4}$	0.1579	0.8924	0.0298	1.7990	$2.8151 \cdot 10^{-4}$	2.5667
$\frac{1}{8}$	0.0795	0.9900	0.0077	1.9524	$3.5165 \cdot 10^{-5}$	3.001
$\frac{1}{16}$	0.0399	0.9946	0.0019	2.0189	$4.555 \cdot 10^{-6}$	2.9486

TABLE 8.4. Convergence rates and orders for piecewise interpolations of degree 0, 1 and 2

8.6 Approximation by Splines

In this section we address the matter of approximating a given function using *splines*, which allow for a piecewise interpolation with a global smoothness.

Definition 8.1 Let $x_0, \ldots, x_n$, be $n+1$ distinct nodes of $[a,b]$, with $a = x_0 < x_1 < \ldots < x_n = b$. The function $s_k(x)$ on the interval [a,b] is a *spline* of degree k relative to the nodes x_j if

$$s_{k|[x_j,x_{j+1}]} \in \mathbb{P}_k, \quad j = 0, 1, \ldots, n-1 \tag{8.41}$$

$$s_k \in C^{k-1}[a,b]. \tag{8.42}$$

■

Denoting by $\mathcal{S}_k$ the space of splines s_k on $[a,b]$ relative to $n+1$ distinct nodes, then $\dim \mathcal{S}_k = n+k$. Obviously, any polynomial of degree k on $[a,b]$ is a spline; however, in the practice a spline is represented by a different polynomial on each subinterval and for this reason there could be a discontinuity in its k-th derivative at the internal nodes $x_1, \ldots, x_{n-1}$. The nodes for which this actually happens are called *active* nodes.

It is simple to check that conditions (8.41) and (8.42) do not suffice to characterize a spline of degree k. Indeed, the restriction $s_{k,j} = s_{k|[x_j,x_{j+1}]}$ can be represented as

$$s_{k,j}(x) = \sum_{i=0}^{k} s_{ij}(x - x_j)^i, \quad \text{if } x \in [x_j, x_{j+1}] \tag{8.43}$$

so that $(k+1)n$ coefficients s_{ij} must be determined. On the other hand, from (8.42) it follows that

$$s_{k,j-1}^{(m)}(x_j) = s_{k,j}^{(m)}(x_j), \quad j = 1, \ldots, n-1, \quad m = 0, ..., k-1$$

which amounts to setting $k(n-1)$ conditions. As a consequence, the remaining degrees of freedom are $(k+1)n - k(n-1) = k+n$.
Even if the spline were *interpolatory*, that is, such that $s_k(x_j) = f_j$ for $j = 0, \ldots, n$, where $f_0, \ldots, f_n$ are given values, there would still be $k-1$ unsaturated degrees of freedom. For this reason further constraints are usually imposed, which lead to:

1. *periodic splines*, if

$$s_k^{(m)}(a) = s_k^{(m)}(b), \quad m = 0, 1, \ldots, k-1; \tag{8.44}$$

2. *natural splines*, if for $k = 2l - 1$, with $l \geq 2$

$$s_k^{(l+j)}(a) = s_k^{(l+j)}(b) = 0, \quad j = 0, 1, \ldots, l-2. \tag{8.45}$$

From (8.43) it turns out that a spline can be conveniently represented using $k+n$ spline basis functions, such that (8.42) is automatically satisfied. The simplest choice, which consists of employing a suitably enriched monomial basis (see Exercise 10), is not satisfactory from the numerical standpoint, since it is ill-conditioned. In Sections 8.6.1 and 8.6.2 possible examples of spline basis functions will be provided: cardinal splines for the specific case $k = 3$ and B-splines for a generic k.

8.6.1 Interpolatory Cubic Splines

Interpolatory cubic splines are particularly significant since: *i.* they are the splines of minimum degree that yield C^2 approximations; *ii.* they are sufficiently smooth in the presence of small curvatures.

Let us thus consider, in $[a, b]$, $n+1$ ordered nodes $a = x_0 < x_1 < \ldots < x_n = b$ and the corresponding evaluations f_i, $i = 0, \ldots, n$. Our aim is to provide an efficient procedure for constructing the cubic spline interpolating those values. Since the spline is of degree 3, its second-order derivative must be continuous. Let us introduce the following notation

$$f_i = s_3(x_i), \quad m_i = s_3'(x_i), \quad M_i = s_3''(x_i), \quad i = 0, \ldots, n.$$

Since $s_{3,i-1} \in \mathbb{P}_3$, $s''_{3,i-1}$ is linear and

$$s''_{3,i-1}(x) = M_{i-1}\frac{x_i - x}{h_i} + M_i\frac{x - x_{i-1}}{h_i} \quad \text{for } x \in [x_{i-1}, x_i] \tag{8.46}$$

where $h_i = x_i - x_{i-1}$. Integrating (8.46) twice we get

$$s_{3,i-1}(x) = M_{i-1}\frac{(x_i - x)^3}{6h_i} + M_i\frac{(x - x_{i-1})^3}{6h_i} + C_{i-1}(x - x_{i-1}) + \widetilde{C}_{i-1},$$

and the constants C_{i-1} and $\widetilde{C}_{i-1}$ are determined by imposing the end point values $s_3(x_{i-1}) = f_{i-1}$ and $s_3(x_i) = f_i$. This yields, for $i = 1, \ldots, n-1$

$$\widetilde{C}_{i-1} = f_{i-1} - M_{i-1}\frac{h_i^2}{6}, \quad C_{i-1} = \frac{f_i - f_{i-1}}{h_i} - \frac{h_i}{6}(M_i - M_{i-1}).$$

Let us now enforce the continuity of the first derivatives at x_i; we get

$$\begin{aligned} s_3'(x_i^-) &= \frac{h_i}{6}M_{i-1} + \frac{h_i}{3}M_i + \frac{f_i - f_{i-1}}{h_i} \\ &= -\frac{h_{i+1}}{3}M_i - \frac{h_{i+1}}{6}M_{i+1} + \frac{f_{i+1} - f_i}{h_{i+1}} = s_3'(x_i^+), \end{aligned}$$

where $s_3'(x_i^{\pm}) = \lim_{t\to 0} s_3'(x_i \pm t)$. This leads to the following linear system (called M-continuity system)

$$\mu_i M_{i-1} + 2M_i + \lambda_i M_{i+1} = d_i \quad i = 1, \dots, n-1 \tag{8.47}$$

where we have set

$$\mu_i = \frac{h_i}{h_i + h_{i+1}}, \qquad \lambda_i = \frac{h_{i+1}}{h_i + h_{i+1}},$$
$$d_i = \frac{6}{h_i + h_{i+1}} \left(\frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i} \right), \quad i = 1, \dots, n-1.$$

System (8.47) has $n+1$ unknowns and $n-1$ equations; thus, $2(= k-1)$ conditions are still lacking. In general, these conditions can be of the form

$$2M_0 + \lambda_0 M_1 = d_0, \quad \mu_n M_{n-1} + 2M_n = d_n,$$

with $0 \le \lambda_0, \mu_n \le 1$ and d_0, d_n given values. For instance, in order to obtain the natural splines (satisfying $s_3''(a) = s_3''(b) = 0$), we must set the above coefficients equal to zero. A popular choice sets $\lambda_0 = \mu_n = 1$ and $d_0 = d_1$, $d_n = d_{n-1}$, which corresponds to prolongating the spline outside the end points of the interval $[a, b]$ and treating a and b as internal points. This strategy produces a spline with a "smooth" behavior. In general, the resulting linear system is tridiagonal of the form

$$\begin{bmatrix} 2 & \lambda_0 & 0 & \dots & 0 \\ \mu_1 & 2 & \lambda_1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \mu_{n-1} & 2 & \lambda_{n-1} \\ 0 & \dots & 0 & \mu_n & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} \tag{8.48}$$

and it can be efficiently solved using the Thomas algorithm (3.53).

A closure condition for system (8.48), which can be useful when the derivatives $f'(a)$ and $f'(b)$ are not available, consists of enforcing the continuity of $s_3'''(x)$ at x_1 and x_{n-1}. Since the nodes x_1 and x_{n-1} do not actually contribute in constructing the cubic spline, it is called a *not-a-knot spline*, with "active" knots $\{x_0, x_2, \dots, x_{n-2}, x_n\}$ and interpolating f at all the nodes $\{x_0, x_1, x_2, \dots, x_{n-2}, x_{n-1}, x_n\}$.

Remark 8.2 (Specific software) Several packages exist for dealing with interpolating splines. In the case of cubic splines, we mention the command `spline`, which uses the *not-a-knot* condition introduced above, or, in general, the `spline toolbox` of MATLAB [dB90] and the library FITPACK [Die87a], [Die87b]. ■

A completely different approach for generating s_3 consists of providing a basis $\{\varphi_i\}$ for the space $\mathcal{S}_3$ of cubic splines, whose dimension is equal to $n+3$. We consider here the case in which the $n+3$ basis functions φ_i have global support in the interval $[a,b]$, referring to Section 8.6.2 for the case of a basis with local support.

Functions φ_i, for $i,j=0,\ldots,n$, are defined through the following interpolation constraints

$$\varphi_i(x_j)=\delta_{ij}, \qquad \varphi_i'(x_0)=\varphi_i'(x_n)=0,$$

and two suitable splines must be added, φ_{n+1} and φ_{n+2}. For instance, if the spline must satisfy some assigned conditions on the derivative at the end points, we ask that

$$\begin{aligned}
&\varphi_{n+1}(x_j)=0, \qquad j=0,...,n \quad \varphi_{n+1}'(x_0)=1, \quad \varphi_{n+1}'(x_n)=0,\\
&\varphi_{n+2}(x_j)=0, \qquad j=0,...,n \quad \varphi_{n+2}'(x_0)=0, \quad \varphi_{n+2}'(x_n)=1.
\end{aligned}$$

By doing so, the spline takes the form

$$s_3(x)=\sum_{i=0}^{n} f_i\varphi_i(x)+f_0'\varphi_{n+1}(x)+f_n'\varphi_{n+2}(x),$$

where f_0' and f_n' are two given values. The resulting basis $\{\varphi_i,\ i=0,...,n+2\}$ is called a *cardinal spline basis* and is frequently employed in the numerical solution of differential or integral equations. Figure 8.9 shows a generic cardinal spline, which is computed over a virtually unbounded interval where the interpolation nodes x_j are the integers. The spline changes sign in any adjacent intervals $[x_{j-1},x_j]$ and $[x_j,x_{j+1}]$ and rapidly decays to zero.

Restricting ourselves to the positive axis, it can be shown (see [SL89]) that the extremant of the function on the interval $[x_j,x_{j+1}]$ is equal to the extremant on the interval $[x_{j+1},x_{j+2}]$ multiplied by a decaying factor $\lambda\in(0,1)$. In such a way, possible errors arising over an interval are rapidly damped on the next one, thus ensuring the stability of the algorithm.

Let us summarize the main properties of interpolating cubic splines, referring to [Sch81] and [dB83] for the proofs and more general results.

Property 8.2 *Let $f\in C^2([a,b])$, and let s_3 be the natural cubic spline interpolating f. Then*

$$\int_a^b [s_3''(x)]^2dx \le \int_a^b [f''(x)]^2dx, \tag{8.49}$$

where equality holds if and only if $f=s_3$.

The above result is known as the *minimum norm property* and has the meaning of the minimum energy principle in mechanics. Property (8.49)

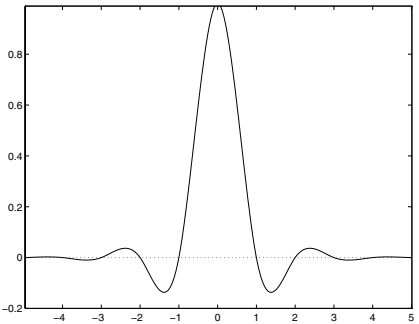

FIGURE 8.9. Cardinal spline

still holds if conditions on the first derivative of the spline at the end points are assigned instead of natural conditions (in such a case, the spline is called *constrained*, see Exercise 11).

The cubic interpolating spline s_f of a function $f \in C^2([a,b])$, with $s'_f(a) = f'(a)$ and $s'_f(b) = f'(b)$, also satisfies the following property

$$\int_a^b [f''(x) - s''_f(x)]^2 dx \leq \int_a^b [f''(x) - s''(x)]^2 dx, \ \forall s \in S_3.$$

As far as the error estimate is concerned, the following result holds.

Property 8.3 *Let $f \in C^4([a,b])$ and fix a partition of $[a,b]$ into subintervals of width h_i such that $h = \max_i h_i$ and $\beta = h/\min_i h_i$. Let s_3 be the cubic spline interpolating f. Then*

$$\|f^{(r)} - s_3^{(r)}\|_\infty \leq C_r h^{4-r} \|f^{(4)}\|_\infty, \qquad r = 0, 1, 2, 3, \tag{8.50}$$

with $C_0 = 5/384$, $C_1 = 1/24$, $C_2 = 3/8$ and $C_3 = (\beta + \beta^{-1})/2$.

As a consequence, spline s_3 and its first and second order derivatives uniformly converge to f and to its derivatives, as h tends to zero. The third order derivative converges as well, provided that β is uniformly bounded.

Example 8.8 Figure 8.10 shows the cubic spline approximating the function in the Runge's example, and its first, second and third order derivatives, on a grid of 11 equally spaced nodes, while in Table 8.5 the error $\|s_3 - f\|_\infty$ is reported as a function of h together with the computed order of convergence p. The results clearly demonstrate that p tends to 4 (the theoretical order) as h tends to zero. •

h	1	0.5	0.25	0.125	0.0625
$\|s_3 - f\|_\infty$	0.022	0.0032	2.7741e-4	1.5983e-5	9.6343e-7
p	–	2.7881	3.5197	4.1175	4.0522

TABLE 8.5. Experimental interpolation error for Runge's function using cubic splines

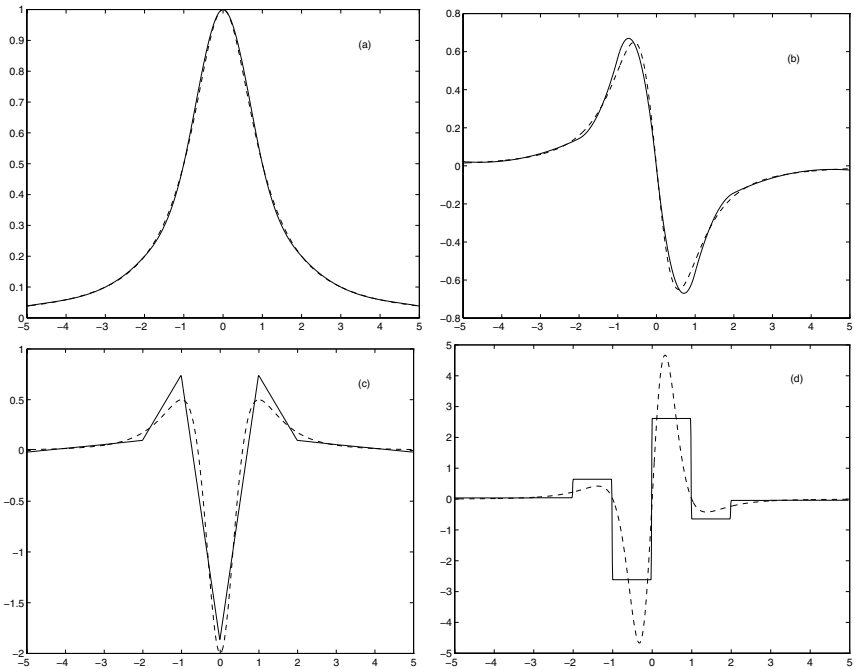

FIGURE 8.10. Interpolating spline (a) and its first (b), second (c) and third (d) order derivatives (in solid line) for the function of Runge's example (in dashed line)

8.6.2 B-splines

Let us go back to splines of a generic degree k, and consider the B-spline (or *bell-spline*) basis, referring to divided differences introduced in Section 8.2.1.

Definition 8.2 The *normalized B-spline* $B_{i,k+1}$ of degree k relative to the distinct nodes $x_i, \ldots, x_{i+k+1}$ is defined as

$$B_{i,k+1}(x) = (x_{i+k+1} - x_i)g[x_i, \ldots, x_{i+k+1}], \tag{8.51}$$

where

$$g(t) = (t-x)_+^k = \begin{cases} (t-x)^k & \text{if } x \le t, \\ 0 & \text{otherwise.} \end{cases} \tag{8.52}$$

■

Substituting (8.18) into (8.51) yields the following explicit representation

$$B_{i,k+1}(x) = (x_{i+k+1} - x_i) \sum_{j=0}^{k+1} \frac{(x_{j+i} - x)_+^k}{\prod_{\substack{l=0 \\ l \neq j}}^{k+1} (x_{i+j} - x_{i+l})}. \tag{8.53}$$

From (8.53) it turns out that the active nodes of $B_{i,k+1}(x)$ are $x_i, \ldots, x_{i+k+1}$ and that $B_{i,k+1}(x)$ is non null only within the interval $[x_i, x_{i+k+1}]$.

Actually, it can be proved that it is the unique non null spline of minimum support relative to nodes $x_i, \ldots, x_{i+k+1}$ [Sch67]. It can also be shown that $B_{i,k+1}(x) \geq 0$ [dB83] and $|B_{i,k+1}^{(l)}(x_i)| = |B_{i,k+1}^{(l)}(x_{i+k+1})|$ for $l = 0, \ldots, k-1$ [Sch81]. B-splines admit the following recursive formulation ([dB72], [Cox72])

$$\begin{aligned} B_{i,1}(x) &= \begin{cases} 1 & \text{if } x \in [x_i, x_{i+1}], \\ 0 & \text{otherwise,} \end{cases} \\ B_{i,k+1}(x) &= \frac{x - x_i}{x_{i+k} - x_i} B_{i,k}(x) + \frac{x_{i+k+1} - x}{x_{i+k+1} - x_{i+1}} B_{i+1,k}(x), \ k \geq 1, \end{aligned} \tag{8.54}$$

which is usually preferred to (8.53) when evaluating a B-spline at a given point.

Remark 8.3 It is possible to define B-splines even in the case of partially coincident nodes, by suitably extending the definition of divided differences. This leads to a new recursive form of Newton divided differences given by (see for further details [Die93])

$$f[x_0, \ldots, x_n] = \begin{cases} \dfrac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0} & \text{if } x_0 < x_1 < \ldots < x_n \\ \dfrac{f^{(n+1)}(x_0)}{(n+1)!} & \text{if } x_0 = x_1 = \ldots = x_n. \end{cases}$$

Assuming that m (with $1 < m < k+2$) of the $k+2$ nodes $x_i, \ldots, x_{i+k+1}$ are coincident and equal to λ, then (8.46) will contain a linear combination of the functions $(\lambda - x)_+^{k+1-j}$, for $j = 1, \ldots, m$. As a consequence, the B-spline can have continuous derivatives at λ only up to order $k - m$ and, therefore, it is discontinuous if $m = k + 1$. It can be checked [Die93] that, if $x_{i-1} < x_i = \ldots = x_{i+k} < x_{i+k+1}$, then

$$B_{i,k+1}(x) = \begin{cases} \left(\dfrac{x_{i+k+1} - x}{x_{i+k+1} - x_i} \right)^k & \text{if } x \in [x_i, x_{i+k+1}], \\ 0 & \text{otherwise,} \end{cases}$$

while for $x_i < x_{i+1} = \ldots = x_{i+k+1} < x_{i+k+2}$

$$B_{i,k+1}(x) = \begin{cases} \left(\dfrac{x - x_i}{x_{i+k+1} - x_i}\right)^k & \text{if } x \in [x_i, x_{i+k+1}], \\ 0 & \text{otherwise.} \end{cases}$$

Combining these formulae with the recursive relation (8.54) allows for constructing B-splines with coincident nodes. ■

Example 8.9 Let us examine the special case of cubic B-splines on equally spaced nodes $x_{i+1} = x_i + h$ for $i = 0, ..., n-1$. Equation (8.53) becomes

$$6h^3 B_{i,4}(x) =$$

$$\begin{cases} (x - x_i)^3, & \text{if } x \in [x_i, x_{i+1}], \\ h^3 + 3h^2(x - x_{i+1}) + 3h(x - x_{i+1})^2 - 3(x - x_{i+1})^3, & \text{if } x \in [x_{i+1}, x_{i+2}], \\ h^3 + 3h^2(x_{i+3} - x) + 3h(x_{i+3} - x)^2 - 3(x_{i+3} - x)^3, & \text{if } x \in [x_{i+2}, x_{i+3}], \\ (x_{i+4} - x)^3, & \text{if } x \in [x_{i+3}, x_{i+4}], \\ 0 & \text{otherwise.} \end{cases}$$

In Figure 8.11 the graph of $B_{i,4}$ is shown in the case of distinct nodes and of partially coincident nodes. •

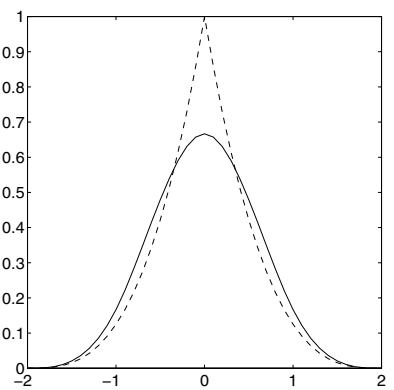

FIGURE 8.11. B-spline with distinct nodes (in solid line) and with three coincident nodes at the origin (in dashed line). Notice the discontinuity of the first derivative

Given $n+1$ distinct nodes x_j, $j = 0, \ldots, n$, $n-k$ linearly independent B-splines of degree k can be constructed, though $2k$ degrees of freedom are

still available to generate a basis for $\mathcal{S}_k$. One way of proceeding consists of introducing $2k$ fictitious nodes

$$\begin{aligned} &x_{-k} \leq x_{-k+1} \leq \ldots \leq x_{-1} \leq x_0 = a, \\ &b = x_n \leq x_{n+1} \leq \ldots \leq x_{n+k} \end{aligned} \tag{8.55}$$

which the B-splines $B_{i,k+1}$, with $i = -k, \ldots, -1$ and $i = n-k, \ldots, n-1$, are associated with. By doing so, any spline $s_k \in \mathcal{S}_k$ can be uniquely written as

$$s_k(x) = \sum_{i=-k}^{n-1} c_i B_{i,k+1}(x). \tag{8.56}$$

The real numbers c_i are the *B-spline coefficients* of s_k. Nodes (8.55) are usually chosen as coincident or periodic.

1. *Coincident*: this choice is suitable for enforcing the values attained by a spline at the end points of its definition interval. In such a case, indeed, thanks to Remark 8.3 about B-splines with coincident nodes, we get

$$s_k(a) = c_{-k}, \quad s_k(b) = c_{n-1}. \tag{8.57}$$

2. *Periodic*, that is

$$x_{-i} = x_{n-i} - b + a, \quad x_{i+n} = x_i + b - a, \quad i = 1, \ldots, k.$$

This choice is useful if the periodicity conditions (8.44) have to be imposed.

Remark 8.4 (Inserting nodes) Using B-splines instead of cardinal splines is advantageous when handling, with a reduced computational effort, a given configuration of nodes for which a spline s_k is known. In particular, assume that the coefficients c_i of s_k (in form (8.56)) are available over the nodes $x_{-k}, x_{-k+1}, \ldots, x_{n+k}$, and that we wish to add to these a new node $\widetilde{x}$.

The spline $\widetilde{s}_k \in \mathcal{S}_k$, defined over the new set of nodes, admits the following representation with respect to a new B-spline basis $\left\{\tilde{B}_{i,k+1}\right\}$

$$\widetilde{s}_k(x) = \sum_{i=-k}^{n-1} d_i \widetilde{B}_{i,k+1}(x).$$

The new coefficients d_i can be computed starting from the known coefficients c_i using the following algorithm [Boe80]:

let $\widetilde{x} \in [x_j, x_{j+1})$; then, construct a new set of nodes $\{y_i\}$ such that

$$y_i = x_i \text{ for } i = -k, \dots, j, \qquad y_{j+1} = \widetilde{x},$$
$$y_i = x_{i-1} \text{ for } i = j+2, \dots, n+k+1;$$

define

$$\omega_i = \begin{cases} 1 & \text{for } i = -k, \dots, j-k, \\ \dfrac{y_{j+1} - y_i}{y_{i+k+1} - y_i} & \text{for } i = j-k+1, \dots, j, \\ 0 & \text{for } i = j+1, \dots, n; \end{cases}$$

compute

$$d_i = \omega_i c_i + (1 - \omega_i) c_i \qquad i = -k, ..., n-1.$$

This algorithm has good stability properties and can be generalized to the case where more than one node is inserted at the same time (see [Die93]). ■

8.7 Splines in Parametric Form

Using interpolating splines presents the following two drawbacks:

1. the resulting approximation is of good quality only if the function f does not exhibit large derivatives (in particular, we require that $|f'(x)| < 1$ for every x). Otherwise, oscillating behaviors may arise in the spline, as demonstrated by the example considered in Figure 8.12 which shows, in solid line, the cubic interpolating spline over the following set of data (from [SL89])

x_i	8.125	8.4	9	9.845	9.6	9.959	10.166	10.2
f_i	0.0774	0.099	0.28	0.6	0.708	1.3	1.8	2.177

2. s_k depends on the choice of the coordinate system. In fact, performing a clockwise rotation of 36 degrees of the coordinate system in the above example, would lead to the spline without spurious oscillations reported in the boxed frame in Figure 8.12.

   All the interpolation procedures considered so far depend on the chosen Cartesian reference system, which is a negative feature if the spline is used for a graphical representation of a given figure (for instance, an ellipse). Indeed, we would like such a representation to be independent of the reference system, that is, to have a geometric invariance property.

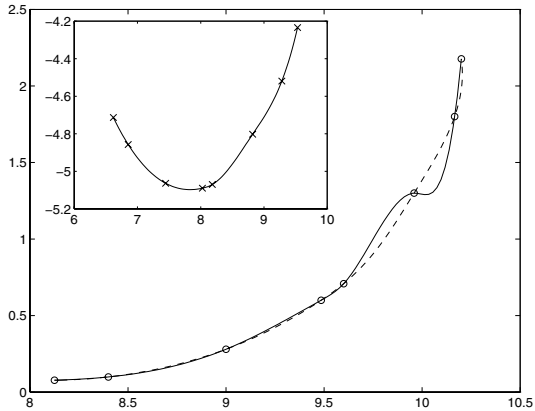

FIGURE 8.12. Geometric noninvariance for an interpolating cubic spline s_3: the set of data for s_3 in the boxed frame is the same as in the main figure, rotated by 36 degrees. The rotation diminishes the slope of the interpolated curve and eliminates any oscillation from s_3. Notice that resorting to a parametric spline (dashed line) removes the oscillations in s_3 without any rotation of the reference system

A solution is provided by parametric splines, in which any component of the curve, written in parametric form, is approximated by a spline function. Consider a plane curve in parametric form $\mathbf{P}(t) = (x(t), y(t))$, with $t \in [0, T]$, then take the set of the points in the plane of coordinates $\mathbf{P}_i = (x_i, y_i)$, for $i = 0, \ldots, n$, and introduce a partition onto $[0, T]$: $0 = t_0 < t_1 < \ldots < t_n = T$.

Using the two sets of values $\{t_i, x_i\}$ and $\{t_i, y_i\}$ as interpolation data, we obtain the two splines $s_{k,x}$ and $s_{k,y}$, with respect to the independent variable t, that interpolate $x(t)$ and $y(t)$, respectively. The parametric curve $\mathbf{S}_k(t) = (s_{k,x}(t), s_{k,y}(t))$ is called the *parametric spline*. Obviously, different parameterizations of the interval $[0, T]$ yield different splines (see Figure 8.13).

A reasonable choice of the parameterization makes use of the length of each segment $\mathbf{P}_{i-1}\mathbf{P}_i$,

$$l_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}, \quad i = 1, \ldots, n.$$

Setting $t_0 = 0$ and $t_i = \sum_{k=1}^{i} l_k$ for $i = 1, \ldots, n$, every t_i represents the cumulative length of the piecewise line that joins the points from $\mathbf{P}_0$ to $\mathbf{P}_i$. This function is called the *cumulative length spline* and approximates satisfactorily even those curves with large curvature. Moreover, it can also be proved (see [SL89]) that it is geometrically invariant.

Program 68 implements the construction of cumulative parametric cubic splines in two dimensions (it can be easily generalized to the three-

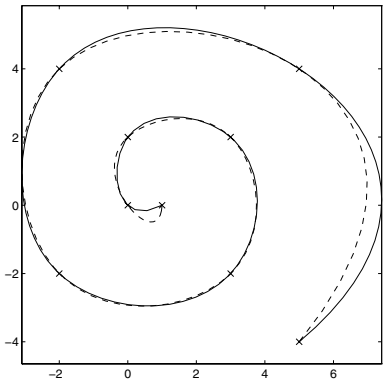

FIGURE 8.13. Parametric splines for a spiral-like node distribution. The spline of cumulative length is drawn in solid line

dimensional case). Composite parametric splines can be generated as well by enforcing suitable continuity conditions (see [SL89]).

Program 68 - par_spline : Parametric splines

```
function [xi,yi] = par_spline (x, y)
t (1) = 0;
for i = 1:length (x)-1
  t (i+1) = t (i) + sqrt ( (x(i+1)-x(i))^2 + (y(i+1)-y(i))^2 );
end
z = [t(1):(t(length(t))-t(1))/100:t(length(t))];
xi = spline (t,x,z);
yi = spline (t,y,z);
```

8.7.1 Bézier Curves and Parametric B-splines

The Bézier curves and parametric B-splines are widely employed in graphical applications, where the nodes' locations might be affected by some uncertainty.

Let $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n$ be $n+1$ points ordered in the plane. The oriented polygon formed by them is called the *characteristic polygon* or *Bézier polygon*. Let us introduce the Bernstein polynomials over the interval $[0,1]$ defined as

$$b_{n,k}(t) = \left(\begin{array}{c} n \\ k \end{array} \right) t^k (1-t)^{n-k} = \frac{n!}{k!(n-k)!} t^k (1-t)^{n-k},$$

for $n = 0, 1, \dots$ and $k = 0, \dots, n$. They can be obtained by the following recursive formula

$$\begin{cases} b_{n,0}(t) = (1-t)^n \\ b_{n,k}(t) = (1-t)b_{n-1,k}(t) + tb_{n-1,k-1}(t), \quad k = 1, \dots, n, \ t \in [0,1]. \end{cases}$$

It is easily seen that $b_{n,k} \in \mathbb{P}_n$, for $k = 0, \dots, n$. Also, $\{b_{n,k},\ k = 0, \dots, n\}$ provides a basis for $\mathbb{P}_n$. The Bézier curve is defined as follows

$$B_n(\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n, t) = \sum_{k=0}^{n} \mathbf{P}_k b_{n,k}(t), \qquad 0 \le t \le 1. \tag{8.58}$$

This expression can be regarded as a weighted average of the points $\mathbf{P}_k$, with weights $b_{n,k}(t)$.

The Bézier curves can also be obtained by a pure geometric approach starting from the characteristic polygon. Indeed, for any fixed $t \in [0,1]$, we define $\mathbf{P}_{i,1}(t) = (1-t)\mathbf{P}_i + t\mathbf{P}_{i+1}$ for $i = 0, \dots, n-1$ and, for t fixed, the piecewise line that joins the new nodes $\mathbf{P}_{i,1}(t)$ forms a polygon of $n-1$ edges. We can now repeat the procedure by generating the new vertices $\mathbf{P}_{i,2}(t)$ $(i = 0, \dots, n-2)$, and terminating as soon as the polygon comprises only the vertices $\mathbf{P}_{0,n-1}(t)$ and $\mathbf{P}_{1,n-1}(t)$. It can be shown that

$$\mathbf{P}_{0,n}(t) = (1-t)\mathbf{P}_{0,n-1}(t) + t\mathbf{P}_{1,n-1}(t) = B_n(\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n, t),$$

that is, $\mathbf{P}_{0,n}(t)$ is equal to the value of the Bézier curve B_n at the points corresponding to the fixed value of t. Repeating the process for several values of the parameter t yields the construction of the curve in the considered region of the plane.

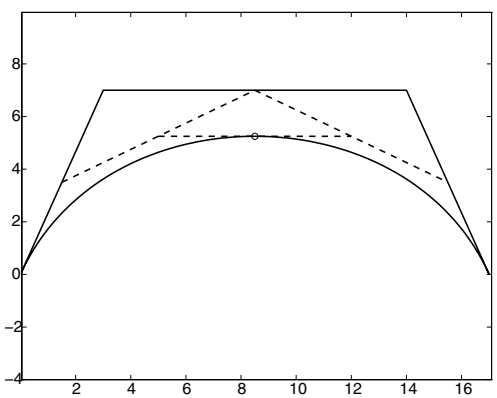

FIGURE 8.14. Computation of the value of B_3 relative to the points (0,0), (4,7), (14,7), (17,0) for $t = 0.5$, using the graphical method described in the text

Notice that, for a given node configuration, several curves can be constructed according to the ordering of points $\mathbf{P}_i$. Moreover, the Bézier curve

$B_n(\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n, t)$ coincides with $B_n(\mathbf{P}_n, \mathbf{P}_{n-1}, \ldots, \mathbf{P}_0, t)$, apart from the orientation.

Program 69 computes $b_{n,k}$ at the point x for $x \in [0, 1]$.

Program 69 - bernstein : Bernstein polynomials

```
function [bnk]=bernstein (n,k,x)
if k == 0,
   C = 1;
else,
   C = prod ([1:n])/( prod([1:k])*prod([1:n-k]));
end
bnk = C * x^k * (1-x)^(n-k);
```

Program 70 plots the Bézier curve relative to the set of points $(\mathtt{x}, \mathtt{y})$.

Program 70 - bezier : Bézier curves

```
function [bezx,bezy] = bezier (x, y, n)
i = 0; k = 0;
for t = 0:0.01:1,
    i = i + 1; bnk = bernstein (n,k,t); ber(i) = bnk;
end
bezx = ber * x (1); bezy = ber * y (1);
for k = 1:n
    i = 0;
    for t = 0:0.01:1
        i = i + 1; bnk = bernstein (n,k,t); ber(i) = bnk;
    end
  bezx = bezx + ber * x (k+1); bezy = bezy + ber * y (k+1);
end
plot(bezx,bezy)
```

In practice, the Bézier curves are rarely used since they do not provide a sufficiently accurate approximation to the characteristic polygon. For this reason, in the 70's the *parametric B-splines* were introduced, and they are used in (8.58) instead of the Bernstein polynomials. Parametric B-splines are widely employed in packages for computer graphics since they enjoy the following properties:

1. perturbing a single vertex of the characteristic polygon yields a local perturbation of the curve only around the vertex itself;

2. the parametric B-spline better approximates the control polygon than the corresponding Bézier curve does, and it is always contained within the convex hull of the polygon.

In Figure 8.15 a comparison is made between Bézier curves and parametric B-splines for the approximation of a given characteristic polygon.

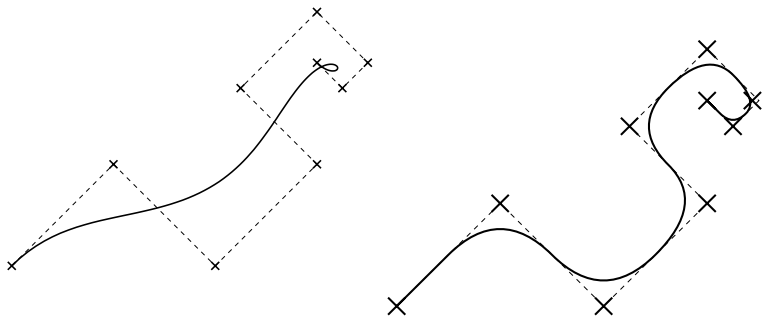

FIGURE 8.15. Comparison of a Bézier curve (left) and a parametric B-spline (right). The vertices of the characteristic polygon are denoted by $\times$

We conclude this section by noticing that parametric cubic B-splines allow for obtaining locally straight lines by aligning four consecutive vertices (see Figure 8.16) and that a parametric B-spline can be constrained at a specific point of the characteristic polygon by simply making three consecutive points of the polygon coincide with the desired point.

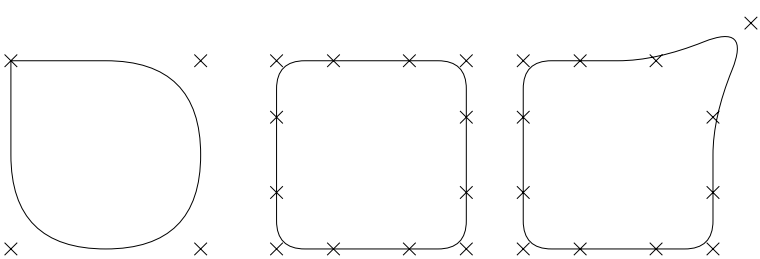

FIGURE 8.16. Some parametric B-splines as functions of the number and positions of the vertices of the characteristic polygon. Notice in the last figure (right) the localization effects due to moving a single vertex

8.8 Applications

In this section we consider two problems arising from the solution of fourth-order differential equations and from the reconstruction of images in axial tomographies.

8.8.1 Finite Element Analysis of a Clamped Beam

Let us employ piecewise Hermite polynomials (see Section 8.4) for the numerical approximation of the transversal bending of a clamped beam. This problem was already considered in Section 4.7.2 where centered finite differences were used.

The mathematical model is the fourth-order boundary value problem (4.74), here presented in the following general formulation

$$\begin{cases} (\alpha(x)u''(x))'' = f(x), & 0 < x < \mathcal{L} \\ u(0) = u(\mathcal{L}) = 0, & u'(0) = u'(\mathcal{L}) = 0. \end{cases} \tag{8.59}$$

In the particular case of (4.74) we have $\alpha = EJ$ and $f = P$; we assume henceforth that α is a positive and bounded function over $(0, \mathcal{L})$ and that $f \in \mathrm{L}^2(0, \mathcal{L})$.

We multiply (8.59) by a sufficiently smooth arbitrary function v, then, we integrate by parts twice, to obtain

$$\int_0^{\mathcal{L}} \alpha u'' v'' dx - [\alpha u''' v]_0^{\mathcal{L}} + [\alpha u'' v']_0^{\mathcal{L}} = \int_0^{\mathcal{L}} f v dx.$$

Problem (8.59) is then replaced by the following problem in integral form

$$\text{find } u \in V \text{ such that } \quad \int_0^{\mathcal{L}} \alpha u'' v'' dx = \int_0^{\mathcal{L}} f v dx, \qquad \forall v \in V, \tag{8.60}$$

where

$$V = \left\{ v : \, v^{(k)} \in \mathrm{L}^2(0, \mathcal{L}),\, k = 0, 1, 2,\, v^{(k)}(0) = v^{(k)}(\mathcal{L}) = 0,\, k = 0, 1 \right\}.$$

Problem (8.60) admits a unique solution, which represents the deformed configuration that minimizes the total potential energy of the beam over the space V (see, for instance, [Red86], p. 156)

$$J(u) = \int_0^{\mathcal{L}} \left(\frac{1}{2} \alpha (u'')^2 - f u \right) dx.$$

In view of the numerical solution of problem (8.60), we introduce a partition $\mathcal{T}_h$ of $[0, \mathcal{L}]$ into K subintervals $T_k = [x_{k-1}, x_k]$, $(k = 1, \ldots, K)$ of uniform length $h = \mathcal{L}/K$, with $x_k = kh$, and the finite dimensional space

$$\begin{aligned} V_h = \; & \left\{ v_h \in C^1([0, \mathcal{L}]),\, v_h|_T \in \mathbb{P}_3(T) \right. \\ & \left. \forall T \in \mathcal{T}_h, v_h^{(k)}(0) = v_h^{(k)}(\mathcal{L}) = 0,\, k = 0, 1 \right\}. \end{aligned} \tag{8.61}$$

Let us equip V_h with a basis. For this purpose, we associate with each internal node x_i $(i = 1, \dots, K-1)$ a support $\sigma_i = T_i \cup T_{i+1}$ and *two* functions φ_i, ψ_i defined as follows: for any k, $\varphi_i|_{T_k} \in \mathbb{P}_3(T_k)$, $\psi_i|_{T_k} \in \mathbb{P}_3(T_k)$ and for any $j = 0, \dots, K$,

$$\begin{cases} \varphi_i(x_j) = \delta_{ij}, & \varphi_i'(x_j) = 0, \\ \psi_i(x_j) = 0, & \psi_i'(x_j) = \delta_{ij}. \end{cases} \tag{8.62}$$

Notice that the above functions belong to V_h and define a basis

$$B_h = \{\varphi_i,\, \psi_i,\, i = 1, \dots, K-1\}. \tag{8.63}$$

These basis functions can be brought back to the reference interval $\hat{T} = [0, 1]$ for $0 \le \hat{x} \le 1$, by the affine maps $x = h\hat{x} + x_{k-1}$ between $\hat{T}$ and T_k, for $k = 1, \dots, K$.

Therefore, let us introduce on the interval $\hat{T}$ the basis functions $\hat{\varphi}_0^{(0)}$ and $\hat{\varphi}_0^{(1)}$, associated with the node $\hat{x} = 0$, and $\hat{\varphi}_1^{(0)}$ and $\hat{\varphi}_1^{(1)}$, associated with node $\hat{x} = 1$. Each of these is of the form $\hat{\varphi} = a_0 + a_1\hat{x} + a_2\hat{x}^2 + a_3\hat{x}^3$; in particular, the functions with superscript "0" must satisfy the first two conditions in (8.62), while those with superscript "1" must fulfill the remaining two conditions. Solving the (4×4) associated system, we get

$$\begin{aligned} &\hat{\varphi}_0^{(0)}(\hat{x}) = 1 - 3\hat{x}^2 + 2\hat{x}^3, \quad &&\hat{\varphi}_0^{(1)}(\hat{x}) = \hat{x} - 2\hat{x}^2 + \hat{x}^3, \\ &\hat{\varphi}_1^{(0)}(\hat{x}) = 3\hat{x}^2 - 2\hat{x}^3, &&\hat{\varphi}_1^{(1)}(\hat{x}) = -\hat{x}^2 + \hat{x}^3. \end{aligned} \tag{8.64}$$

The graphs of the functions (8.64) are drawn in Figure 8.17 (left), where (0), (1), (2) and (3) denote $\hat{\varphi}_0^{(0)}$, $\hat{\varphi}_1^{(0)}$, $\hat{\varphi}_0^{(1)}$ and $\hat{\varphi}_1^{(1)}$, respectively.

The function $u_h \in V_h$ can be written as

$$u_h(x) = \sum_{i=1}^{K-1} u_i \varphi_i(x) + \sum_{i=1}^{K-1} u_i^{(1)} \psi_i(x). \tag{8.65}$$

The coefficients and the *degrees of freedom* of u_h have the following meaning: $u_i = u_h(x_i)$, $u_i^{(1)}(x_i) = u_h'(x_i)$ for $i = 1, \dots, K-1$. Notice that (8.65) is a special instance of (8.32), having set $m_i = 1$.

The discretization of problem (8.60) reads

$$\text{find } u_h \in V_h \text{ such that } \quad \int_0^{\mathcal{L}} \alpha u_h'' v_h'' dx = \int_0^{\mathcal{L}} f v_h dx, \qquad \forall v_h \in B_h. \tag{8.66}$$

This is called the *Galerkin finite element* approximation of the differential problem (8.59). We refer to Chapter 12, Sections 12.4 and 12.4.5, for a more comprehensive discussion and analysis of the method.

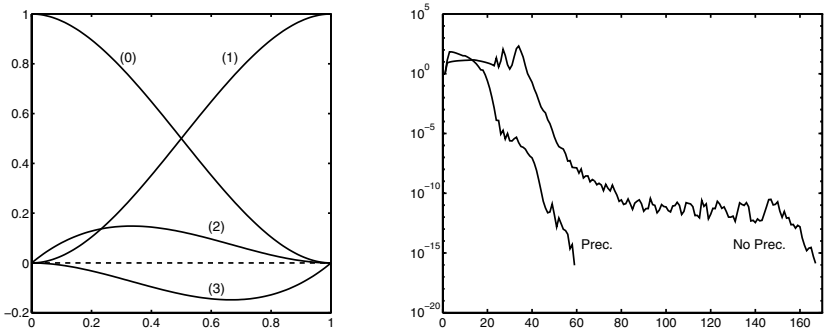

FIGURE 8.17. Canonical Hermite basis on the reference interval $0 \leq \hat{x} \leq 1$ (left); convergence histories for the conjugate gradient method in the solution of system (8.69) (right). On the x-axis the number of iterations k is shown, while the y-axis represents the quantity $\|\mathbf{r}^{(k)}\|_2/\|\mathbf{b}_1\|_2$, where $\mathbf{r}$ is the residual of system (8.69)

Using the representation (8.65) we end up with the following system in the $2K-2$ unknowns $u_1, u_2, \ldots, u_{K-1}, u_1^{(1)}, u_2^{(1)}, \ldots u_{K-1}^{(1)}$

$$\left\{ \begin{array}{l} \displaystyle\sum_{j=1}^{K-1} \left\{ u_j \int_0^{\mathcal{L}} \alpha \varphi_j'' \varphi_i'' dx + u_j^{(1)} \int_0^{\mathcal{L}} \alpha \psi_j'' \varphi_i'' dx \right\} = \int_0^{\mathcal{L}} f \varphi_i dx, \\ \displaystyle\sum_{j=1}^{K-1} \left\{ u_j \int_0^{\mathcal{L}} \alpha \varphi_j'' \psi_i'' dx + u_j^{(1)} \int_0^{\mathcal{L}} \alpha \psi_j'' \psi_i'' dx \right\} = \int_0^{\mathcal{L}} f \psi_i dx, \end{array} \right. \tag{8.67}$$

for $i = 1, \ldots, K-1$. Assuming, for the sake of simplicity, that the beam has unit length $\mathcal{L}$, that α and f are two constants and computing the integrals in (8.67), the final system reads in matrix form

$$\left\{ \begin{array}{ll} \mathrm{A}\mathbf{u} + \mathrm{B}\mathbf{p} = & \mathbf{b}_1 \\ \mathrm{B}^T\mathbf{u} + \mathrm{C}\mathbf{p} = & \mathbf{0}, \end{array} \right. \tag{8.68}$$

where the vectors $\mathbf{u}, \mathbf{p} \in \mathbb{R}^{K-1}$ contain the nodal unknowns u_i and $u_i^{(1)}$, $\mathbf{b}_1 \in \mathbb{R}^{K-1}$ is the vector of components equal to $h^4 f/\alpha$, while

$$\begin{aligned} \mathrm{A} &= \mathrm{tridiag}_{K-1}(-12, 24, -12), \\ \mathrm{B} &= \mathrm{tridiag}_{K-1}(-6, 0, 6), \\ \mathrm{C} &= \mathrm{tridiag}_{K-1}(2, 8, 2). \end{aligned}$$

System (8.68) has size equal to $2(K-1)$; eliminating the unknown $\mathbf{p}$ from the second equation, we get the reduced system (of size $K-1$)

$$\left(\mathrm{A} - \mathrm{B}\mathrm{C}^{-1}\mathrm{B}^T\right)\mathbf{u} = \mathbf{b}_1. \tag{8.69}$$

Since B is skew-symmetric and A is symmetric and positive definite (s.p.d.), the matrix $\mathrm{M} = \mathrm{A} - \mathrm{B}\mathrm{C}^{-1}\mathrm{B}^T$ is s.p.d. too. Using Cholesky factorization for solving system (8.69) is impractical as C^{-1} is full. An alternative is thus the conjugate gradient method (CG) supplied with a suitable preconditioner as the spectral condition number of M is of the order of $h^{-4} = K^4$.

We notice that computing the residual at each step $k \geq 0$ requires solving a linear system whose right side is the vector $\mathrm{B}^T\mathbf{u}^{(k)}$, $\mathbf{u}^{(k)}$ being the current iterate of CG method, and whose coefficient matrix is matrix C. This system can be solved using the Thomas algorithm (3.53) with a cost of the order of K flops.

The CG algorithm terminates in correspondence to the lowest value of k for which $\|\mathbf{r}^{(k)}\|_2 \leq \mathtt{u}\|\mathbf{b}_1\|_2$, where $\mathbf{r}^{(k)}$ is the residual of system (8.69) and $\mathtt{u}$ is the *roundoff* unit.

The results obtained running the CG method in the case of a uniform partition of $[0, 1]$ with $K = 50$ elements and setting $\alpha = f = 1$ are summarized in Figure 8.17 (right), which shows the convergence histories of the method in both nonpreconditioned form (denoted by "Non Prec.") and with SSOR preconditioner (denoted by "Prec."), having set the relaxation parameter $\omega = 1.95$.

We notice that the CG method does not converge within $K - 1$ steps due to the effect of the rounding errors. Notice also the effectiveness of the SSOR preconditioner in terms of the reduction of the number of iterations. However, the high computational cost of this preconditioner prompts us to devise another choice. Looking at the structure of the matrix M a natural preconditioner is $\mathcal{M} = \mathrm{A} - \mathrm{B}\widetilde{\mathrm{C}}^{-1}\mathrm{B}^T$, where $\widetilde{\mathrm{C}}$ is the diagonal matrix whose entries are $\widetilde{c}_{ii} = \sum_{j=1}^{K-1} |c_{ij}|$. The matrix $\mathcal{M}$ is banded so that its inversion requires a strongly reduced cost than for the SSOR preconditioner. Moreover, as shown in Table 8.6, using $\mathcal{M}$ provides a dramatic decrease of the number of iterations to converge.

K	Without Precond.	SSOR	$\mathcal{M}$
25	51	27	12
50	178	61	25
100	685	118	33
200	2849	237	34

TABLE 8.6. Number of iterations as a function of K

8.8.2 Geometric Reconstruction Based on Computer Tomographies

A typical application of the algorithms presented in Section 8.7 deals with the reconstruction of the three-dimensional structure of internal organs of human body based on *computer tomographies* (CT).

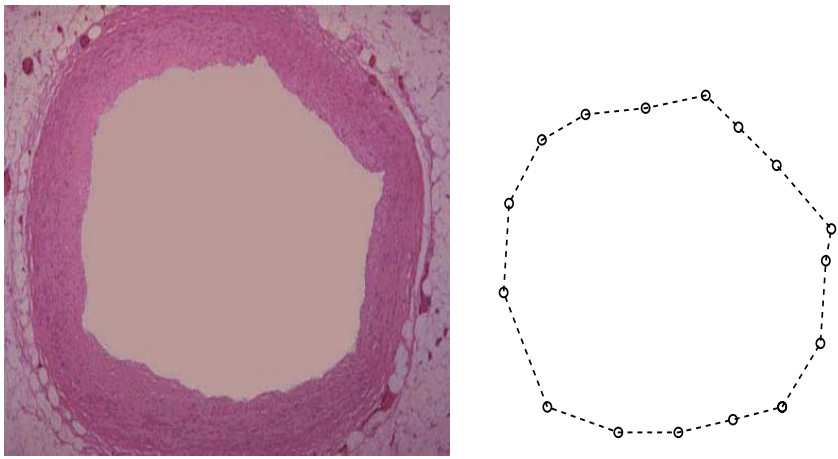

FIGURE 8.18. Cross-section of a blood vessel (left) and an associated characteristic polygon using 16 points $\mathbf{P}_i$ (right)

The CT usually provides a sequence of images which represent the sections of an organ at several horizontal planes; as a convention, we say that the CT produces sections of the x, y plane in correspondance of several values of z. The result is analogous to what we would get by sectioning the organ at different values of z and taking the picture of the corresponding sections. Obviously, the great advantage in using the CT is that the organ under investigation can be visualized without being hidden by the neighboring ones, as happens in other kinds of medical images, e.g., angiographies.

The image that is obtained for each section is coded into a matrix of *pixels* (abbreviation of *pictures elements*) in the x, y plane; a certain value is associated with each pixel expressing the level of grey of the image at that point. This level is determined by the density of X rays which are collected by a detector after passing through the human body. In practice, the information contained in a CT at a given value of z is expressed by a set of points (x_i, y_i) which identify the boundary of the organ at z.

To improve the diagnostics it is often useful to reconstruct the three-dimensional structure of the organ under examination starting from the sections provided by the CT. With this aim, it is necessary to convert the information coded by pixels into a parametric representation which can be expressed by suitable functions interpolating the image at some significant points on its boundary. This reconstruction can be carried out by using the methods described in Section 8.7 as shown in Figure 8.19.
A set of curves like those shown in Figure 8.19 can be suitably stacked to provide an overall three-dimensional view of the organ under examination.

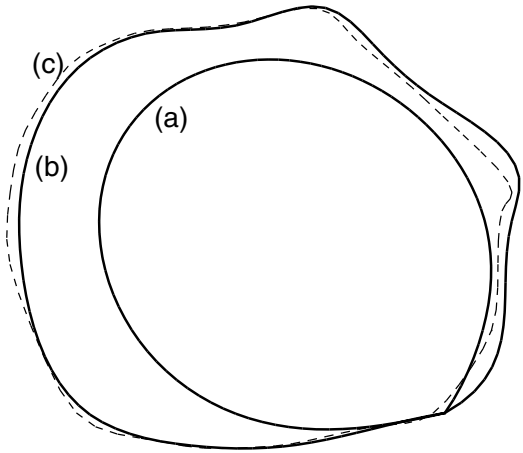

FIGURE 8.19. Reconstruction of the internal vessel of Figure 8.18 using different interpolating splines with the same characteristic polygon: (a) Bézier curves, (b) parametric splines and (c) parametric B-splines

8.9 Exercises

1. Prove that the characteristic polynomials $l_i \in \mathbb{P}_n$ defined in (8.3) form a basis for $\mathbb{P}_n$.

2. An alternative approach to the method in Theorem 8.1, for constructing the interpolating polynomial, consists of directly enforcing the $n+1$ interpolation constraints on Π_n and then computing the coefficients a_i. By doing so, we end up with a linear system $\mathrm{X}\mathbf{a}=\mathbf{y}$, with $\mathbf{a}=(a_0,\ldots,a_n)^T$, $\mathbf{y}=(y_0,\ldots,y_n)^T$ and $\mathrm{X}=[x_i^j]$. X is called *Vandermonde matrix*. Prove that X is nonsingular if the nodes x_i are distinct.

   [*Hint*: show that $\det(\mathrm{X})=\prod_{0\le j<i\le n}(x_i-x_j)$ by recursion on n.]

3. Prove that $\omega'_{n+1}(x_i)=\prod_{\substack{j=0\\ j\neq i}}^{n}(x_i-x_j)$ where ω_{n+1} is the nodal polynomial (8.6). Then, check (8.5).

4. Provide an estimate of $\|\omega_{n+1}\|_\infty$, in the cases $n=1$ and $n=2$, for a distribution of equally spaced nodes.

5. Prove that

   $$(n-1)!h^{n-1}|(x-x_{n-1})(x-x_n)| \le |\omega_{n+1}(x)| \le n!h^{n-1}|(x-x_{n-1})(x-x_n)|,$$

   where n is even, $-1=x_0<x_1<\ldots<x_{n-1}<x_n=1$, $x\in(x_{n-1},x_n)$ and $h=2/n$.

[*Hint* : let $N = n/2$ and show first that

$$\begin{aligned}\omega_{n+1}(x) = (x + Nh)(x + (N-1)h)\dots(x+h)x \\ (x-h)\dots(x-(N-1)h)(x-Nh).\end{aligned} \tag{8.70}$$

Then, take $x = rh$ with $N - 1 < r < N$.]

6. Under the assumptions of Exercise 5, show that $|\omega_{n+1}|$ is maximum if $x \in (x_{n-1}, x_n)$ (notice that $|\omega_{n+1}|$ is an even function).
[*Hint* : use (8.70) to prove that $|\omega_{n+1}(x+h)/\omega_{n+1}(x)| > 1$ for any $x \in (0, x_{n-1})$ with x not coinciding with any interpolation node.]
7. Prove the recursive relation (8.19) for Newton divided differences.
8. Determine an interpolating polynomial $Hf \in \mathbb{P}_n$ such that

$$(Hf)^{(k)}(x_0) = f^{(k)}(x_0), \qquad k = 0, \dots, n,$$

and check that

$$Hf(x) = \sum_{j=0}^{n} \frac{f^{(j)}(x_0)}{j!}(x - x_0)^j,$$

that is, the Hermite interpolating polynomial on one node coincides with the *Taylor polynomial*.
9. Given the following set of data

$$\left\{ f_0 = f(-1) = 1,\ f_1 = f'(-1) = 1,\ f_2 = f'(1) = 2,\ f_3 = f(2) = 1 \right\},$$

prove that the Hermite-Birkoff interpolating polynomial H_3 does not exist for them.
[*Solution* : letting $H_3(x) = a_3x^3 + a_2x^2 + a_1x + a_0$, one must check that the matrix of the linear system $H_3(x_i) = f_i$ for $i = 0, \dots, 3$ is singular.]
10. Check that any $s_k \in \mathcal{S}_k[a, b]$ admits a representation of the form

$$s_k(x) = \sum_{i=0}^{k} b_i x^i + \sum_{i=1}^{g} c_i (x - x_i)_+^k,$$

that is, $1, x, x^2, \dots, x^k, (x - x_1)_+^k, \dots, (x - x_g)_+^k$ form a basis for $\mathcal{S}_k[a, b]$.
11. Prove Property 8.2 and check its validity even in the case where the spline s satisfies conditions of the form $s'(a) = f'(a)$, $s'(b) = f'(b)$.
[*Hint*: start from

$$\int_a^b \left[f''(x) - s''(x)\right] s''(x)dx = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} \left[f''(x) - s''(x)\right] s''dx$$

and integrate by parts twice.]

12. Let $f(x) = \cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$; then, consider the following rational approximation

$$r(x) = \frac{a_0 + a_2 x^2 + a_4 x^4}{1 + b_2 x^2}, \tag{8.71}$$

called the *Padé approximation*. Determine the coefficients of r in such a way that

$$f(x) - r(x) = \gamma_8 x^8 + \gamma_{10} x^{10} + \ldots$$

[*Solution*: $a_0 = 1$, $a_2 = -7/15$, $a_4 = 1/40$, $b_2 = 1/30$.]

13. Assume that the function f of the previous exercise is known at a set of n equally spaced points $x_i \in (-\pi/2, \pi/2)$ with $i = 0, \ldots, n$. Repeat Exercise 12, determining, by using MATLAB, the coefficients of r in such a way that the quantity $\sum_{i=0}^{n} |f(x_i) - r(x_i)|^2$ is minimized. Consider the cases $n = 5$ and $n = 10$.

9
Numerical Integration

In this chapter we present the most commonly used methods for numerical integration. We will mainly consider one-dimensional integrals over bounded intervals, although in Sections 9.8 and 9.9 an extension of the techniques to integration over unbounded intervals (or integration of functions with singularities) and to the multidimensional case will be considered.

9.1 Quadrature Formulae

Let f be a real integrable function over the interval $[a, b]$. Computing explicitly the definite integral $I(f) = \int_a^b f(x)dx$ may be difficult or even impossible. Any explicit formula that is suitable for providing an approximation of $I(f)$ is said to be a *quadrature formula* or *numerical integration formula*.

An example can be obtained by replacing f with an approximation f_n, depending on the integer $n \geq 0$, then computing $I(f_n)$ instead of $I(f)$. Letting $I_n(f) = I(f_n)$, we have

$$I_n(f) = \int_a^b f_n(x)dx, \qquad n \geq 0. \tag{9.1}$$

The dependence on the end points a, b is always understood, so we write $I_n(f)$ instead of $I_n(f; a, b)$.

If $f \in C^0([a,b])$, the *quadrature error* $E_n(f) = I(f) - I_n(f)$ satisfies

$$|E_n(f)| \leq \int_a^b |f(x) - f_n(x)|dx \leq (b-a)\|f - f_n\|_\infty.$$

Therefore, if for some n, $\|f - f_n\|_\infty < \varepsilon$, then $|E_n(f)| \leq \varepsilon(b-a)$.

The approximant f_n must be easily integrable, which is the case if, for example, $f_n \in \mathbb{P}_n$. In this respect, a natural approach consists of using $f_n = \Pi_n f$, the interpolating Lagrange polynomial of f over a set of $n+1$ distinct nodes $\{x_i\}$, with $i = 0, \ldots, n$. By doing so, from (9.1) it follows that

$$I_n(f) = \sum_{i=0}^{n} f(x_i) \int_a^b l_i(x)dx, \tag{9.2}$$

where l_i is the characteristic Lagrange polynomial of degree n associated with node x_i (see Section 8.1). We notice that (9.2) is a special instance of the following quadrature formula

$$I_n(f) = \sum_{i=0}^{n} \alpha_i f(x_i), \tag{9.3}$$

where the coefficients α_i of the linear combination are given by $\int_a^b l_i(x)dx$. Formula (9.3) is a weighted sum of the values of f at the points x_i, for $i = 0, \ldots, n$. These points are said to be the *nodes* of the quadrature formula, while the numbers $\alpha_i \in \mathbb{R}$ are its *coefficients* or *weights*. Both weights and nodes depend in general on n; again, for notational simplicity, this dependence is always understood.

Formula (9.2), called the *Lagrange* quadrature formula, can be generalized to the case where also the values of the derivative of f are available. This leads to the *Hermite* quadrature formula (see Section 9.5)

$$I_n(f) = \sum_{k=0}^{1} \sum_{i=0}^{n} \alpha_{ik} f^{(k)}(x_i) \tag{9.4}$$

where the weights are now denoted by α_{ik}.

Both (9.2) and (9.4) are *interpolatory quadrature formulae*, since the function f has been replaced by its interpolating polynomial (Lagrange and Hermite polynomials, respectively). We define the *degree of exactness* of a quadrature formula as the maximum integer $r \geq 0$ for which

$$I_n(f) = I(f), \qquad \forall f \in \mathbb{P}_r.$$

Any interpolatory quadrature formula that makes use of $n+1$ distinct nodes has degree of exactness equal to at least n. Indeed, if $f \in \mathbb{P}_n$, then

$\Pi_n f = f$ and thus $I_n(\Pi_n f) = I(\Pi_n f)$. The converse statement is also true, that is, a quadrature formula using $n+1$ distinct nodes and having degree of exactness equal at least to n is necessarily of interpolatory type (for the proof see [IK66], p. 316).

As we will see in Section 10.2, the degree of exactness of a Lagrange quadrature formula can be as large as $2n+1$ in the case of the so-called Gaussian quadrature formulae.

9.2 Interpolatory Quadratures

We consider three remarkable instances of formula (9.2), corresponding to $n = 0$, 1 and 2.

9.2.1 The Midpoint or Rectangle Formula

This formula is obtained by replacing f over $[a, b]$ with the constant function equal to the value attained by f at the midpoint of $[a, b]$ (see Figure 9.1, left). This yields

$$I_0(f) = (b-a) f\left(\frac{a+b}{2}\right) \tag{9.5}$$

with weight $\alpha_0 = b - a$ and node $x_0 = (a+b)/2$. If $f \in C^2([a, b])$, the quadrature error is

$$E_0(f) = \frac{h^3}{3} f''(\xi), \quad h = \frac{b-a}{2}, \tag{9.6}$$

where ξ lies within the interval (a, b).

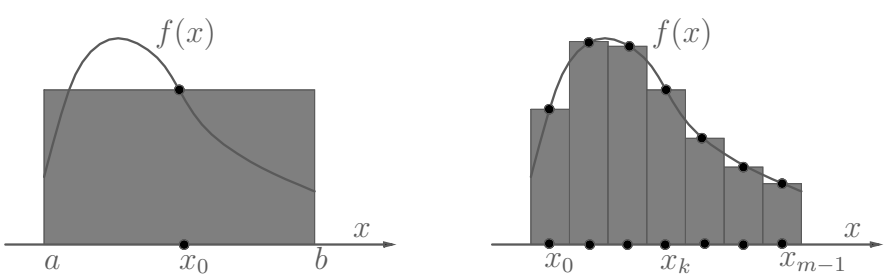

FIGURE 9.1. The midpoint formula (left); the composite midpoint formula (right)

Indeed, expanding f in a Taylor's series around $c = (a+b)/2$ and truncating at the second-order, we get

$$f(x) = f(c) + f'(c)(x-c) + f''(\eta(x))(x-c)^2/2,$$

from which, integrating on (a,b) and using the mean-value theorem, (9.6) follows. From this, it turns out that (9.5) is exact for constant and affine functions (since in both cases $f''(\xi) = 0$ for any $\xi \in (a,b)$), so that the midpoint rule has *degree of exactness* equal to 1.
It is worth noting that if the width of the integration interval $[a,b]$ is not sufficiently small, the quadrature error (9.6) can be quite large. This drawback is common to all the numerical integration formulae that will be described in the three forthcoming sections and can be overcome by resorting to their composite counterparts as discussed in Section 9.4.

Suppose now that we approximate the integral $I(f)$ by replacing f over $[a,b]$ with its composite interpolating polynomial of degree zero, constructed on m subintervals of width $H = (b-a)/m$, for $m \geq 1$ (see Figure 9.1, right). Introducing the quadrature nodes $x_k = a + (2k+1)H/2$, for $k = 0, \ldots, m-1$, we get the composite midpoint formula

$$I_{0,m}(f) = H \sum_{k=0}^{m-1} f(x_k), \qquad m \geq 1. \tag{9.7}$$

The quadrature error $E_{0,m}(f) = I(f) - I_{0,m}(f)$ is given by

$$E_{0,m}(f) = \frac{b-a}{24} H^2 f''(\xi), \quad H = \frac{b-a}{m} \tag{9.8}$$

provided that $f \in C^2([a,b])$ and where $\xi \in (a,b)$. From (9.8) we conclude that (9.7) has degree of exactness equal to 1; (9.8) can be proved by recalling (9.6) and using the additivity of integrals. Indeed, for $k = 0, \ldots, m-1$ and $\xi_k \in (a+kH, a+(k+1)H)$,

$$E_{0,m}(f) = \sum_{k=0}^{m-1} f''(\xi_k)(H/2)^3/3 = \sum_{k=0}^{m-1} f''(\xi_k) \frac{H^2}{24} \frac{b-a}{m} = \frac{b-a}{24} H^2 f''(\xi).$$

The last equality is a consequence of the following theorem, that is applied letting $u = f''$ and $\delta_j = 1$ for $j = 0, \ldots, m-1$.

Theorem 9.1 (discrete mean-value theorem) *Let $u \in C^0([a,b])$ and let x_j be $s+1$ points in $[a,b]$ and δ_j be $s+1$ constants, all having the same sign. Then there exists $\eta \in [a,b]$ such that*

$$\sum_{j=0}^{s} \delta_j u(x_j) = u(\eta) \sum_{j=0}^{s} \delta_j. \tag{9.9}$$

Proof. Let $u_m = \min_{x\in[a,b]} u(x) = u(\bar{x})$ and $u_M = \max_{x\in[a,b]} u(x) = u(\bar{\bar{x}})$, where $\bar{x}$ and $\bar{\bar{x}}$ are two points in (a,b). Then

$$u_m \sum_{j=0}^{s} \delta_j \leq \sum_{j=0}^{s} \delta_j u(x_j) \leq u_M \sum_{j=0}^{s} \delta_j. \tag{9.10}$$

Let $\sigma_s = \sum_{j=0}^{s} \delta_j u(x_j)$ and consider the continuous function $U(x) = u(x) \sum_{j=0}^{s} \delta_j$. Thanks to (9.10), $U(\bar{x}) \leq \sigma_s \leq U(\bar{\bar{x}})$. Applying the mean-value theorem, there exists a point η between a and b such that $U(\eta) = \sigma_s$, which is (9.9). A similar proof can be carried out if the coefficients δ_j are negative. ◇

The composite midpoint formula is implemented in Program 71. Throughout this chapter, we shall denote by `a` and `b` the end points of the integration interval and by `m` the number of quadrature subintervals. The variable `fun` contains the expression of the function f, while the output variable `int` contains the value of the approximate integral.

Program 71 - midpntc : Midpoint composite formula

```
function int = midpntc(a,b,m,fun)
h=(b-a)/m; x=[a+h/2:h:b]; dim = max(size(x)); y=eval(fun);
if size(y)==1, y=diag(ones(dim))*y; end; int=h*sum(y);
```

9.2.2 The Trapezoidal Formula

This formula is obtained by replacing f with $\Pi_1 f$, its Lagrange interpolating polynomial of degree 1, relative to the nodes $x_0 = a$ and $x_1 = b$ (see Figure 9.2, left). The resulting quadrature, having nodes $x_0 = a$, $x_1 = b$ and weights $\alpha_0 = \alpha_1 = (b-a)/2$, is

$$I_1(f) = \frac{b-a}{2} \left[f(a) + f(b) \right]. \tag{9.11}$$

If $f \in C^2([a,b])$, the quadrature error is given by

$$E_1(f) = -\frac{h^3}{12} f''(\xi), \quad h = b - a \tag{9.12}$$

where ξ is a point within the integration interval.

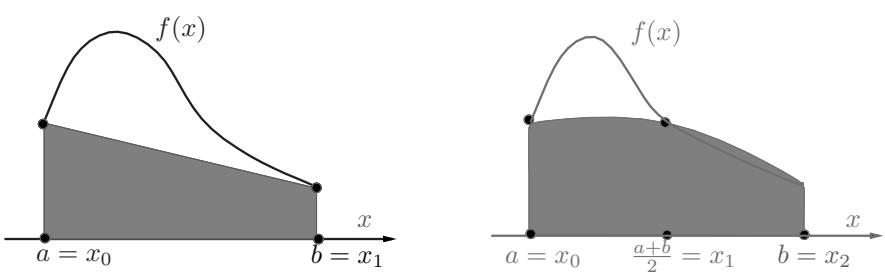

FIGURE 9.2. Trapezoidal formula (left) and Cavalieri-Simpson formula (right)

Indeed, from the expression of the interpolation error (8.7) one gets

$$E_1(f) = \int_a^b (f(x) - \Pi_1 f(x))dx = -\frac{1}{2}\int_a^b f''(\xi(x))(x-a)(b-x)dx.$$

Since $\omega_2(x) = (x-a)(x-b) < 0$ in (a,b), the mean-value theorem yields

$$E_1(f) = (1/2)f''(\xi)\int_a^b \omega_2(x)dx = -f''(\xi)(b-a)^3/12,$$

for some $\xi \in (a,b)$, which is (9.12). The trapezoidal quadrature therefore has degree of exactness equal to 1, as is the case with the midpoint rule.

To obtain the composite trapezoidal formula, we proceed as in the case where $n = 0$, by replacing f over $[a,b]$ with its composite Lagrange polynomial of degree 1 on m subintervals, with $m \geq 1$. Introduce the quadrature nodes $x_k = a + kH$, for $k = 0, \ldots, m$ and $H = (b-a)/m$, getting

$$I_{1,m}(f) = \frac{H}{2}\sum_{k=0}^{m-1}\left(f(x_k) + f(x_{k+1})\right), \qquad m \geq 1. \tag{9.13}$$

Each term in (9.13) is counted twice, except the first and the last one, so that the formula can be written as

$$I_{1,m}(f) = H\left[\frac{1}{2}f(x_0) + f(x_1) + \ldots + f(x_{m-1}) + \frac{1}{2}f(x_m)\right]. \tag{9.14}$$

As was done for (9.8), it can be shown that the quadrature error associated with (9.14) is

$$E_{1,m}(f) = -\frac{b-a}{12}H^2 f''(\xi),$$

provided that $f \in C^2([a,b])$, where $\xi \in (a,b)$. The degree of exactness is again equal to 1.

The composite trapezoidal rule is implemented in Program 72.

Program 72 - trapezc : Composite trapezoidal formula

```
function int = trapezc(a,b,m,fun)
h=(b-a)/m; x=[a:h:b]; dim = max(size(x)); y=eval(fun);
if size(y)==1, y=diag(ones(dim))*y; end;
int=h*(0.5*y(1)+sum(y(2:m))+0.5*y(m+1));
```

9.2.3 The Cavalieri-Simpson Formula

The Cavalieri-Simpson formula can be obtained by replacing f over $[a, b]$ with its interpolating polynomial of degree 2 at the nodes $x_0 = a$, $x_1 = (a + b)/2$ and $x_2 = b$ (see Figure 9.2, right). The weights are given by $\alpha_0 = \alpha_2 = (b - a)/6$ and $\alpha_1 = 4(b - a)/6$, and the resulting formula reads

$$I_2(f) = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]. \tag{9.15}$$

It can be shown that the quadrature error is

$$E_2(f) = -\frac{h^5}{90}f^{(4)}(\xi), \quad h = \frac{b-a}{2} \tag{9.16}$$

provided that $f \in C^4([a, b])$, and where ξ lies within (a, b). From (9.16) it turns out that (9.15) has degree of exactness equal to 3.

Replacing f with its composite polynomial of degree 2 over $[a, b]$ yields the composite formula corresponding to (9.15). Introducing the quadrature nodes $x_k = a + kH/2$, for $k = 0, \ldots, 2m$ and letting $H = (b - a)/m$, with $m \geq 1$ gives

$$I_{2,m} = \frac{H}{6}\left[f(x_0) + 2\sum_{r=1}^{m-1} f(x_{2r}) + 4\sum_{s=0}^{m-1} f(x_{2s+1}) + f(x_{2m})\right]. \tag{9.17}$$

The quadrature error associated with (9.17) is

$$E_{2,m}(f) = -\frac{b-a}{180}(H/2)^4 f^{(4)}(\xi),$$

provided that $f \in C^4([a, b])$ and where $\xi \in (a, b)$; the degree of exactness of the formula is 3.
The composite Cavalieri-Simpson quadrature is implemented in Program 73.

Program 73 - simpsonc : Composite Cavalieri-Simpson formula

```
function int = simpsonc(a,b,m,fun)
h=(b-a)/m; x=[a:h/2:b]; dim = max(size(x)); y=eval(fun);
if size(y)==1, y=diag(ones(dim))*y; end;
int=(h/6)*(y(1)+2*sum(y(3:2:2*m-1))+4*sum(y(2:2:2*m))+y(2*m+1));
```

Example 9.1 Let us employ the midpoint, trapezoidal and Cavalieri-Simpson composite formulae to compute the integral

$$\int_0^{2\pi} xe^{-x}\cos(2x)dx = \frac{\left[3(e^{-2\pi} - 1) - 10\pi e^{-2\pi}\right]}{25} \simeq -0.122122. \tag{9.18}$$

Table 9.1 shows in even columns the behavior of the absolute value of the error when halving H (thus, doubling m), while in odd columns the ratio $\mathcal{R}_m = |E_m|/|E_{2m}|$ between two consecutive errors is given. As predicted by the previous theoretical analysis, $\mathcal{R}_m$ tends to 4 for the midpoint and trapezoidal rules and to 16 for the Cavalieri-Simpson formula. •

m	$\lvert E_{0,m}\rvert$	$\mathcal{R}_m$	$\lvert E_{1,m}\rvert$	$\mathcal{R}_m$	$\lvert E_{2,m}\rvert$	$\mathcal{R}_m$
1	0.9751		1.589e-01		7.030e-01	
2	1.037	0.9406	0.5670	0.2804	0.5021	1.400
4	0.1221	8.489	0.2348	2.415	$3.139 \cdot 10^{-3}$	159.96
8	$2.980 \cdot 10^{-2}$	4.097	$5.635 \cdot 10^{-2}$	4.167	$1.085 \cdot 10^{-3}$	2.892
16	$6.748 \cdot 10^{-3}$	4.417	$1.327 \cdot 10^{-2}$	4.245	$7.381 \cdot 10^{-5}$	14.704
32	$1.639 \cdot 10^{-3}$	4.118	$3.263 \cdot 10^{-3}$	4.068	$4.682 \cdot 10^{-6}$	15.765
64	$4.066 \cdot 10^{-4}$	4.030	$8.123 \cdot 10^{-4}$	4.017	$2.936 \cdot 10^{-7}$	15.946
128	$1.014 \cdot 10^{-4}$	4.008	$2.028 \cdot 10^{-4}$	4.004	$1.836 \cdot 10^{-8}$	15.987
256	$2.535 \cdot 10^{-5}$	4.002	$5.070 \cdot 10^{-5}$	4.001	$1.148 \cdot 10^{-9}$	15.997

TABLE 9.1. Absolute error for midpoint, trapezoidal and Cavalieri-Simpson composite formulae in the approximate evaluation of integral (9.18)

9.3 Newton-Cotes Formulae

These formulae are based on Lagrange interpolation with *equally spaced* nodes in $[a, b]$. For a fixed $n \geq 0$, let us denote the quadrature nodes by $x_k = x_0 + kh$, $k = 0, \ldots, n$. The midpoint, trapezoidal and Simpson formulae are special instances of the Newton-Cotes formulae, taking $n = 0$, $n = 1$ and $n = 2$ respectively. In the general case, we define:

- *closed formulae*, those where $x_0 = a$, $x_n = b$ and $h = \dfrac{b-a}{n}$ $(n \geq 1)$;
- *open formulae*, those where $x_0 = a+h$, $x_n = b-h$ and $h = \dfrac{b-a}{n+2}$ $(n \geq 0)$.

A significant property of the Newton-Cotes formulae is that the quadrature weights α_i depend explicitly only on n and h, but not on the integration interval $[a, b]$. To check this property in the case of closed formulae, let us introduce the change of variable $x = \Psi(t) = x_0 + th$. Noting that $\Psi(0) = a$, $\Psi(n) = b$ and $x_k = a + kh$, we get

$$\frac{x - x_k}{x_i - x_k} = \frac{a + th - (a + kh)}{a + ih - (a + kh)} = \frac{t - k}{i - k}.$$

Therefore, if $n \geq 1$

$$l_i(x) = \prod_{k=0, k\neq i}^{n} \frac{t-k}{i-k} = \varphi_i(t), \qquad 0 \leq i \leq n.$$

The following expression for the quadrature weights is obtained

$$\alpha_i = \int_a^b l_i(x)dx = \int_0^n \varphi_i(t)hdt = h\int_0^n \varphi_i(t)dt,$$

from which we get the formula

$$I_n(f) = h\sum_{i=0}^{n} w_i f(x_i), \qquad w_i = \int_0^n \varphi_i(t)dt.$$

Open formulae can be interpreted in a similar manner. Actually, using again the mapping $x = \Psi(t)$, we get $x_0 = a+h$, $x_n = b-h$ and $x_k = a+h(k+1)$ for $k = 1, \ldots, n-1$. Letting, for sake of coherence, $x_{-1} = a$, $x_{n+1} = b$ and proceeding as in the case of closed formulae, we get $\alpha_i = h\int_{-1}^{n+1} \varphi_i(t)dt$, and thus

$$I_n(f) = h\sum_{i=0}^{n} w_i f(x_i), \qquad w_i = \int_{-1}^{n+1} \varphi_i(t)dt.$$

In the special case where $n = 0$, since $l_0(x) = \varphi_0(t) = 1$, we get $w_0 = 2$.

The coefficients w_i do not depend on a, b, h and f, but only depend on n, and can therefore be tabulated *a priori*. In the case of closed formulae, the polynomials φ_i and φ_{n-i}, for $i = 0, \ldots, n-1$, have by symmetry the same integral, so that also the corresponding weights w_i and w_{n-i} are equal for $i = 0, \ldots, n-1$. In the case of open formulae, the weights w_i and w_{n-i} are equal for $i = 0, \ldots, n$. For this reason, we show in Table 9.2 only the first half of the weights.

Notice the presence of *negative weights* in open formulae for $n \geq 2$. This can be a source of numerical instability, in particular due to rounding errors.

n	1	2	3	4	5	6
w_0	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{3}{8}$	$\frac{14}{45}$	$\frac{95}{288}$	$\frac{41}{140}$
w_1	0	$\frac{4}{3}$	$\frac{9}{8}$	$\frac{64}{45}$	$\frac{375}{288}$	$\frac{216}{140}$
w_2	0	0	0	$\frac{24}{45}$	$\frac{250}{288}$	$\frac{27}{140}$
w_3	0	0	0	0	0	$\frac{272}{140}$

n	0	1	2	3	4	5
w_0	2	$\frac{3}{2}$	$\frac{8}{3}$	$\frac{55}{24}$	$\frac{66}{20}$	$\frac{4277}{1440}$
w_1	0	0	$-\frac{4}{3}$	$\frac{5}{24}$	$-\frac{84}{20}$	$-\frac{3171}{1440}$
w_2	0	0	0	0	$\frac{156}{20}$	$\frac{3934}{1440}$

TABLE 9.2. Weights of closed (left) and open Newton-Cotes formulae (right)

Besides its degree of exactness, a quadrature formula can also be qualified by its *order of infinitesimal* with respect to the integration stepsize h, which is defined as the maximum integer p such that $|I(f) - I_n(f)| = \mathcal{O}(h^p)$. Regarding this, the following result holds

Theorem 9.2 *For any Newton-Cotes formula corresponding to an even value of n, the following error characterization holds*

$$E_n(f) = \frac{M_n}{(n+2)!} h^{n+3} f^{(n+2)}(\xi), \tag{9.19}$$

provided $f \in C^{n+2}([a,b])$, where $\xi \in (a,b)$ and

$$M_n = \begin{cases} \displaystyle\int_0^n t\ \pi_{n+1}(t)dt < 0 & \text{for closed formulae,} \\ \displaystyle\int_{-1}^{n+1} t\ \pi_{n+1}(t)dt > 0 & \text{for open formulae,} \end{cases}$$

having defined $\pi_{n+1}(t) = \prod_{i=0}^{n}(t-i)$. From (9.19), it turns out that the degree of exactness is equal to $n+1$ and the order of infinitesimal is $n+3$.

Similarly, for odd values of n, the following error characterization holds

$$E_n(f) = \frac{K_n}{(n+1)!} h^{n+2} f^{(n+1)}(\eta), \tag{9.20}$$

provided $f \in C^{n+1}([a,b])$, where $\eta \in (a,b)$ and

$$K_n = \begin{cases} \displaystyle\int_0^n \pi_{n+1}(t)dt < 0 & \text{for closed formulae,} \\ \displaystyle\int_{-1}^{n+1} \pi_{n+1}(t)dt > 0 & \text{for open formulae.} \end{cases}$$

The degree of exactness is thus equal to n and the order of infinitesimal is $n+2$.

Proof. We give a proof in the particular case of closed formulae with n even, referring to [IK66], pp. 308-314, for a complete demonstration of the theorem.

Thanks to (8.20), we have

$$E_n(f) = I(f) - I_n(f) = \int_a^b f[x_0, \ldots, x_n, x]\omega_{n+1}(x)dx. \tag{9.21}$$

Set $W(x) = \int_a^x \omega_{n+1}(t)dt$. Clearly, $W(a) = 0$; moreover, $\omega_{n+1}(t)$ is an odd function with respect to the midpoint $(a+b)/2$ so that $W(b) = 0$. Integrating by

parts (9.21) we get

$$\begin{aligned} E_n(f) &= \int_a^b f[x_0,\dots,x_n,x]W'(x)dx = -\int_a^b \frac{d}{dx} f[x_0,\dots,x_n,x]W(x)dx \\ &= -\int_a^b \frac{f^{(n+2)}(\xi(x))}{(n+2)!}W(x)dx. \end{aligned}$$

In deriving the formula above we have used the following identity (see Exercise 4)

$$\frac{\mathrm{d}}{\mathrm{d}x} f[x_0,\dots,x_n,x] = f[x_0,\dots,x_n,x,x]. \tag{9.22}$$

Since $W(x) > 0$ for $a < x < b$ (see [IK66], p. 309), using the mean-value theorem we obtain

$$E_n(f) = -\frac{f^{(n+2)}(\xi)}{(n+2)!}\int_a^b W(x)dx = -\frac{f^{(n+2)}(\xi)}{(n+2)!}\int_a^b\int_a^x \omega_{n+1}(t)\ dt\ dx \tag{9.23}$$

where ξ lies within (a,b). Exchanging the order of integration, letting $s = x_0 + \tau h$, for $0 \le \tau \le n$, and recalling that $a = x_0$, $b = x_n$, yields

$$\begin{aligned} \int_a^b W(x)dx &= \int_a^b\int_s^b (s-x_0)\dots(s-x_n)dxds \\ &= \int_{x_0}^{x_n} (s-x_0)\dots(s-x_{n-1})(s-x_n)(x_n-s)ds \\ &= -h^{n+3}\int_0^n \tau(\tau-1)\dots(\tau-n+1)(\tau-n)^2 d\tau. \end{aligned}$$

Finally, letting $t = n - \tau$ and combining this result with (9.23), we get (9.19). $\diamond$

Relations (9.19) and (9.20) are *a priori estimates* for the quadrature error (see Chapter 2, Section 2.3). Their use in generating *a posteriori estimates* of the error in the frame of adaptive algorithms will be examined in Section 9.7.

In the case of closed Newton-Cotes formulae, we show in Table 9.3, for $1 \le n \le 6$, the degree of exactness (that we denote henceforth by r_n) and the absolute value of the constant $\mathcal{M}_n = M_n/(n+2)!$ (if n is even) or $\mathcal{K}_n = K_n/(n+1)!$ (if n is odd).

Example 9.2 The purpose of this example is to assess the importance of the regularity assumption on f for the error estimates (9.19) and (9.20). Consider the closed Newton-Cotes formulae, for $1 \le n \le 6$, to approximate the integral $\int_0^1 x^{5/2}dx = 2/7 \simeq 0.2857$. Since f is only $C^2([0,1])$, we do not expect a substantial increase of the accuracy as n gets larger. Actually, this is confirmed by Table 9.4, where the results obtained by running Program 74 are reported.

n	r_n	$\mathcal{M}_n$	$\mathcal{K}_n$	n	r_n	$\mathcal{M}_n$	$\mathcal{K}_n$	n	r_n	$\mathcal{M}_n$	$\mathcal{K}_n$
1	1		$\frac{1}{12}$	3	3		$\frac{3}{80}$	5	5		$\frac{275}{12096}$
2	3	$\frac{1}{90}$		4	5	$\frac{8}{945}$		6	7	$\frac{9}{1400}$	

TABLE 9.3. Degree of exactness and error constants for closed Newton-Cotes formulae

For $n = 1, \ldots, 6$, we have denoted by $E_n^c(f)$ the module of the absolute error, by q_n^c the computed order of infinitesimal and by q_n^s the corresponding theoretical value predicted by (9.19) and (9.20) under optimal regularity assumptions for f. As is clearly seen, q_n^c is definitely less than the potential theoretical value q_n^s. •

n	$E_n^c(f)$	q_n^c	q_n^s	n	$E_n^c(f)$	q_n^c	q_n^s
1	0.2143	3	3	4	$5.009 \cdot 10^{-5}$	4.7	7
2	$1.196 \cdot 10^{-3}$	3.2	5	5	$3.189 \cdot 10^{-5}$	2.6	7
3	$5.753 \cdot 10^{-4}$	3.8	5	6	$7.857 \cdot 10^{-6}$	3.7	9

TABLE 9.4. Error in the approximation of $\int_0^1 x^{5/2} dx$

Example 9.3 From a brief analysis of error estimates (9.19) and (9.20), we could be led to believe that only non-smooth functions can be a source of trouble when dealing with Newton-Cotes formulae. Thus, it is a little surprising to see results like those in Table 9.5, concerning the approximation of the integral

$$I(f) = \int_{-5}^{5} \frac{1}{1+x^2} dx = 2 \arctan 5 \simeq 2.747, \tag{9.24}$$

where $f(x) = 1/(1+x^2)$ is Runge's function (see Section 8.1.2), which belongs to $C^\infty(\mathbb{R})$. The results clearly demonstrate that the error remains almost unchanged as n grows. This is due to the fact that singularities on the imaginary axis may also affect the convergence properties of a quadrature formula. This is indeed the case with the function at hand, which exhibits two singularities at $\pm\sqrt{-1}$ (see [DR75], pp. 64-66). •

n	$E_n(f)$	n	$E_n(f)$	n	$E_n(f)$
1	0.8601	3	0.2422	5	0.1599
2	-1.474	4	0.1357	6	-0.4091

TABLE 9.5. Relative error $E_n(f) = [I(f) - I_n(f)]/I_n(f)$ in the approximate evaluation of (9.24) using closed Newton-Cotes formulae

To increase the accuracy of an interpolatory quadrature rule, it is by no means convenient to increase the value of n. By doing so, the same

drawbacks of Lagrange interpolation on equally spaced nodes would arise. For example, the weights of the closed Newton-Cotes formula with $n = 8$ do not have the same sign (see Table 9.6 and recall that $w_i = w_{n-i}$ for $i = 0, \ldots, n-1$).

n	w_0	w_1	w_2	w_3	w_4	r_n	M_n
8	$\frac{3956}{14175}$	$\frac{23552}{14175}$	$-\frac{3712}{14175}$	$\frac{41984}{14175}$	$-\frac{18160}{14175}$	9	$\frac{2368}{467775}$

TABLE 9.6. Weights of the closed Newton-Cotes formula with 9 nodes

This can give rise to numerical instabilities, due to rounding errors (see Chapter 2), and makes this formula useless in the practice, as happens for all the Newton-Cotes formulae using more than 8 nodes. As an alternative, one can resort to composite formulae, whose error analysis is addressed in Section 9.4, or to Gaussian formulae, which will be dealt with in Chapter 10 and which yield maximum degree of exactness with a non equally spaced nodes distribution.

The closed Newton-Cotes formulae, for $1 \leq n \leq 6$, are implemented in Program 74.

Program 74 - newtcot : Closed Newton-Cotes formulae

```
function int = newtcot(a,b,n,fun)
h=(b-a)/n; n2=fix(n/2);
if n > 6, disp('maximum value of n equal to 6  '); return; end
a03=1/3; a08=1/8; a45=1/45; a288=1/288; a140=1/140;
alpha=[0.5     0        0        0; ...
       a03     4*a03    0        0; ...
       3*a08   9*a08    0        0; ...
       14*a45  64*a45   24*a45   0; ...
       95*a288 375*a288 250*a288 0; ...
       41*a140 216*a140 27*a140  272*a140];
x=a; y(1)=eval(fun);
for j=2:n+1,    x=x+h; y(j)=eval(f); end;    int=0;
for j=1:n2+1,   int=int+y(j)*alpha(n,j);     end;
for j=n2+2:n+1, int=int+y(j)*alpha(n,n-j+2); end; int=int*h;
```

9.4 Composite Newton-Cotes Formulae

The examples of Section 9.2 have already pointed out that composite Newton-Cotes formulae can be constructed by replacing f with its composite Lagrange interpolating polynomial, introduced in Section 8.1.

The general procedure consists of partitioning the integration interval $[a,b]$ into m subintervals $T_j = [y_j, y_{j+1}]$ such that $y_j = a + jH$, where $H = (b-a)/m$ for $j = 0, \ldots, m$. Then, over each subinterval, an interpolatory formula with nodes $\{x_k^{(j)}, 0 \le k \le n\}$ and weights $\{\alpha_k^{(j)}, 0 \le k \le n\}$ is used. Since

$$I(f) = \int_a^b f(x)dx = \sum_{j=0}^{m-1} \int_{T_j} f(x)dx,$$

a composite interpolatory quadrature formula is obtained by replacing $I(f)$ with

$$I_{n,m}(f) = \sum_{j=0}^{m-1} \sum_{k=0}^{n} \alpha_k^{(j)} f(x_k^{(j)}). \tag{9.25}$$

The quadrature error is defined as $E_{n,m}(f) = I(f) - I_{n,m}(f)$. In particular, over each subinterval T_j one can resort to a Newton-Cotes formula with $n+1$ equally spaced nodes: in such a case, the weights $\alpha_k^{(j)} = hw_k$ are still independent of T_j.

Using the same notation as in Theorem 9.2, the following convergence result holds for composite formulae.

Theorem 9.3 *Let a composite Newton-Cotes formula, with n even, be used. If $f \in C^{n+2}([a,b])$, then*

$$E_{n,m}(f) = \frac{b-a}{(n+2)!} \frac{M_n}{(n+2)^{n+3}} H^{n+2} f^{(n+2)}(\xi) \tag{9.26}$$

where $\xi \in (a,b)$. Therefore, the quadrature error is an infinitesimal in H of order $n+2$ and the formula has degree of exactness equal to $n+1$.

For a composite Newton-Cotes formula, with n odd, if $f \in C^{n+1}([a,b])$

$$E_{n,m}(f) = \frac{b-a}{(n+1)!} \frac{K_n}{n^{n+2}} H^{n+1} f^{(n+1)}(\eta) \tag{9.27}$$

where $\eta \in (a,b)$. Thus, the quadrature error is an infinitesimal in H of order $n+1$ and the formula has degree of exactness equal to n.

Proof. We only consider the case where n is even. Using (9.19), and noticing that M_n does not depend on the integration interval, we get

$$E_{n,m}(f) = \sum_{j=0}^{m-1} \left[I(f)|_{T_j} - I_n(f)|_{T_j} \right] = \frac{M_n}{(n+2)!} \sum_{j=0}^{m-1} h_j^{n+3} f^{(n+2)}(\xi_j),$$

where, for $j = 0, \ldots, (m-1)$, $h_j = |T_j|/(n+2) = (b-a)/(m(n+2))$; this time, ξ_j is a suitable point of T_j. Since $(b-a)/m = H$, we obtain

$$E_{n,m}(f) = \frac{M_n}{(n+2)!} \frac{b-a}{m(n+2)^{n+3}} H^{n+2} \sum_{j=0}^{m-1} f^{(n+2)}(\xi_j),$$

from which, applying Theorem 9.1 with $u(x) = f^{(n+2)}(x)$ and $\delta_j = 1$ for $j = 0, \ldots, m-1$, (9.26) immediately follows. A similar procedure can be followed to prove (9.27). ◇

We notice that, for n fixed, $E_{n,m}(f) \to 0$ as $m \to \infty$ (i.e., as $H \to 0$). This ensures the convergence of the numerical integral to the exact value $I(f)$. We notice also that the degree of exactness of composite formulae coincides with that of simple formulae, whereas its order of infinitesimal (with respect to H) is reduced by 1 with respect to the order of infinitesimal (in h) of simple formulae.
In practical computations, it is convenient to resort to a local interpolation of low degree (typically $n \leq 2$, as done in Section 9.2), this leads to composite quadrature rules with positive weights, with a minimization of the rounding errors.

Example 9.4 For the same integral (9.24) considered in Example 9.3, we show in Table 9.7 the behavior of the absolute error as a function of the number of subintervals m, in the case of the composite midpoint, trapezoidal and Cavalieri-Simpson formulae. Convergence of $I_{n,m}(f)$ to $I(f)$ as m increases can be clearly observed. Moreover, we notice that $E_{0,m}(f) \simeq E_{1,m}(f)/2$ for $m \geq 32$ (see Exercise 1).

m	$\lvert E_{0,m}\rvert$	$\lvert E_{1,m}\rvert$	$\lvert E_{2,m}\rvert$
1	7.253	2.362	4.04
2	1.367	2.445	$9.65 \cdot 10^{-2}$
8	$3.90 \cdot 10^{-2}$	$3.77 \cdot 10^{-2}$	$1.35 \cdot 10^{-2}$
32	$1.20 \cdot 10^{-4}$	$2.40 \cdot 10^{-4}$	$4.55 \cdot 10^{-8}$
128	$7.52 \cdot 10^{-6}$	$1.50 \cdot 10^{-5}$	$1.63 \cdot 10^{-10}$
512	$4.70 \cdot 10^{-7}$	$9.40 \cdot 10^{-7}$	$6.36 \cdot 10^{-13}$

TABLE 9.7. Absolute error for composite quadratures in the computation of (9.24)

•

Convergence of $I_{n,m}(f)$ to $I(f)$ can be established under less stringent regularity assumptions on f than those required by Theorem 9.3. In this regard, the following result holds (see for the proof [IK66], pp. 341-343).

Property 9.1 *Let $f \in C^0([a,b])$ and assume that the weights $\alpha_k^{(j)}$ in (9.25) are nonnegative. Then*

$$\lim_{m\to\infty} I_{n,m}(f) = \int_a^b f(x)dx, \qquad \forall n \geq 0.$$

Moreover

$$\left| \int_a^b f(x)dx - I_{n,m}(f) \right| \leq 2(b-a)\Omega(f;H),$$

where

$$\Omega(f;H)=\sup\{|f(x)-f(y)|,\ x,y\in[a,b],\ x\neq y,\ |x-y|\leq H\}$$

is the module of continuity of function f*.*

9.5 Hermite Quadrature Formulae

Thus far we have considered quadrature formulae based on Lagrange interpolation (simple or composite). More accurate formulae can be devised by resorting to Hermite interpolation (see Section 8.4).

Suppose that $2(n+1)$ values $f(x_k)$, $f'(x_k)$ are available at $n+1$ distinct points $x_0,\dots,x_n$, then the Hermite interpolating polynomial of f is given by

$$H_{2n+1}f(x)=\sum_{i=0}^{n}f(x_i)\mathcal{L}_i(x)+\sum_{i=0}^{n}f'(x_i)\mathcal{M}_i(x), \tag{9.28}$$

where the polynomials $\mathcal{L}_k,\mathcal{M}_k\in\mathbb{P}_{2n+1}$ are defined, for $k=0,\dots,n$, as

$$\mathcal{L}_k(x)=\left[1-\frac{\omega''_{n+1}(x_k)}{\omega'_{n+1}(x_k)}(x-x_k)\right]l_k^2(x),\qquad \mathcal{M}_k(x)=(x-x_k)l_k^2(x).$$

Integrating (9.28) over $[a,b]$, we get the quadrature formula of type (9.4)

$$I_n(f)=\sum_{k=0}^{n}\alpha_k f(x_k)+\sum_{k=0}^{n}\beta_k f'(x_k) \tag{9.29}$$

where

$$\alpha_k=I(\mathcal{L}_k),\quad \beta_k=I(\mathcal{M}_k),\quad k=0,\dots,n.$$

Formula (9.29) has degree of exactness equal to $2n+1$. Taking $n=1$, the so-called *corrected trapezoidal formula* is obtained

$$I_1^{corr}(f)=\frac{b-a}{2}\left[f(a)+f(b)\right]+\frac{(b-a)^2}{12}\left[f'(a)-f'(b)\right] \tag{9.30}$$

with weights $\alpha_0=\alpha_1=(b-a)/2$, $\beta_0=(b-a)^2/12$ and $\beta_1=-\beta_0$. Assuming $f\in C^4([a,b])$, the quadrature error associated with (9.30) is

$$E_1^{corr}(f)=\frac{h^5}{720}f^{(4)}(\xi),\qquad h=b-a \tag{9.31}$$

with $\xi\in(a,b)$. Notice the increase of accuracy from $\mathcal{O}(h^3)$ to $\mathcal{O}(h^5)$ with respect to the corresponding expression (9.12) (of the same order as the

Cavalieri-Simpson formula (9.15)). The composite formula can be generated in a similar manner

$$I_{1,m}^{corr}(f) = \frac{b-a}{m}\left\{\frac{1}{2}\left[f(x_0)+f(x_m)\right]\right. \\ \left. +f(x_1)+\ldots+f(x_{m-1})\right\} + \frac{(b-a)^2}{12}\left[f'(a)-f'(b)\right], \tag{9.32}$$

where the assumption that $f \in C^1([a,b])$ gives rise to the cancellation of the first derivatives at the nodes x_k, with $k = 1, \ldots, m-1$.

Example 9.5 Let us check experimentally the error estimate (9.31) in the simple ($m = 1$) and composite ($m > 1$) cases, running Program 75 for the approximate computation of integral (9.18). Table 9.8 reports the behavior of the module of the absolute error as H is halved (that is, m is doubled) and the ratio $\mathcal{R}_m$ between two consecutive errors. This ratio, as happens in the case of Cavalieri-Simpson formula, tends to 16, demonstrating that formula (9.32) has order of infinitesimal equal to 4. Comparing Table 9.8 with the corresponding Table 9.1, we can also notice that $|E_{1,m}^{corr}(f)| \simeq 4|E_{2,m}(f)|$ (see Exercise 9). •

m	$E_{1,m}^{corr}(f)$	$\mathcal{R}_m$	m	$E_{1,m}^{corr}(f)$	$\mathcal{R}_m$	m	$E_{1,m}^{corr}(f)$	$\mathcal{R}_m$
1	3.4813		8	$4.4 \cdot 10^{-3}$	6.1	64	$1.1 \cdot 10^{-6}$	15.957
2	1.398	2.4	16	$2.9 \cdot 10^{-4}$	14.9	128	$7.3 \cdot 10^{-8}$	15.990
4	$2.72 \cdot 10^{-2}$	51.4	32	$1.8 \cdot 10^{-5}$	15.8	256	$4.5 \cdot 10^{-9}$	15.997

TABLE 9.8. Absolute error for the corrected trapezoidal formula in the computation of $I(f) = \int_0^{2\pi} xe^{-x}\cos(2x)dx$

The corrected composite trapezoidal quadrature is implemented in Program 75, where `dfun` contains the expression of the derivative of f.

Program 75 - trapmodc : Composite corrected trapezoidal formula

```
function int = trapmodc(a,b,m,fun,dfun)
h=(b-a)/m; x=[a:h:b]; y=eval(fun);
f1a=feval(dfun,a); f1b=feval(dfun,b);
int=h*(0.5*y(1)+sum(y(2:m))+0.5*y(m+1))+(h^2/12)*(f1a-f1b);
```

9.6 Richardson Extrapolation

The *Richardson extrapolation method* is a procedure which combines several approximations of a certain quantity α_0 in a smart way to yield a more accurate approximation of α_0. More precisely, assume that a method is available to approximate α_0 by a quantity $\mathcal{A}(h)$ that is computable for any

value of the parameter $h \neq 0$. Moreover, assume that, for a suitable $k \geq 0$, $\mathcal{A}(h)$ can be expanded as follows

$$\mathcal{A}(h) = \alpha_0 + \alpha_1 h + \ldots + \alpha_k h^k + \mathcal{R}_{k+1}(h), \tag{9.33}$$

where $|\mathcal{R}_{k+1}(h)| \leq C_{k+1} h^{k+1}$. The constants C_{k+1} and the coefficients α_i, for $i = 0, \ldots, k$, are independent of h. Henceforth, $\alpha_0 = \lim_{h \to 0} \mathcal{A}(h)$.

Writing (9.33) with δh instead of h, for $0 < \delta < 1$ (typically, $\delta = 1/2$), we get

$$\mathcal{A}(\delta h) = \alpha_0 + \alpha_1(\delta h) + \ldots + \alpha_k(\delta h)^k + \mathcal{R}_{k+1}(\delta h).$$

Subtracting (9.33) multiplied by δ from this expression then yields

$$\mathcal{B}(h) = \frac{\mathcal{A}(\delta h) - \delta \mathcal{A}(h)}{1 - \delta} = \alpha_0 + \widetilde{\alpha}_2 h^2 + \ldots + \widetilde{\alpha}_k h^k + \widetilde{\mathcal{R}}_{k+1}(h),$$

having defined, for $k \geq 2$, $\widetilde{\alpha}_i = \alpha_i(\delta^i - \delta)/(1 - \delta)$, for $i = 2, \ldots, k$ and $\widetilde{\mathcal{R}}_{k+1}(h) = [\mathcal{R}_{k+1}(\delta h) - \delta \mathcal{R}_{k+1}(h)]/(1 - \delta)$.
Notice that $\widetilde{\alpha}_i \neq 0$ iff $\alpha_i \neq 0$. In particular, if $\alpha_1 \neq 0$, then $\mathcal{A}(h)$ is a first-order approximation of α_0, while $\mathcal{B}(h)$ is at least second-order accurate. More generally, if $\mathcal{A}(h)$ is an approximation of α_0 of order p, then the quantity $\mathcal{B}(h) = [\mathcal{A}(\delta h) - \delta^p \mathcal{A}(h)]/(1 - \delta^p)$ approximates α_0 up to order $p + 1$ (at least).
Proceeding by induction, the following Richardson extrapolation algorithm is generated: setting $n \geq 0$, $h > 0$ and $\delta \in (0, 1)$, we construct the sequences

$$\begin{aligned} &\mathcal{A}_{m,0} = \mathcal{A}(\delta^m h), && m = 0, \ldots, n, \\ &\mathcal{A}_{m,q+1} = \frac{\mathcal{A}_{m,q} - \delta^{q+1} \mathcal{A}_{m-1,q}}{1 - \delta^{q+1}}, && q = 0, \ldots, n-1, \\ & && m = q+1, \ldots, n, \end{aligned} \tag{9.34}$$

which can be represented by the diagram below

$$\begin{array}{ccccccccccc}
\mathcal{A}_{0,0} & & & & & & & & & & \\
 & \searrow & & & & & & & & & \\
\mathcal{A}_{1,0} & \to & \mathcal{A}_{1,1} & & & & & & & & \\
 & \searrow & & \searrow & & & & & & & \\
\mathcal{A}_{2,0} & \to & \mathcal{A}_{2,1} & \to & \mathcal{A}_{2,2} & & & & & & \\
 & \searrow & & \searrow & & \searrow & & & & & \\
\mathcal{A}_{3,0} & \to & \mathcal{A}_{3,1} & \to & \mathcal{A}_{3,2} & \to & \mathcal{A}_{3,3} & & & & \\
 & \searrow & & \searrow & & \searrow & & \searrow & & & \\
\vdots & \ddots & & \ddots & & \ddots & & \ddots & & & \\
 & \searrow & & \searrow & & \searrow & & & \searrow & & \\
\mathcal{A}_{n,0} & \to & \mathcal{A}_{n,1} & \to & \mathcal{A}_{n,2} & \to & \mathcal{A}_{n,3} & \ldots & \to & \mathcal{A}_{n,n}
\end{array}$$

where the arrows indicate the way the terms which have been already computed contribute to the construction of the "new" ones.

The following result can be proved (see [Com95], Proposition 4.1).

Property 9.2 *For $n \geq 0$ and $\delta \in (0,1)$*

$$\mathcal{A}_{m,n} = \alpha_0 + \mathcal{O}((\delta^m h)^{n+1}), \qquad m = 0, \ldots, n. \tag{9.35}$$

In particular, for the terms in the first column ($n = 0$) the convergence rate to α_0 is $\mathcal{O}((\delta^m h))$, while for those of the last one it is $\mathcal{O}((\delta^m h)^{n+1})$, i.e., n times higher.

Example 9.6 Richardson extrapolation has been employed to approximate at $\overline{x} = 0$ the derivative of the function $f(x) = xe^{-x}\cos(2x)$, introduced in Example 9.1. For this purpose, algorithm (9.34) has been executed with $\mathcal{A}(h) = [f(\overline{x}+h) - f(\overline{x})]/h$, $\delta = 0.5$, $n = 5$ and $h = 0.1$. Table 9.9 reports the sequence of absolute errors $E_{m,k} = |\alpha_0 - \mathcal{A}_{m,k}|$. The results demonstrate that the error decays as predicted by (9.35). •

$E_{m,0}$	$E_{m,1}$	$E_{m,2}$	$E_{m,3}$	$E_{m,4}$	$E_{m,5}$
0.113	–	–	–	–	–
$5.3 \cdot 10^{-2}$	$6.1 \cdot 10^{-3}$	–	–	–	–
$2.6 \cdot 10^{-2}$	$1.7 \cdot 10^{-3}$	$2.2 \cdot 10^{-4}$	–	–	–
$1.3 \cdot 10^{-2}$	$4.5 \cdot 10^{-4}$	$2.8 \cdot 10^{-5}$	$5.5 \cdot 10^{-7}$	–	–
$6.3 \cdot 10^{-3}$	$1.1 \cdot 10^{-4}$	$3.5 \cdot 10^{-6}$	$3.1 \cdot 10^{-8}$	$3.0 \cdot 10^{-9}$	–
$3.1 \cdot 10^{-3}$	$2.9 \cdot 10^{-5}$	$4.5 \cdot 10^{-7}$	$1.9 \cdot 10^{-9}$	$9.9 \cdot 10^{-11}$	$4.9 \cdot 10^{-12}$

TABLE 9.9. Errors in the Richardson extrapolation for the approximate evaluation of $f'(0)$ where $f(x) = xe^{-x}\cos(2x)$

9.6.1 Romberg Integration

The *Romberg integration method* is an application of Richardson extrapolation to the composite trapezoidal rule. The following result, known as the Euler-MacLaurin formula, will be useful (for its proof see, e.g., [Ral65], pp. 131-133, and [DR75], pp. 106-111).

Property 9.3 *Let $f \in C^{2k+2}([a,b])$, for $k \geq 0$, and let us approximate $\alpha_0 = \int_a^b f(x)dx$ by the composite trapezoidal rule* (9.14)*. Letting $h_m = (b-a)/m$ for $m \geq 1$,*

$$\begin{aligned} I_{1,m}(f) = \alpha_0 \quad &+ \sum_{i=1}^{k} \frac{B_{2i}}{(2i)!} h_m^{2i} \left(f^{(2i-1)}(b) - f^{(2i-1)}(a) \right) \\ &+ \frac{B_{2k+2}}{(2k+2)!} h_m^{2k+2} (b-a) f^{(2k+2)}(\eta), \end{aligned} \tag{9.36}$$

where $\eta \in (a,b)$ and $B_{2j} = (-1)^{j-1} \left[\sum_{n=1}^{+\infty} 2/(2n\pi)^{2j} \right] (2j)!$, for $j \geq 1$, are the Bernoulli numbers.

Equation (9.36) is a special case of (9.33) where $h = h_m^2$ and $\mathcal{A}(h) = I_{1,m}(f)$; notice that *only even powers* of the parameter h appear in the expansion.

The Richardson extrapolation algorithm (9.34) applied to (9.36) gives

$$\begin{aligned} &\mathcal{A}_{m,0} = \mathcal{A}(\delta^m h), && m = 0, \dots, n, \\ &\mathcal{A}_{m,q+1} = \frac{\mathcal{A}_{m,q} - \delta^{2(q+1)}\mathcal{A}_{m-1,q}}{1 - \delta^{2(q+1)}}, && q = 0, \dots, n-1, \\ & && m = q+1, \dots, n. \end{aligned} \tag{9.37}$$

Setting $h = b - a$ and $\delta = 1/2$ into (9.37) and denoting by $T(h_s) = I_{1,s}(f)$ the composite trapezoidal formula (9.14) over $s = 2^m$ subintervals of width $h_s = (b-a)/2^m$, for $m \geq 0$, the algorithm (9.37) becomes

$$\begin{aligned} &\mathcal{A}_{m,0} = T((b-a)/2^m), && m = 0, \dots, n, \\ &\mathcal{A}_{m,q+1} = \frac{4^{q+1}\mathcal{A}_{m,q} - \mathcal{A}_{m-1,q}}{4^{q+1} - 1}, && q = 0, \dots, n-1, \\ & && m = q+1, \dots, n. \end{aligned}$$

This is the Romberg numerical integration algorithm. Recalling (9.35), the following convergence result holds for Romberg integration

$$\mathcal{A}_{m,n} = \int_a^b f(x)dx + \mathcal{O}(h_s^{2(n+1)}), \quad n \geq 0.$$

Example 9.7 Table 9.10 shows the results obtained by running Program 76 to compute the quantity α_0 in the two cases $\alpha_0^{(1)} = \int_0^\pi e^x \cos(x)dx = -(e^\pi + 1)/2$ and $\alpha_0^{(2)} = \int_0^1 \sqrt{x}dx = 2/3$.

The maximum size n has been set equal to 9. In the second and third columns we show the modules of the absolute errors $E_k^{(r)} = |\alpha_0^{(r)} - \mathcal{A}_{k+1,k+1}^{(r)}|$, for $r = 1, 2$ and $k = 0, \dots, 6$.

The convergence to zero is much faster for $E_k^{(1)}$ than for $E_k^{(2)}$. Indeed, the first integrand function is infinitely differentiable whereas the second is only continuous. •

k	$E_k^{(1)}$	$E_k^{(2)}$	k	$E_k^{(1)}$	$E_k^{(2)}$
0	22.71	0.1670	4	$8.923 \cdot 10^{-7}$	$1.074 \cdot 10^{-3}$
1	0.4775	$2.860 \cdot 10^{-2}$	5	$6.850 \cdot 10^{-11}$	$3.790 \cdot 10^{-4}$
2	$5.926 \cdot 10^{-2}$	$8.910 \cdot 10^{-3}$	6	$5.330 \cdot 10^{-14}$	$1.340 \cdot 10^{-4}$
3	$7.410 \cdot 10^{-5}$	$3.060 \cdot 10^{-3}$	7	0	$4.734 \cdot 10^{-5}$

TABLE 9.10. Romberg integration for the approximate evaluation of $\int_0^\pi e^x \cos(x)dx$ (error $E_k^{(1)}$) and $\int_0^1 \sqrt{x}dx$ (error $E_k^{(2)}$)

The Romberg algorithm is implemented in Program 76.

Program 76 - romberg : Romberg integration

```
function [A]=romberg(a,b,n,fun);
for i=1:(n+1), A(i,1)=trapezc(a,b,2^(i-1),fun); end;
for j=2:(n+1), for i=j:(n+1),
A(i,j)=(4^(j-1)*A(i,j-1)-A(i-1,j-1))/(4^(j-1)-1); end; end;
```

9.7 Automatic Integration

An *automatic numerical integration* program, or *automatic integrator*, is a set of algorithms which yield an approximation of the integral $I(f) = \int_a^b f(x)dx$, within a given tolerance, ε_a, or relative tolerance, ε_r, prescribed by the user.

With this aim, the program generates a sequence $\{\mathcal{I}_k, \mathcal{E}_k\}$, for $k = 1, \ldots, N$, where $\mathcal{I}_k$ is the approximation of $I(f)$ at the k-th step of the computational process, $\mathcal{E}_k$ is an estimate of the error $I(f) - \mathcal{I}_k$, and is N a suitable fixed integer.

The sequence terminates at the s-th level, with $s \leq N$, such that the automatic integrator fulfills the following requirement on the accuracy

$$\max\left\{\varepsilon_a, \varepsilon_r|\widetilde{I}(f)|\right\} \geq |\mathcal{E}_s| (\simeq |I(f) - \mathcal{I}_s|), \tag{9.38}$$

where $\widetilde{I}(f)$ is a reasonable guess of the integral $I(f)$ provided as an input datum by the user. Otherwise, the integrator returns the last computed approximation $\mathcal{I}_N$, together with a suitable error message that warns the user of the algorithm's failure to converge.
Ideally, an automatic integrator should:

(a) provide a reliable criterion for determining $|\mathcal{E}_s|$ that allows for monitoring the convergence check (9.38);

(b) ensure an *efficient implementation*, which minimizes the number of functional evaluations for yielding the desired approximation $\mathcal{I}_s$.

In computational practice, for each $k \geq 1$, moving from level k to level $k+1$ of the automatic integration process can be done according to two different strategies, which we define as *non adaptive* or *adaptive*.

In the non adaptive case, the law of distribution of the quadrature nodes is fixed *a priori* and the quality of the estimate $\mathcal{I}_k$ is refined by increasing the number of nodes corresponding to each level of the computational process. An example of an automatic integrator that is based on such a procedure is provided by the composite Newton-Cotes formulae on m and

$2m$ subintervals, respectively, at levels k and $k+1$, as described in Section 9.7.1.

In the adaptive case, the positions of the nodes is not set *a priori*, but at each level k of the process they depend on the information that has been stored during the previous $k-1$ levels. An adaptive automatic integration algorithm is performed by partitioning the interval $[a, b]$ into successive subdivisions which are characterized by a nonuniform density of the nodes, this density being typically higher in a neighborhood of strong gradients or singularities of f. An example of an adaptive integrator based on the Cavalieri-Simpson formula is described in Section 9.7.2.

9.7.1 Non Adaptive Integration Algorithms

In this section, we employ the composite Newton-Cotes formulae. Our aim is to devise a criterion for estimating the absolute error $|I(f) - \mathcal{I}_k|$ by using Richardson extrapolation. From (9.26) and (9.27) it turns out that, for $m \geq 1$ and $n \geq 0$, $I_{n,m}(f)$ has order of infinitesimal equal to H^{n+p}, with $p = 2$ for n even and $p = 1$ for n odd, where m, n and $H = (b-a)/m$ are the number of partitions of $[a, b]$, the number of quadrature nodes over each subinterval and the constant length of each subinterval, respectively. By doubling the value of m (i.e., halving the stepsize H) and proceeding by extrapolation, we get

$$I(f) - I_{n,2m}(f) \simeq \frac{1}{2^{n+p}} \left[I(f) - I_{n,m}(f) \right]. \tag{9.39}$$

The use of the symbol $\simeq$ instead of $=$ is due to the fact that the point ξ or η, where the derivative in (9.26) and (9.27) must be evaluated, changes when passing from m to $2m$ subintervals. Solving (9.39) with respect to $I(f)$ yields the following *absolute error estimate* for $I_{n,2m}(f)$

$$I(f) - I_{n,2m}(f) \simeq \frac{I_{n,2m}(f) - I_{n,m}(f)}{2^{n+p} - 1}. \tag{9.40}$$

If the composite Simpson rule is considered (i.e., $n = 2$), (9.40) predicts a reduction of the absolute error by a factor of 15 when passing from m to $2m$ subintervals. Notice also that only 2^{m-1} extra functional evaluations are needed to compute the new approximation $I_{1,2m}(f)$ starting from $I_{1,m}(f)$. Relation (9.40) is an instance of an *a posteriori error estimate* (see Chapter 2, Section 2.3). It is based on the combined use of an *a priori estimate* (in this case, (9.26) or (9.27)) and of two evaluations of the quantity to be approximated (the integral $I(f)$) for two different values of the discretization parameter (that is, $H = (b-a)/m$).

Example 9.8 Let us employ the *a posteriori* estimate (9.40) in the case of the composite Simpson formula ($n = p = 2$), for the approximation of the integral

$$\int_0^\pi (e^{x/2} + \cos 4x)dx = 2(e^\pi - 1) \simeq 7.621,$$

where we require the absolute error to be less than 10^{-4}. For $k = 0, 1, \ldots$, set $h_k = (b-a)/2^k$ and denote by $I_{2,m(k)}(f)$ the integral of f which is computed using the composite Simpson formula on a grid of size h_k with $m(k) = 2^k$ intervals. We can thus assume as a conservative estimate of the quadrature error the following quantity

$$|E_k| = |I(f) - I_{2,m(k)}(f)| \simeq \frac{1}{10}|I_{2,2m(k)}(f) - I_{2,m(k)}(f)| = |\mathcal{E}_k|, \qquad k \geq 1. \tag{9.41}$$

Table 9.11 shows the sequence of the estimated errors $|\mathcal{E}_k|$ and of the corresponding absolute errors $|E_k|$ that have been *actually* made by the numerical integration process. Notice that, when convergence has been achieved, the error estimated by (9.41) is definitely higher than the actual error, due to the conservative choice above. •

k	$\lvert\mathcal{E}_k\rvert$	$\lvert E_k\rvert$	k	$\lvert\mathcal{E}_k\rvert$	$\lvert E_k\rvert$
0		3.156	2	0.10	$4.52 \cdot 10^{-5}$
1	0.42	1.047	3	$5.8 \cdot 10^{-6}$	$2 \cdot 10^{-9}$

TABLE 9.11. Non adaptive automatic Simpson rule for the approximation of $\int_0^\pi (e^{x/2} + \cos 4x)dx$

An alternative approach for fulfilling the constraints (a) and (b) consists of employing a *nested sequence* of special Gaussian quadratures $I_k(f)$ (see Chapter 10), having increasing degree of exactness for $k = 1, \ldots, N$. These formulae are constructed in such a way that, denoting by $\mathcal{S}_{n_k} = \{x_1, \ldots, x_{n_k}\}$ the set of quadrature nodes relative to quadrature $I_k(f)$, $\mathcal{S}_{n_k} \subset \mathcal{S}_{n_{k+1}}$ for any $k = 1, \ldots, N-1$. As a result, for $k \geq 1$, the formula at the $k+1$-th level employs *all* the nodes of the formula at level k and this makes nested formulae quite effective for computer implementation.

As an example, we recall the Gauss-Kronrod formulae with 10, 21, 43 and 87 points, that are available in [PdKÜK83] (in this case, $N = 4$). The Gauss-Kronrod formulae have degree of exactness r_{n_k} (optimal) equal to $2n_k - 1$, where n_k is the number of nodes for each formula, with $n_1 = 10$ and $n_{k+1} = 2n_k + 1$ for $k = 1, 2, 3$. The criterion for devising an error estimate is based on comparing the results given by two successive formulae $I_{n_k}(f)$ and $I_{n_{k+1}}(f)$ with $k = 1, 2, 3$, and then terminating the computational process at the level k such that (see also [DR75], p. 321)

$$|\mathcal{I}_{k+1} - \mathcal{I}_k| \leq \max\{\varepsilon_a, \varepsilon_r|\mathcal{I}_{k+1}|\}.$$

9.7.2 Adaptive Integration Algorithms

The goal of an adaptive integrator is to yield an approximation of $I(f)$ within a fixed tolerance ε by a *non uniform* distribution of the integration stepsize along the interval $[a, b]$. An optimal algorithm is able to adapt automatically the choice of the steplength according to the behavior of the integrand function, by increasing the density of the quadrature nodes where the function exhibits stronger variations.

In view of describing the method, it is convenient to restrict our attention to a generic subinterval $[\alpha, \beta] \subseteq [a, b]$. Recalling the error estimates for the Newton-Cotes formulae, it turns out that the evaluation of the derivatives of f, up to a certain order, is needed to set a stepsize h such that a fixed accuracy is ensured, say $\varepsilon(\beta-\alpha)/(b-a)$. This procedure, which is unfeasible in practical computations, is carried out by an automatic integrator as follows. We consider throughout this section the Cavalieri-Simpson formula (9.15), although the method can be extended to other quadrature rules.

Set $I_f(\alpha, \beta) = \int_\alpha^\beta f(x)dx$, $h = h_0 = (\beta - \alpha)/2$ and

$$S_f(\alpha, \beta) = (h_0/3)\left[f(\alpha) + 4f(\alpha + h_0) + f(\beta)\right].$$

From (9.16) we get

$$I_f(\alpha, \beta) - S_f(\alpha, \beta) = -\frac{h_0^5}{90} f^{(4)}(\xi), \tag{9.42}$$

where ξ is a point in (α, β). To estimate the error $I_f(\alpha, \beta) - S_f(\alpha, \beta)$ *without* using explicitly the function $f^{(4)}$ we employ again the Cavalieri-Simpson formula over the union of the two subintervals $[\alpha, (\alpha + \beta)/2]$ and $[(\alpha + \beta)/2, \beta]$, obtaining, for $h = h_0/2 = (\beta - \alpha)/4$

$$I_f(\alpha, \beta) - S_{f,2}(\alpha, \beta) = -\frac{(h_0/2)^5}{90}\left(f^{(4)}(\xi) + f^{(4)}(\eta)\right),$$

where $\xi \in (\alpha, (\alpha + \beta)/2)$, $\eta \in ((\alpha + \beta)/2, \beta)$ and $S_{f,2}(\alpha, \beta) = S_f(\alpha, (\alpha + \beta)/2) + S_f((\alpha + \beta)/2, \beta)$.

Let us now make the assumption that $f^{(4)}(\xi) \simeq f^{(4)}(\eta)$ (which is true, in general, only if the function $f^{(4)}$ does not vary "too much" on $[\alpha, \beta]$). Then,

$$I_f(\alpha, \beta) - S_{f,2}(\alpha, \beta) \simeq -\frac{1}{16}\frac{h_0^5}{90} f^{(4)}(\xi), \tag{9.43}$$

with a reduction of the error by a factor 16 with respect to (9.42), corresponding to the choice of a steplength of doubled size. Comparing (9.42) and (9.43), we get the estimate

$$\frac{h_0^5}{90} f^{(4)}(\xi) \simeq \frac{16}{15}\mathcal{E}_f(\alpha, \beta),$$

where $\mathcal{E}_f(\alpha,\beta) = S_f(\alpha,\beta) - S_{f,2}(\alpha,\beta)$. Then, from (9.43), we have

$$|I_f(\alpha,\beta) - S_{f,2}(\alpha,\beta)| \simeq \frac{|\mathcal{E}_f(\alpha,\beta)|}{15}. \tag{9.44}$$

We have thus obtained a formula that allows for easily *computing* the error made by using composite Cavalieri-Simpson numerical integration on the generic interval $[\alpha,\beta]$. Relation (9.44), as well as (9.40), is another instance of an *a posteriori error estimate*. It combines the use of an *a priori estimate* (in this case, (9.16)) and of two evaluations of the quantity to be approximated (the integral $I(f)$) for two different values of the discretization parameter h.

In the practice, it might be convenient to assume a more conservative error estimate, precisely

$$|I_f(\alpha,\beta) - S_{f,2}(\alpha,\beta)| \simeq |\mathcal{E}_f(\alpha,\beta)|/10.$$

Moreover, to ensure a global accuracy on $[a,b]$ equal to the fixed tolerance ε, it will suffice to enforce that the error $\mathcal{E}_f(\alpha,\beta)$ satisfies on each single subinterval $[\alpha,\beta] \subseteq [a,b]$ the following constraint

$$\frac{|\mathcal{E}_f(\alpha,\beta)|}{10} \leq \varepsilon \frac{\beta-\alpha}{b-a}. \tag{9.45}$$

The adaptive automatic integration algorithm can be described as follows. Denote by:

1. A: the *active* integration interval, i.e., the interval where the integral is being computed;
2. S: the integration interval already examined, for which the error test (9.45) has been successfully passed;
3. N: the integration interval yet to be examined.

At the beginning of the integration process we have $N = [a,b]$, $A = N$ and $S = \emptyset$, while the situation at the generic step of the algorithm is depicted in Figure 9.3. Set $J_S(f) \simeq \int_a^\alpha f(x)dx$, with $J_S(f) = 0$ at the beginning of the process; if the algorithm successfully terminates, $J_S(f)$ yields the desired approximation of $I(f)$. We also denote by $J_{(\alpha,\beta)}(f)$ the approximate integral of f over the "active" interval $[\alpha,\beta]$. This interval is drawn in bold in Figure 9.3. At each step of the adaptive integration method the following decisions are taken:

1. if the local error test (9.45) is passed, then:
   (i) $J_S(f)$ is increased by $J_{(\alpha,\beta)}(f)$, that is, $J_S(f) \leftarrow J_S(f) + J_{(\alpha,\beta)}(f)$;
   (ii) we let $S \leftarrow S \cup A$, $A = N$ (corresponding to the path (I) in Figure 9.3), $\beta = b$.

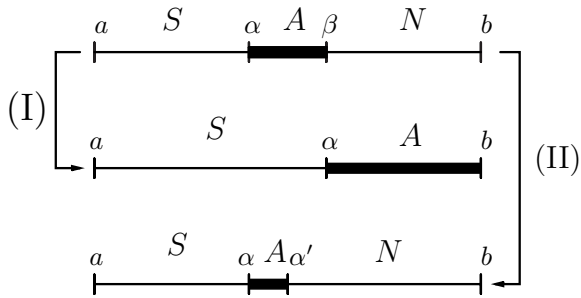

FIGURE 9.3. Distribution of the integration intervals at the generic step of the adaptive algorithm and updating of the integration grid

2. If the local error test (9.45) fails, then:

   (j) A is halved, and the new active interval is set to $A = [\alpha, \alpha']$ with $\alpha' = (\alpha + \beta)/2$ (corresponding to the path (II) in Figure 9.3);

   (jj) we let $N \leftarrow N \cup [\alpha', \beta]$, $\beta \leftarrow \alpha'$;

   (jjj) a new error estimate is provided.

In order to prevent the algorithm from generating too small stepsizes, it is convenient to monitor the width of A and warn the user, in case of an excessive reduction of the steplength, about the presence of a possible singularity in the integrand function (see Section 9.8).

Example 9.9 Let us employ Cavalieri-Simpson adaptive integration for computing the integral

$$\begin{aligned} I(f) &= \int_{-3}^{4} \tan^{-1}(10x)dx \\ &= 4\tan^{-1}(40) + 3\tan^{-1}(-30) - (1/20)\log(16/9) \simeq 1.54201193. \end{aligned}$$

Running Program 77 with `tol` $= 10^{-4}$ and `hmin` $= 10^{-3}$ yields an approximation of the integral with an absolute error of $2.104 \cdot 10^{-5}$. The algorithm performs 77 functional evaluations, corresponding to partitioning the interval $[a, b]$ into 38 nonuniform subintervals. We notice that the corresponding composite formula with uniform stepsize would have required 128 subintervals with an absolute error of $2.413 \cdot 10^{-5}$.

In Figure 9.4 (left) we show, together with the plot of the integrand function, the distribution of the quadrature nodes as a function of x, while on the right the integration step density (piecewise constant) $\Delta_h(x)$ is shown, defined as the inverse of the step size h over each active interval A. Notice the high value attained by Δ_h at $x = 0$, where the derivative of the integrand function is maximum. •

The adaptive algorithm described above is implemented in Program 77. Among the input parameters, `hmin` is the minimum admissible value of the integration steplength. In output the program returns the approximate

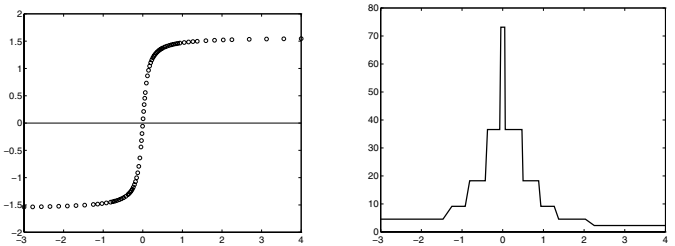

FIGURE 9.4. Distribution of quadrature nodes (left); density of the integration stepsize in the approximation of the integral of Example 9.9 (right)

value of the integral `integ`, the total number of functional evaluations `nfv` and the set of integration points `xfv`.

Program 77 - simpadpt : Adaptive Cavalieri-Simpson formula

```
function [integ,xfv,nfv]=simpadpt(a,b,tol,fun,hmin);
integ=0; level=0; i=1; alfa(i)=a; beta(i)=b;
step=(beta(i)-alfa(i))/4; nfv=0;
for k=1:5, x=a+(k-1)*step; f(i,k)=eval(fun); nfv=nfv+1; end
while (i > 0),
 S=0;  S2=0; h=(beta(i)-alfa(i))/2; S=(h/3)*(f(i,1)+4*f(i,3)+f(i,5));
 h=h/2; S2=(h/3)*(f(i,1)+4*f(i,2)+f(i,3));
 S2=S2+(h/3)*(f(i,3)+4*f(i,4)+f(i,5));
 tolrv=tol*(beta(i)-alfa(i))/(b-a); errrv=abs(S-S2)/10;
 if (errrv > tolrv)
  i=i+1; alfa(i)=alfa(i-1); beta(i)=(alfa(i-1)+beta(i-1))/2;
  f(i,1)=f(i-1,1);f(i,3)=f(i-1,2);f(i,5)=f(i-1,3);len=abs(beta(i)-alfa(i));
  if (len >= hmin),
   if (len <= 11*hmin)
    disp(' Steplength close to hmin  '),
    str=sprintf('The approximate integral is %12.7e',integ);disp(str),end;
    step=len/4; x=alfa(i)+step; f(i,2)=eval(fun);
    nfv=nfv+1; x=beta(i)-step; f(i,4)=eval(fun); nfv=nfv+1;
   else, xfv=xfv'; disp(' Too small steplength   ')
    str=sprintf('The approximate integral is %12.7e',integ);
    disp(str), return
  end, else
  integ=integ+S2; level=level+1; if (level==1),
  for k=1:5, xfv(k)=alfa(i)+(k-1)*h; end; ist=5;
 else, for k=1:4, xfv(ist+k)=alfa(i)+k*h; end; ist=ist+4; end;
  if (beta(i)==b), xfv=xfv';
  str=sprintf('The approximate integral is %12.7e',integ);
  disp(str), return, end; i=i-1; alfa(i)=beta(i+1);
  f(i,1)=f(i+1,5); f(i,3)=f(i,4); step=abs(beta(i)-alfa(i))/4;
  x=alfa(i)+step; f(i,2)=eval(fun); nfv=nfv+1; x=beta(i)-step;
  f(i,4)=eval(fun); nfv=nfv+1;
```

```
 end
end
```

9.8 Singular Integrals

In this section we extend our analysis to deal with *singular integrals*, arising when f has finite jumps or is even infinite at some point. Besides, we will consider the case of integrals of bounded functions over unbounded intervals. We briefly address the most relevant numerical techniques for properly handling these integrals.

9.8.1 Integrals of Functions with Finite Jump Discontinuities

Let c be a *known* point within $[a, b]$ and assume that f is a continuous and bounded function in $[a, c)$ and $(c, b]$, with finite jump $f(c^+) - f(c^-)$. Since

$$I(f) = \int_a^b f(x)dx = \int_a^c f(x)dx + \int_c^b f(x)dx, \tag{9.46}$$

any integration formula of the previous sections can be used on $[a, c^-]$ and $[c^+, b]$ to furnish an approximation of $I(f)$. We proceed similarly if f admits a *finite* number of jump discontinuities within $[a, b]$.

When the position of the discontinuity points of f is *not* known *a priori*, a preliminary analysis of the graph of the function should be carried out. Alternatively, one can resort to an adaptive integrator that is able to detect the presence of discontinuities when the integration steplength falls below a given tolerance (see Section 9.7.2).

9.8.2 Integrals of Infinite Functions

Let us deal with the case in which $\lim_{x\to a^+} f(x) = \infty$; similar considerations hold when f is infinite as $x \to b^-$, while the case of a point of singularity c internal to the interval $[a, b]$ can be recast to one of the previous two cases owing to (9.46). Assume that the integrand function is of the form

$$f(x) = \frac{\phi(x)}{(x-a)^\mu}, \qquad 0 \le \mu < 1,$$

where ϕ is a function whose absolute value is bounded by M. Then

$$|I(f)| \le M \lim_{t\to a^+} \int_t^b \frac{1}{(x-a)^\mu} dx = M \frac{(b-a)^{1-\mu}}{1-\mu}.$$

Suppose we wish to approximate $I(f)$ up to a fixed tolerance δ. For this, let us describe the following two methods (for further details, see also [IK66], Section 7.6, and [DR75], Section 2.12 and Appendix 1).

Method 1. For any ε such that $0 < \varepsilon < (b-a)$, we write the singular integral as $I(f) = I_1 + I_2$, where

$$I_1 = \int_a^{a+\varepsilon} \frac{\phi(x)}{(x-a)^\mu} dx, \qquad I_2 = \int_{a+\varepsilon}^b \frac{\phi(x)}{(x-a)^\mu} dx.$$

The computation of I_2 is not troublesome. After replacing ϕ by its p-th order Taylor's expansion around $x = a$, we obtain

$$\phi(x) = \Phi_p(x) + \frac{(x-a)^{p+1}}{(p+1)!}\phi^{(p+1)}(\xi(x)), \qquad p \geq 0 \tag{9.47}$$

where $\Phi_p(x) = \sum_{k=0}^p \phi^{(k)}(a)(x-a)^k/k!$. Then

$$I_1 = \varepsilon^{1-\mu}\sum_{k=0}^p \frac{\varepsilon^k \phi^{(k)}(a)}{k!(k+1-\mu)} + \frac{1}{(p+1)!}\int_a^{a+\varepsilon} (x-a)^{p+1-\mu}\phi^{(p+1)}(\xi(x))dx.$$

Replacing I_1 by the finite sum, the corresponding error E_1 can be bounded as

$$|E_1| \leq \frac{\varepsilon^{p+2-\mu}}{(p+1)!(p+2-\mu)} \max_{a\leq x\leq a+\varepsilon} |\phi^{(p+1)}(x)|, \qquad p \geq 0. \tag{9.48}$$

For fixed p, the right side of (9.48) is an increasing function of ε. On the other hand, taking $\varepsilon < 1$ and assuming that the successive derivatives of ϕ do not grow too much as p increases, the same function is decreasing as p grows.

Let us next approximate I_2 using a composite Newton-Cotes formula with m subintervals and n quadrature nodes for each subinterval, n being an even integer. Recalling (9.26) and aiming at equidistributing the error δ between I_1 and I_2, it turns out that

$$|E_2| \leq \mathcal{M}^{(n+2)}(\varepsilon)\frac{b-a-\varepsilon}{(n+2)!}\frac{|M_n|}{n^{n+3}}\left(\frac{b-a-\varepsilon}{m}\right)^{n+2} = \delta/2, \tag{9.49}$$

where

$$\mathcal{M}^{(n+2)}(\varepsilon) = \max_{a+\varepsilon\leq x\leq b}\left|\frac{d^{n+2}}{dx^{n+2}}\left(\frac{\phi(x)}{(x-a)^\mu}\right)\right|.$$

The value of the constant $\mathcal{M}^{(n+2)}(\varepsilon)$ grows rapidly as ε tends to zero; as a consequence, (9.49) might require such a large number of subintervals $m_\varepsilon = m(\varepsilon)$ to make the method at hand of little practical use.

Example 9.10 Consider the singular integral (known as the *Fresnel integral*)

$$I(f) = \int_0^{\pi/2} \frac{\cos(x)}{\sqrt{x}} dx. \tag{9.50}$$

Expanding the integrand function in a Taylor's series around the origin and applying the theorem of integration by series, we get

$$I(f) = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k)!} \frac{1}{(2k+1/2)} (\pi/2)^{2k+1/2}.$$

Truncating the series at the first 10 terms, we obtain an approximate value of the integral equal to 1.9549.

Using the composite Cavalieri-Simpson formula, the *a priori* estimate (9.49) yields, as ε tends to zero and letting $n = 2$, $|M_2| = 4/15$,

$$m_\varepsilon \simeq \left[\frac{0.018}{\delta} \left(\frac{\pi}{2} - \varepsilon \right)^5 \varepsilon^{-9/2} \right]^{1/4}.$$

For $\delta = 10^{-4}$, taking $\varepsilon = 10^{-2}$, it turns out that 1140 (uniform) subintervals are needed, while for $\varepsilon = 10^{-4}$ and $\varepsilon = 10^{-6}$ the number of subintervals is $2 \cdot 10^5$ and $3.6 \cdot 10^7$, respectively.

As a comparison, running Program 77 (adaptive integration with Cavalieri-Simpson formula) with `a` $= \varepsilon = 10^{-10}$, `hmin` $= 10^{-12}$ and `tol` $= 10^{-4}$, we get the approximate value 1.955 for the integral at the price of 1057 functional evaluations, which correspond to 528 nonuniform subdivisions of the interval $[0, \pi/2]$. •

Method 2. Using the Taylor expansion (9.47) we obtain

$$I(f) = \int_a^b \frac{\phi(x) - \Phi_p(x)}{(x-a)^\mu} dx + \int_a^b \frac{\Phi_p(x)}{(x-a)^\mu} dx = I_1 + I_2.$$

Exact computation of I_2 yields

$$I_2 = (b-a)^{1-\mu} \sum_{k=0}^{p} \frac{(b-a)^k \phi^{(k)}(a)}{k!(k+1-\mu)}. \tag{9.51}$$

The integral I_1 is, for $p \geq 0$

$$I_1 = \int_a^b (x-a)^{p+1-\mu} \frac{\phi^{(p+1)}(\xi(x))}{(p+1)!} dx = \int_a^b g(x) dx. \tag{9.52}$$

Unlike the case of method 1, the integrand function g *does not* blow up at $x = a$, since its first p derivatives are finite at $x = a$. As a consequence, assuming we approximate I_1 using a composite Newton-Cotes formula, it is possible to give an estimate of the quadrature error, provided that $p \geq n+2$, for $n \geq 0$ even, or $p \geq n+1$, for n odd.

Example 9.11 Consider again the singular Fresnel integral (9.50), and assume we use the composite Cavalieri-Simpson formula for approximating I_1. We will take $p = 4$ in (9.51) and (9.52). Computing I_2 yields the value $(\pi/2)^{1/2}(2 - (1/5)(\pi/2)^2 + (1/108)(\pi/2)^4) \simeq 1.9588$. Applying the error estimate (9.26) with $n = 2$ shows that only 2 subdivisions of $[0, \pi/2]$ suffice for approximating I_1 up to an error $\delta = 10^{-4}$, obtaining the value $I_1 \simeq -0.0173$. As a whole, method 2 returns for (9.50) the approximate value 1.9415. •

9.8.3 Integrals over Unbounded Intervals

Let $f \in C^0([a, +\infty))$; should it exist and be finite, the following limit

$$\lim_{t \to +\infty} \int_a^t f(x)dx$$

is taken as being the value of the singular integral

$$I(f) = \int_a^{\infty} f(x)dx = \lim_{t \to +\infty} \int_a^t f(x)dx. \tag{9.53}$$

An analogous definition holds if f is continuous over $(-\infty, b]$, while for a function $f : \mathbb{R} \to \mathbb{R}$, integrable over any bounded interval, we let

$$\int_{-\infty}^{\infty} f(x)dx = \int_{-\infty}^{c} f(x)dx + \int_c^{+\infty} f(x)dx \tag{9.54}$$

if c is any real number and the two singular integrals on the right hand side of (9.54) are convergent. This definition is correct since the value of $I(f)$ *does not* depend on the choice of c.
A sufficient condition for f to be integrable over $[a, +\infty)$ is that

$$\exists \rho > 0, \text{ such that } \lim_{x \to +\infty} x^{1+\rho} f(x) = 0,$$

that is, we require f to be infinitesimal of order > 1 with respect to $1/x$ as $x \to \infty$. For the numerical approximation of (9.53) up to a tolerance δ, we consider the following methods, referring for further details to [DR75], Chapter 3.

Method 1. To compute (9.53), we can split $I(f)$ as $I(f) = I_1 + I_2$, where $I_1 = \int_a^c f(x)dx$ and $I_2 = \int_c^{\infty} f(x)dx$.

The end-point c, which can be taken arbitrarily, is chosen in such a way that the contribution of I_2 is negligible. Precisely, taking advantage of the asymptotic behavior of f, c is selected to guarantee that I_2 equals a fraction of the fixed tolerance, say, $I_2 = \delta/2$.

Then, I_1 will be computed up to an absolute error equal to $\delta/2$. This ensures that the global error in the computation of $I_1 + I_2$ is below the tolerance δ.

Example 9.12 Compute up to an error $\delta = 10^{-3}$ the integral

$$I(f) = \int_0^\infty \cos^2(x)e^{-x}dx = 3/5.$$

For any given $c > 0$, we have $I_2 = \int_c^\infty \cos^2(x)e^{-x}dx \leq \int_c^\infty e^{-x}dx = e^{-c}$; requiring that $e^{-c} = \delta/2$, one gets $c \simeq 7.6$. Then, assuming we use the composite trapezoidal formula for approximating I_1, thanks to (9.27) with $n = 1$ and $M = \max_{0\leq x\leq c} |f''(x)| \simeq 1.04$, we obtain $m \geq \left(Mc^3/(6\delta)\right)^{1/2} = 277$.

Program 72 returns the value $\mathcal{I}_1 \simeq 0.599905$, instead of the exact value $I_1 = 3/5 - e^{-c}(\cos^2(c) - (\sin(2c) + 2\cos(2c))/5) \simeq 0.599842$, with an absolute error of about $6.27 \cdot 10^{-5}$. The global numerical outcome is thus $\mathcal{I}_1 + I_2 \simeq 0.600405$, with an absolute error with respect to $I(f)$ equal to $4.05 \cdot 10^{-4}$. •

Method 2. For any real number c, we let $I(f) = I_1 + I_2$, as for method 1, then we introduce the change of variable $x = 1/t$ in order to transform I_2 into an integral over the *bounded* interval $[0, 1/c]$

$$I_2 = \int_0^{1/c} f(t)t^{-2}dt = \int_0^{1/c} g(t)dt. \tag{9.55}$$

If $g(t)$ is not singular at $t = 0$, (9.55) can be treated by any quadrature formula introduced in this chapter. Otherwise, one can resort to the integration methods considered in Section 9.8.2.

Method 3. Gaussian interpolatory formulae are used, where the integration nodes are the zeros of Laguerre and Hermite orthogonal polynomials (see Section 10.5).

9.9 Multidimensional Numerical Integration

Let Ω be a bounded domain in $\mathbb{R}^2$ with a sufficiently smooth boundary. We consider the problem of approximating the integral $I(f) = \int_\Omega f(x, y)dxdy$, where f is a continuous function in $\overline{\Omega}$. For this purpose, in Sections 9.9.1 and 9.9.2 we address two methods.

The first method applies when Ω is a *normal* domain with respect to a coordinate axis. It is based on the reduction formula for double integrals and consists of using one-dimensional quadratures along both coordinate direction. The second method, which applies when Ω is a polygon, consists of employing composite quadratures of low degree on a triangular decomposition of the domain Ω. Section 9.9.3 briefly addresses the Monte Carlo

method, which is particularly well-suited to integration in several dimensions.

9.9.1 The Method of Reduction Formula

Let Ω be a normal domain with respect to the x axis, as drawn in Figure 9.5, and assume for the sake of simplicity that $\phi_2(x) > \phi_1(x)$, $\forall x \in [a, b]$.

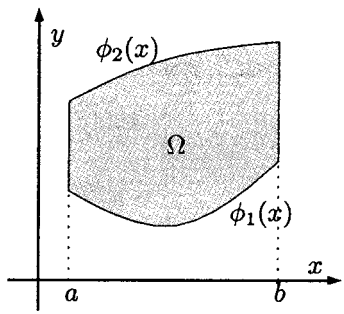

FIGURE 9.5. Normal domain with respect to x axis

The reduction formula for double integrals gives (with obvious choice of notation)

$$I(f) = \int_a^b \int_{\phi_1(x)}^{\phi_2(x)} f(x, y)dydx = \int_a^b F_f(x)dx. \tag{9.56}$$

The integral $I(F_f) = \int_a^b F_f(x)dx$ can be approximated by a composite quadrature rule using M_x subintervals $\{J_k,\, k = 1, \ldots, M_x\}$, of width $H = (b - a)/M_x$, and in each subinterval $n_x^{(k)} + 1$ nodes $\{x_i^k,\, i = 0, \ldots, n_x^{(k)}\}$. Thus, in the x direction we can write

$$I_{n_x}^c(f) = \sum_{k=1}^{M_x} \sum_{i=0}^{n_x^{(k)}} \alpha_i^k F_f(x_i^k),$$

where the coefficients α_i^k are the quadrature weights on each subinterval J_k. For each node x_i^k, the approximate evaluation of the integral $F_f(x_i^k)$ is then carried out by a composite quadrature using M_y subintervals $\{J_m,\, m = 1, \ldots, M_y\}$, of width $h_i^k = (\phi_2(x_i^k) - \phi_1(x_i^k))/M_y$ and in each subinterval $n_y^{(m)} + 1$ nodes $\{y_{j,m}^{i,k},\, j = 0, \ldots, n_y^{(m)}\}$.

In the particular case $M_x = M_y = M$, $n_x^{(k)} = n_y^{(m)} = 0$, for $k, m = 1, \ldots, M$, the resulting quadrature formula is the *midpoint reduction formula*

$$I_{0,0}^c(f) = H \sum_{k=1}^{M} h_0^k \sum_{m=1}^{M} f(x_0^k, y_{0,m}^{0,k}),$$

where $H = (b-a)/M$, $x_0^k = a + (k-1/2)H$ for $k = 1, \ldots, M$ and $y_{0,m}^{0,k} = \phi_1(x_0^k) + (m-1/2)h_0^k$ for $m = 1, \ldots, M$. With a similar procedure the *trapezoidal reduction formula* can be constructed along the coordinate directions (in that case, $n_x^{(k)} = n_y^{(m)} = 1$, for $k, m = 1, \ldots, M$).
The efficiency of the approach can obviously be increased by employing the adaptive method described in Section 9.7.2 to suitably allocate the quadrature nodes x_i^k and $y_{j,m}^{i,k}$ according to the variations of f over the domain Ω. The use of the reduction formulae above becomes less and less convenient as the dimension d of the domain $\Omega \subset \mathbb{R}^d$ gets larger, due to the large increase in the computational effort. Indeed, if any simple integral requires N functional evaluations, the overall cost would be equal to N^d.

The midpoint and trapezoidal reduction formulae for approximating the integral (9.56) are implemented in Programs 78 and 79. For the sake of simplicity, we set $M_x = M_y = M$. The variables `phi1` and `phi2` contain the expressions of the functions ϕ_1 and ϕ_2 which delimitate the integration domain.

Program 78 - redmidpt : Midpoint reduction formula

```
function inte=redmidpt(a,b,phi1,phi2,m,fun)
H=(b-a)/m; xx=[a+H/2:H:b]; dim=max(size(xx));
for i=1:dim, x=xx(i); d=eval(phi2); c=eval(phi1); h=(d-c)/m;
y=[c+h/2:h:d]; w=eval(fun); psi(i)=h*sum(w(1:m)); end;
inte=H*sum(psi(1:m));
```

Program 79 - redtrap : Trapezoidal reduction formula

```
function inte=redtrap(a,b,phi1,phi2,m,fun)
H=(b-a)/m; xx=[a:H:b]; dim=max(size(xx));
for i=1:dim, x=xx(i); d=eval(phi2); c=eval(phi1); h=(d-c)/m;
 y=[c:h:d]; w=eval(fun); psi(i)=h*(0.5*w(1)+sum(w(2:m))+0.5*w(m+1));
end; inte=H*(0.5*psi(1)+sum(psi(2:m))+0.5*psi(m+1));
```

9.9.2 Two-Dimensional Composite Quadratures

In this section we extend to the two-dimensional case the composite interpolatory quadratures that have been considered in Section 9.4. We assume that Ω is a convex polygon on which we introduce a *triangulation* $\mathcal{T}_h$ of N_T triangles or *elements*, such that $\overline{\Omega} = \bigcup_{T \in \mathcal{T}_h} T$, where the parameter $h > 0$ is the maximum edge-length in $\mathcal{T}_h$ (see Section 8.5.2).
Exactly as happens in the one-dimensional case, interpolatory composite quadrature rules on triangles can be devised by replacing $\int_\Omega f(x,y)dxdy$ with $\int_\Omega \Pi_h^k f(x,y)dxdy$, where, for $k \geq 0$, $\Pi_h^k f$ is the composite interpolating polynomial of f on the triangulation $\mathcal{T}_h$ introduced in Section 8.5.2.

For an efficient evaluation of this last integral, we employ the property of additivity which, combined with (8.38), leads to the following interpolatory composite rule

$$\begin{aligned} I_k^c(f) &= \int_\Omega \Pi_h^k f(x,y)dxdy = \sum_{T\in\mathcal{T}_h}\int_T \Pi_T^k f(x,y)dxdy = \sum_{T\in\mathcal{T}_h} I_k^T(f) \\ &= \sum_{T\in\mathcal{T}_h}\sum_{j=0}^{d_k-1} f(\tilde{\mathbf{z}}_j^T)\int_T l_j^T(x,y)dxdy = \sum_{T\in\mathcal{T}_h}\sum_{j=0}^{d_k-1}\alpha_j^T f(\tilde{\mathbf{z}}_j^T). \end{aligned} \tag{9.57}$$

The coefficients $\alpha_T^{(j)}$ and the points $\tilde{\mathbf{z}}_T^{(j)}$ are called the *local* weights and nodes of the quadrature formula (9.57), respectively.

The weights $\alpha_T^{(j)}$ can be computed on the reference triangle $\hat{T}$ of vertices $(0,0)$, $(1,0)$ and $(0,1)$, as follows

$$\alpha_T^{(j)} = \int_T l_{j,T}(x,y)dxdy = 2|T|\int_{\hat{T}}\hat{l}_j(\hat{x},\hat{y})d\hat{x}d\hat{y}, \qquad j=0,\ldots,d_k-1,$$

where $|T|$ is the area of T. If $k=0$, we get $\alpha_T^{(0)} = |T|$, while if $k=1$ we have $\alpha_T^{(j)} = |T|/3$, for $j=0,1,2$.
Denoting respectively by $\mathbf{a}_T^{(j)}$ and $\mathbf{a}_T = \sum_{j=1}^3(\mathbf{a}_T^{(j)})/3$, for $j=1,2,3$, the vertices and the center of gravity of the triangle $T\in\mathcal{T}_h$, the following formulae are obtained.

Composite midpoint formula

$$I_0^c(f) = \sum_{T\in\mathcal{T}_h}|T|f(\mathbf{a}_T). \tag{9.58}$$

Composite trapezoidal formula

$$I_1^c(f) = \frac{1}{3}\sum_{T\in\mathcal{T}_h}|T|\sum_{j=1}^3 f(\mathbf{a}_T^{(j)}). \tag{9.59}$$

In view of the analysis of the quadrature error $E_k^c(f) = I(f) - I_k^c(f)$, we introduce the following definition.

Definition 9.1 The quadrature formula (9.57) has *degree of exactness equal to* n, with $n\geq 0$, if $I_k^{\widehat{T}}(p) = \int_{\widehat{T}} pdxdy$ for any $p\in\mathbb{P}_n(\widehat{T})$, where $\mathbb{P}_n(\widehat{T})$ is defined in (8.35). ■

The following result can be proved (see [IK66], pp. 361–362).

Property 9.4 *Assume that the quadrature rule* (9.57) *has degree of exactness on* Ω *equal to* n*, with* $n \geq 0$*, and that its weights are all nonnegative. Then, there exists a positive constant* K_n*, independent of* h*, such that*

$$|E_k^c(f)| \leq K_n h^{n+1} |\Omega| M_{n+1},$$

for any function $f \in C^{n+1}(\Omega)$*, where* M_{n+1} *is the maximum value of the modules of the derivatives of order* $n+1$ *of* f *and* $|\Omega|$ *is the area of* Ω*.*

The composite formulae (9.58) and (9.59) both have degrees of exactness equal to 1; then, due to Property 9.4, their order of infinitesimal with respect to h is equal to 2.
An alternative family of quadrature rules on triangles is provided by the so-called *symmetric formulae*. These are Gaussian formulae with n nodes and high degree of exactness, and exhibit the feature that the quadrature nodes occupy symmetric positions with respect to all corners of the reference triangle $\widehat{T}$ or, as happens for Gauss-Radau formulae, with respect to the straight line $\widehat{y} = \widehat{x}$.
Considering the generic triangle $T \in \mathcal{T}_h$ and denoting by $\mathbf{a}_{(j)}^T$, $j = 1, 2, 3$, the midpoints of the edges of T, two examples of symmetric formulae, having degree of exactness equal to 2 and 3, respectively, are the following

$$I_3(f) = \frac{|T|}{3} \sum_{j=1}^{3} f(\mathbf{a}_{(j)}^T), \qquad n = 3,$$

$$I_7(f) = \frac{|T|}{60} \left(3 \sum_{i=1}^{3} f(\mathbf{a}_T^{(i)}) + 8 \sum_{j=1}^{3} f(\mathbf{a}_{(j)}^T) + 27 f(\mathbf{a}_T) \right), \qquad n = 7.$$

For a description and analysis of symmetric formulae for triangles, see [Dun85], while we refer to [Kea86] and [Dun86] for their extension to tetrahedra and cubes.

The composite quadrature rules (9.58) and (9.59) are implemented in Programs 80 and 81 for the approximate evaluation of the integral of $f(x, y)$ over a single triangle $T \in \mathcal{T}_h$. To compute the integral over Ω it suffices to sum the result provided by the program over each triangle of $\mathcal{T}_h$. The coordinates of the vertices of the triangle T are stored in the arrays `xv` and `yv`.

Program 80 - midptr2d : Midpoint rule on a triangle

```
function inte=midptr2d(xv,yv,fun)
y12=yv(1)-yv(2); y23=yv(2)-yv(3); y31=yv(3)-yv(1);
areat=0.5*abs(xv(1)*y23+xv(2)*y31+xv(3)*y12);
x=sum(xv)/3; y=sum(yv)/3; inte=areat*eval(fun);
```

Program 81 - traptr2d : Trapezoidal rule on a triangle

```
function inte=traptr2d(xv,yv,fun)
y12=yv(1)-yv(2); y23=yv(2)-yv(3); y31=yv(3)-yv(1);
areat=0.5*abs(xv(1)*y23+xv(2)*y31+xv(3)*y12); inte=0;
for i=1:3, x=xv(i); y=yv(i); inte=inte+eval(fun); end;
inte=inte*areat/3;
```

9.9.3 Monte Carlo Methods for Numerical Integration

Numerical integration methods based on Monte Carlo techniques are a valid tool for approximating multidimensional integrals when the space dimension of $\mathbb{R}^n$ gets large. These methods differ from the approaches considered thus far, since the choice of quadrature nodes is done *statistically* according to the values attained by random variables having a known probability distribution.

The basic idea of the method is to interpret the integral as a *statistic mean value*

$$\int_\Omega f(\mathbf{x})d\mathbf{x} = |\Omega| \int_{\mathbb{R}^n} |\Omega|^{-1}\chi_\Omega(\mathbf{x})f(\mathbf{x})d\mathbf{x} = |\Omega|\mu(f),$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ and $|\Omega|$ denotes the n-dimensional volume of Ω, $\chi_\Omega(\mathbf{x})$ is the characteristic function of the set Ω, equal to 1 for $\mathbf{x} \in \Omega$ and to 0 elsewhere, while $\mu(f)$ is the mean value of the function $f(X)$, where X is a random variable with uniform probability density $|\Omega|^{-1}\chi_\Omega$ over $\mathbb{R}^n$.

We recall that the *random variable* $X \in \mathbb{R}^n$ (or, more properly, *random vector*) is an n-tuple of real numbers $X_1(\zeta), \ldots, X_n(\zeta)$ assigned to every outcome ζ of a random experiment (see [Pap87], Chapter 4).

Having fixed a vector $\mathbf{x} \in \mathbb{R}^n$, the probability $\mathcal{P}\{X \leq \mathbf{x}\}$ of the random event $\{X_1 \leq x_1, \ldots, X_n \leq x_n\}$ is given by

$$\mathcal{P}\{X \leq \mathbf{x}\} = \int_{-\infty}^{x_1} \ldots \int_{-\infty}^{x_n} f(X_1, \ldots, X_n)dX_1 \ldots dX_n$$

where $f(X) = f(X_1, \ldots, X_n)$ is the *probability density* of the random variable $X \in \mathbb{R}^n$, such that

$$f(X_1, \ldots, X_n) \geq 0, \qquad \int_{\mathbb{R}^n} f(X_1, \ldots, X_n)dX = 1.$$

The numerical computation of the mean value $\mu(f)$ is carried out by taking N independent samples $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^n$ with probability density $|\Omega|^{-1}\chi_\Omega$ and evaluating their *average*

$$\overline{f}_N = \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i) = I_N(f). \tag{9.60}$$

From a statistical standpoint, the samples $\mathbf{x}_1, \ldots, \mathbf{x}_N$ can be regarded as the realizations of a sequence of N random variables $\{X_1, \ldots, X_N\}$, mutually independent and each with probability density $|\Omega|^{-1}\chi_\Omega$.

For such a sequence the strong law of large numbers ensures with probability 1 the convergence of the average $I_N(f) = \left(\sum_{i=1}^N f(X_i)\right)/N$ to the mean value $\mu(f)$ as $N \to \infty$. In computational practice the sequence of samples $\mathbf{x}_1, \ldots, \mathbf{x}_N$ is deterministically produced by a random-number generator, giving rise to the so-called *pseudo-random integration formulae*.

The quadrature error $E_N(f) = \mu(f) - I_N(f)$ as a function of N can be characterized through the *variance*

$$\sigma(I_N(f)) = \sqrt{\mu\left(I_N(f) - \mu(f)\right)^2}.$$

Interpreting again f as a function of the random variable X, distributed with uniform probability density $|\Omega|^{-1}$ in $\Omega \subseteq \mathbb{R}^n$ and variance $\sigma(f)$, we have

$$\sigma(I_N(f)) = \frac{\sigma(f)}{\sqrt{N}}, \tag{9.61}$$

from which, as $N \to \infty$, a convergence rate of $\mathcal{O}(N^{-1/2})$ follows for the statistical estimate of the error $\sigma(I_N(f))$. Such convergence rate *does not* depend on the dimension n of the integration domain, and this is a most relevant feature of the Monte Carlo method. However, it is worth noting that the convergence rate is independent of the *regularity* of f; thus, unlike interpolatory quadratures, Monte Carlo methods *do not* yield more accurate results when dealing with smooth integrands.

The estimate (9.61) is extremely weak and in practice one does often obtain poorly accurate results. A more efficient implementation of Monte Carlo methods is based on composite approach or semi-analytical methods; an example of these techniques is provided in [NAG95], where a composite Monte Carlo method is employed for the computation of integrals over hypercubes in $\mathbb{R}^n$.

9.10 Applications

We consider in the next sections the computation of two integrals suggested by applications in geometry and the mechanics of rigid bodies.

9.10.1 Computation of an Ellipsoid Surface

Let E be the ellipsoid obtained by rotating the ellipse in Figure 9.6 around the x axis, where the radius ρ is described as a function of the axial coor-

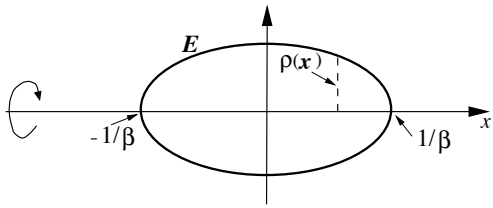

FIGURE 9.6. Section of the ellipsoid

dinate by the equation

$$\rho^2(x) = \alpha^2(1-\beta^2x^2), \qquad -\frac{1}{\beta} \le x \le \frac{1}{\beta},$$

α and β being given constants, assigned in such a way that $\alpha^2\beta^2 < 1$. We set the following values for the parameters: $\alpha^2 = (3-2\sqrt{2})/100$ and $\beta^2 = 100$. Letting $K^2 = \beta^2\sqrt{1-\alpha^2\beta^2}$, $f(x) = \sqrt{1-K^2x^2}$ and $\theta = \cos^{-1}(K/\beta)$, the computation of the surface of E requires evaluating the integral

$$I(f) = 4\pi\alpha \int_0^{1/\beta} f(x)dx = \frac{2\pi\alpha}{K}\left[(\pi/2-\theta) + \sin(2\theta)/2\right]. \tag{9.62}$$

Notice that $f'(1/\beta) = -100$; this prompts us to use a numerical adaptive formula able to provide a nonuniform distribution of quadrature nodes, with a possible refinement of these nodes around $x = 1/\beta$.
Table 9.12 summarizes the results obtained using the composite midpoint, trapezoidal and Cavalieri-Simpson rules (respectively denoted by (MP), (TR) and (CS)), which are compared with Romberg integration (RO) and with the adaptive Cavalieri-Simpson quadrature introduced in Section 9.7.2 and denoted by (AD).

In the table, m is the number of subintervals, while Err and `flops` denote the absolute quadrature error and the number of floating-point operations required by each algorithm, respectively. In the case of the AD method, we have run Program 77 taking `hmin`=10^{-5} and `tol`=10^{-8}, while for the Romberg method we have used Program 76 with `n`=9.
The results demonstrate the advantage of using the composite adaptive Cavalieri-Simpson formula, both in terms of computational efficiency and accuracy, as can be seen in the plots in Figure 9.7 which allow to check the successful working of the adaptivity procedure. In Figure 9.7 (left), we show, together with the graph of f, the nonuniform distribution of the quadrature nodes on the x axis, while in Figure 9.7 (right) we plot the logarithmic graph of the integration step density (piecewise constant) $\Delta_h(x)$, defined as the inverse of the value of the stepsize h on each active interval A (see Section 9.7.2).

Notice the high value of Δ_h at $x = 1/\beta$, where the derivative of the integrand function is maximum.

	(PM)	(TR)	(CS)	(RO)	(AD)
m	4000	5600	250		50
Err	$3.24e-10$	$3.30e-10$	$2.98e-10$	$3.58e-11$	$3.18e-10$
`flops`	20007	29013	2519	5772	3540

TABLE 9.12. Methods for approximating $I(f) = 4\pi\alpha \int_0^{1/\beta} \sqrt{1 - K^2 x^2} dx$, with $\alpha^2 = (3 - 2\sqrt{2})/100$, $\beta = 10$ and $K^2 = \sqrt{\beta^2(1 - \alpha^2\beta^2)}$

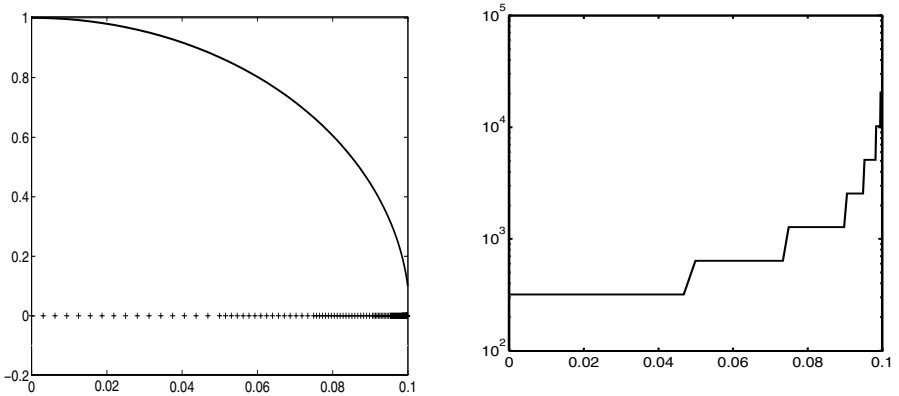

FIGURE 9.7. Distribution of quadrature nodes (left); integration stepsize density in the approximation of integral (9.62) (right)

9.10.2 Computation of the Wind Action on a Sailboat Mast

Let us consider the sailboat schematically drawn in Figure 9.8 (left) and subject to the action of the wind force. The mast, of length L, is denoted by the straight line AB, while one of the two shrouds (strings for the side stiffening of the mast) is represented by the straight line BO. Any infinitesimal element of the sail transmits to the corresponding element of length dx of the mast a force of magnitude equal to $f(x)dx$. The change of f along with the height x, measured from the point A (basis of the mast), is expressed by the following law

$$f(x) = \frac{\alpha x}{x + \beta} e^{-\gamma x},$$

where α, β and γ are given constants.

The resultant R of the force f is defined as

$$R = \int_0^L f(x)dx \equiv I(f), \tag{9.63}$$

and is applied at a point at distance equal to b (to be determined) from the basis of the mast.

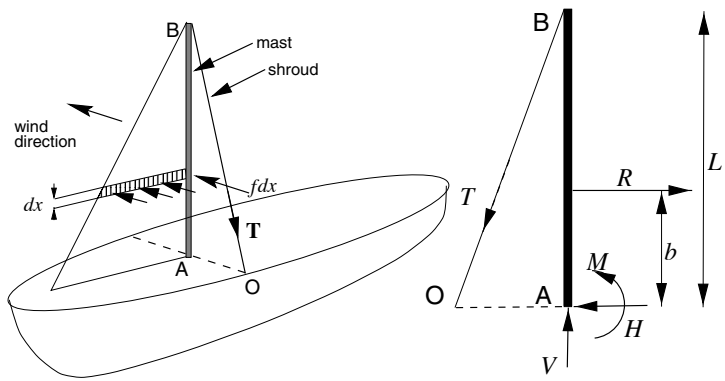

FIGURE 9.8. Schematic representation of a sailboat (left); forces acting on the mast (right)

Computing R and the distance b, given by $b = I(xf)/I(f)$, is crucial for the structural design of the mast and shroud sections. Indeed, once the values of R and b are known, it is possible to analyze the hyperstatic structure mast-shroud (using for instance the method of forces), thus allowing for the computation of the reactions V, H and M at the basis of the mast and the traction T that is transmitted by the shroud, and are drawn in Figure 9.8 (right). Then, the internal actions in the structure can be found, as well as the maximum stresses arising in the mast AB and in the shroud BO, from which, assuming that the safety verifications are satisfied, one can finally design the geometrical parameters of the sections of AB and BO.
For the approximate computation of R we have considered the composite midpoint, trapezoidal and Cavalieri-Simpson rules, denoted henceforth by (MP), (TR) and (CS), and, for a comparison, the adaptive Cavalieri-Simpson quadrature formula introduced in Section 9.7.2 and denoted by (AD). Since a closed-form expression for the integral (9.63) is not available, the composite rules have been applied taking $m_k = 2^k$ uniform partitions of $[0, L]$, with $k = 0, \ldots, 15$.

We have assumed in the numerical experiments $\alpha = 50$, $\beta = 5/3$ and $\gamma = 1/4$ and we have run Program 77 taking `tol`$=10^{-4}$ and `hmin`$=10^{-3}$. The sequence of integrals computed using the composite formulae has been stopped at $k = 12$ (corresponding to $m_k = 2^{12} = 4096$) since the remaining

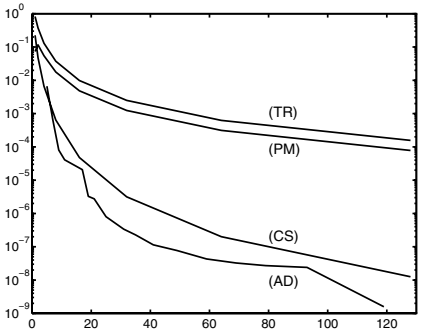

FIGURE 9.9. Relative errors in the approximate computation of the integral $\int_0^L (\alpha x e^{-\gamma x})/(x+\beta)dx$

elements, in the case of formula CS, differ among them only up to the last significant figure. Therefore, we have assumed as the exact value of $I(f)$ the outcome $I_{12}^{(CS)} = 100.0613683179612$ provided by formula CS.

We report in Figure 9.9 the logarithmic plots of the relative error $|I_{12}^{(CS)} - I_k|/I_{12}$, for $k = 0, \ldots, 7$, I_k being the generic element of the sequence for the three considered formulae. As a comparison, we also display the graph of the relative error in the case of formula AD, applied on a number of (nonuniform) partitions equivalent to that of the composite rules.
Notice how, for the same number of partitions, formula AD is more accurate, with a relative error of $2.06 \cdot 10^{-7}$ obtained using 37 (nonuniform) partitions of $[0, L]$. Methods PM and TR achieve a comparable accuracy employing 2048 and 4096 uniform subintervals, respectively, while formula CS requires about 64 partitions. The effectiveness of the adaptivity procedure is demonstrated by the plots in Figure 9.10, which show, together with the graph of f, the distribution of the quadrature nodes (left) and the function $\Delta_h(x)$ (right) that expresses the (piecewise constant) density of the integration stepsize h, defined as the inverse of the value of h over each active interval A (see Section 9.7.2).

Notice also the high value of Δ_h at $x = 0$, where the derivatives of f are maximum.

9.11 Exercises

1. Let $E_0(f)$ and $E_1(f)$ be the quadrature errors in (9.6) and (9.12). Prove that $|E_1(f)| \simeq 2|E_0(f)|$.

2. Check that the error estimates for the midpoint, trapezoidal and Cavalieri-Simpson formulae, given respectively by (9.6), (9.12) and (9.16), are special instances of (9.19) or (9.20). In particular, show that $M_0 = 2/3$, $K_1 =$

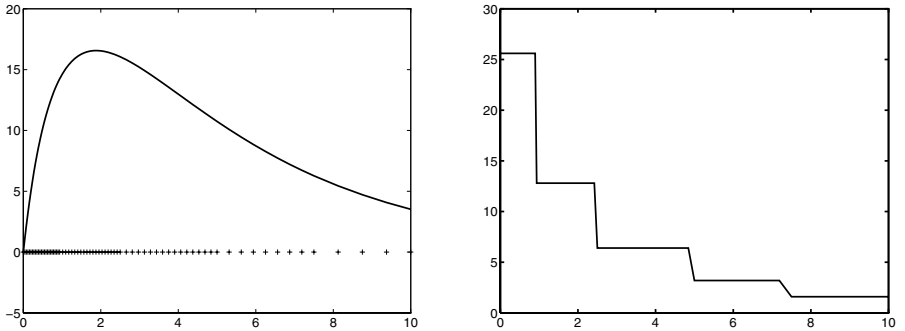

FIGURE 9.10. Distribution of quadrature nodes (left); integration step density in the approximation of the integral $\int_0^L (\alpha x e^{-\gamma x})/(x+\beta)dx$ (right)

$-1/6$ and $M_2 = -4/15$ and determine, using the definition, the degree of exactness r of each formula.

[*Hint*: find r such that $I_n(x^k) = \int_a^b x^k dx$, for $k = 0, \ldots, r$, and $I_n(x^j) \neq \int_a^b x^j dx$, for $j > r$.]

3. Let $I_n(f) = \sum_{k=0}^n \alpha_k f(x_k)$ be a Lagrange quadrature formula on $n+1$ nodes. Compute the degree of exactness r of the formulae:

   (a) $I_2(f) = (2/3)[2f(-1/2) - f(0) + 2f(1/2)]$,

   (b) $I_4(f) = (1/4)[f(-1) + 3f(-1/3) + 3f(1/3) + f(1)]$.

   Which is the order of infinitesimal p for (a) and (b)?

   [*Solution*: $r = 3$ and $p = 5$ for both $I_2(f)$ and $I_4(f)$.]

4. Compute $\mathrm{d}f[x_0, \ldots, x_n, x]/\mathrm{d}x$ by checking (9.22).

   [*Hint*: proceed by computing directly the derivative at x as an incremental ratio, in the case where only one node x_0 exists, then upgrade progressively the order of the divided difference.]

5. Let $I_w(f) = \int_0^1 w(x)f(x)dx$ with $w(x) = \sqrt{x}$, and consider the quadrature formula $Q(f) = af(x_1)$. Find a and x_1 in such a way that Q has maximum degree of exactness r.

   [*Solution*: $a = 2/3$, $x_1 = 3/5$ and $r = 1$.]

6. Let us consider the quadrature formula $Q(f) = \alpha_1 f(0) + \alpha_2 f(1) + \alpha_3 f'(0)$ for the approximation of $I(f) = \int_0^1 f(x)dx$, where $f \in C^1([0,1])$. Determine the coefficients α_j, for $j = 1, 2, 3$ in such a way that Q has degree of exactness $r = 2$.

   [*Solution*: $\alpha_1 = 2/3$, $\alpha_2 = 1/3$ and $\alpha_3 = 1/6$.]

7. Apply the midpoint, trapezoidal and Cavalieri-Simpson composite rules to approximate the integral

$$\int_{-1}^1 |x| e^x dx,$$

   and discuss their convergence as a function of the size H of the subintervals.

8. Consider the integral $I(f) = \int_0^1 e^x dx$ and estimate the minimum number m of subintervals that is needed for computing $I(f)$ up to an absolute error $\leq 5 \cdot 10^{-4}$ using the composite trapezoidal (TR) and Cavalieri-Simpson (CS) rules. Evaluate in both cases the absolute error Err that is actually made.

   [*Solution*: for TR, we have $m = 17$ and $Err = 4.95 \cdot 10^{-4}$, while for CS, $m = 2$ and $Err = 3.70 \cdot 10^{-5}$.]

9. Consider the corrected trapezoidal formula (9.30) and check that $|E_1^{corr}(f)| \simeq 4|E_2(f)|$, where $E_1^{corr}(f)$ and $E_2(f)$ are defined in (9.31) and (9.16), respectively.

10. Compute, with an error less than 10^{-4}, the following integrals:

    (a) $\int_0^\infty \sin(x)/(1+x^4)dx$;

    (b) $\int_0^\infty e^{-x}(1+x)^{-5}dx$;

    (c) $\int_{-\infty}^\infty \cos(x)e^{-x^2}dx$.

11. Use the reduction midpoint and trapezoidal formulae for computing the double integral $I(f) = \int_\Omega \dfrac{y}{(1+xy)} dxdy$ over the domain $\Omega = (0,1)^2$. Run Programs 78 and 79 with $M = 2^i$, for $i = 0, \ldots, 10$ and plot in log-scale the absolute error in the two cases as a function of M. Which method is the most accurate? How many functional evaluations are needed to get an (absolute) accuracy of the order of 10^{-6}?

    [*Solution*: the exact integral is $I(f) = \log(4) - 1$, and almost $200^2 = 40000$ functional evaluations are needed.]

10

Orthogonal Polynomials in Approximation Theory

Trigonometric polynomials, as well as other orthogonal polynomials like Legendre's and Chebyshev's, are widely employed in approximation theory.

This chapter addresses the most relevant properties of orthogonal polynomials, and introduces the transforms associated with them, in particular the discrete Fourier transform and the FFT, but also the Zeta and Wavelet transforms.

Application to interpolation, least-squares approximation, numerical differentiation and Gaussian integration are addressed.

10.1 Approximation of Functions by Generalized Fourier Series

Let $w = w(x)$ be a weight function on the interval $(-1, 1)$, i.e., a nonnegative integrable function in $(-1, 1)$. Let us denote by $\{p_k,\ k = 0, 1, \dots\}$ a system of algebraic polynomials, with p_k of degree equal to k for each k, mutually orthogonal on the interval $(-1, 1)$ with respect to w. This means that

$$\int_{-1}^{1} p_k(x) p_m(x) w(x) dx = 0 \qquad \text{if } k \neq m.$$

Set $(f,g)_w = \int_{-1}^{1} f(x)g(x)w(x)dx$ and $\|f\|_w = (f,f)_w^{1/2}$; $(\cdot,\cdot)_w$ and $\|\cdot\|_w$ are respectively the scalar product and the norm for the function space

$$\mathrm{L}^2_w = \mathrm{L}^2_w(-1,1) = \left\{ f : (-1,1) \to \mathbb{R},\ \int_{-1}^{1} f^2(x)w(x)dx < \infty \right\}. \quad (10.1)$$

For any function $f \in \mathrm{L}^2_w$ the series

$$Sf = \sum_{k=0}^{+\infty} \widehat{f}_k p_k, \qquad \text{with } \widehat{f}_k = \frac{(f,p_k)_w}{\|p_k\|_w^2},$$

is called the *generalized Fourier series of* f, and $\widehat{f}_k$ is the *k-th Fourier coefficient*. As is well-known, Sf converges *in average* (or *in the sense of* L^2_w) to f. This means that, letting for any integer n

$$f_n(x) = \sum_{k=0}^{n} \widehat{f}_k p_k(x) \quad (10.2)$$

($f_n \in \mathbb{P}_n$ is the truncation of order n of the generalized Fourier series of f), the following convergence result holds

$$\lim_{n \to +\infty} \|f - f_n\|_w = 0.$$

Thanks to Parseval's equality, we have

$$\|f\|_w^2 = \sum_{k=0}^{+\infty} \widehat{f}_k^2 \|p_k\|_w^2$$

and, for any n, $\|f - f_n\|_w^2 = \sum_{k=n+1}^{+\infty} \widehat{f}_k^2 \|p_k\|_w^2$ is the square of the remainder of the generalized Fourier series.

The polynomial $f_n \in \mathbb{P}_n$ satisfies the following minimization property

$$\|f - f_n\|_w = \min_{q \in \mathbb{P}_n} \|f - q\|_w. \quad (10.3)$$

Indeed, since $f - f_n = \sum_{k=n+1}^{+\infty} \widehat{f}_k p_k$, the property of orthogonality of polynomials $\{p_k\}$ implies $(f - f_n, q)_w = 0\ \forall q \in \mathbb{P}_n$. Then, the Cauchy-Schwarz inequality (8.29) yields

$$\begin{aligned} \|f - f_n\|_w^2 &= (f - f_n, f - f_n)_w = (f - f_n, f - q)_w + (f - f_n, q - f_n)_w \\ &= (f - f_n, f - q)_w \le \|f - f_n\|_w \|f - q\|_w, \quad \forall q \in \mathbb{P}_n, \end{aligned}$$

and (10.3) follows since q is arbitrary in $\mathbb{P}_n$. In such a case, we say that f_n is the orthogonal projection of f over $\mathbb{P}_n$ in the sense of L^2_w. It is therefore interesting to compute the coefficients $\widehat{f}_k$ of f_n. As will be seen in later

sections, this is usually done by suitably approximating the integrals that appear in the definition of $\widehat{f}_k$. By doing so, one gets the so-called *discrete coefficients* $\tilde{f}_k$ of f, and, as a consequence, the new polynomial

$$f_n^*(x) = \sum_{k=0}^{n} \tilde{f}_k p_k(x) \tag{10.4}$$

which is called the *discrete truncation of order* n of the Fourier series of f. Typically,

$$\tilde{f}_k = \frac{(f, p_k)_n}{\|p_k\|_n^2}, \tag{10.5}$$

where, for any pair of continuous functions f and g, $(f, g)_n$ is the approximation of the scalar product $(f, g)_w$ and $\|g\|_n = \sqrt{(g, g)_n}$ is the seminorm associated with $(\cdot, \cdot)_w$. In a manner analogous to what was done for f_n, it can be checked that

$$\|f - f_n^*\|_n = \min_{q \in \mathbb{P}_n} \|f - q\|_n \tag{10.6}$$

and we say that f_n^* is the approximation to f in $\mathbb{P}_n$ *in the least-squares sense* (the reason for using this name will be made clear later on).

We conclude this section by recalling that, for any family of monic orthogonal polynomials $\{p_k\}$, the following recursive three-term formula holds (for the proof, see for instance [Gau96])

$$\begin{cases} p_{k+1}(x) = (x - \alpha_k) p_k(x) - \beta_k p_{k-1}(x) & k \geq 0, \\ p_{-1}(x) = 0, \quad p_0(x) = 1, \end{cases} \tag{10.7}$$

where

$$\alpha_k = \frac{(x p_k, p_k)_w}{(p_k, p_k)_w}, \qquad \beta_{k+1} = \frac{(p_{k+1}, p_{k+1})_w}{(p_k, p_k)_w}, \qquad k \geq 0. \tag{10.8}$$

Since $p_{-1} = 0$, the coefficient β_0 is arbitrary and is chosen according to the particular family of orthogonal polynomials at hand. The recursive three-term relation is generally quite stable and can thus be conveniently employed in the numerical computation of orthogonal polynomials, as will be seen in Section 10.6.
In the forthcoming sections we introduce two relevant families of orthogonal polynomials.

10.1.1 The Chebyshev Polynomials

Consider the Chebyshev weight function $w(x) = (1-x^2)^{-1/2}$ on the interval $(-1, 1)$, and, according to (10.1), introduce the space of square-integrable

functions with respect to the weight w

$$\mathrm{L}^2_w(-1,1) = \left\{ f : (-1,1) \to \mathbb{R} : \int_{-1}^{1} f^2(x)(1-x^2)^{-1/2}dx < \infty \right\}.$$

A scalar product and a norm for this space are defined as

$$\begin{aligned} (f,g)_w &= \int_{-1}^{1} f(x)g(x)(1-x^2)^{-1/2}dx, \\ \|f\|_w &= \left\{ \int_{-1}^{1} f^2(x)(1-x^2)^{-1/2}dx \right\}^{1/2}. \end{aligned} \tag{10.9}$$

The Chebyshev polynomials are defined as follows

$$T_k(x) = \cos k\theta, \quad \theta = \arccos x, \quad k = 0, 1, 2, \ldots \tag{10.10}$$

They can be recursively generated by the following formula (a consequence of (10.7), see [DR75], pp. 25-26)

$$\begin{cases} T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x) & k = 1, 2, \ldots \\ \\ T_0(x) = 1, \qquad T_1(x) = x. \end{cases} \tag{10.11}$$

In particular, for any $k \geq 0$, we notice that $T_k \in \mathbb{P}_k$, i.e., $T_k(x)$ is an algebraic polynomial of degree k with respect to x. Using well-known trigonometric relations, we have

$$(T_k, T_n)_w = 0 \text{ if } k \neq n, \quad (T_n, T_n)_w = \begin{cases} c_0 = \pi & \text{if } n = 0, \\ \\ c_n = \pi/2 & \text{if } n \neq 0, \end{cases}$$

which expresses the orthogonality of the Chebyshev polynomials with respect to the scalar product $(\cdot,\cdot)_w$. Therefore, the Chebyshev series of a function $f \in \mathrm{L}^2_w$ takes the form

$$Cf = \sum_{k=0}^{\infty} \widehat{f}_k T_k, \quad \text{with} \quad \widehat{f}_k = \frac{1}{c_k} \int_{-1}^{1} f(x)T_k(x)(1-x^2)^{-1/2}dx.$$

Notice that $\|T_n\|_\infty = 1$ for every n and the following *minimax* property holds

$$\|2^{1-n}T_n\|_\infty \leq \min_{p \in \mathbb{P}^1_n} \|p\|_\infty,$$

where $\mathbb{P}^1_n = \{p(x) = \sum_{k=0}^{n} a_k x^k,\ a_n = 1\}$ denotes the subset of polynomials of degree n with leading coefficient equal to 1.

10.1.2 The Legendre Polynomials

The Legendre polynomials are orthogonal polynomials over the interval $(-1,1)$ with respect to the weight function $w(x)=1$. For these polynomials, L^2_w is the usual $\mathrm{L}^2(-1,1)$ space introduced in (8.25), while $(\cdot,\cdot)_w$ and $\|\cdot\|_w$ coincide with the scalar product and norm in $\mathrm{L}^2(-1,1)$, respectively given by

$$(f,g)=\int_{-1}^{1} f(x)g(x)\,dx, \quad \|f\|_{\mathrm{L}^2(-1,1)}=\left(\int_{-1}^{1} f^2(x)\,dx\right)^{\frac{1}{2}}.$$

The Legendre polynomials are defined as

$$L_k(x)=\frac{1}{2^k}\sum_{l=0}^{[k/2]}(-1)^l\binom{k}{l}\binom{2k-2l}{k}x^{k-2l} \qquad k=0,1,\ldots \tag{10.12}$$

where $[k/2]$ is the integer part of $k/2$, or, recursively, through the three-term relation

$$\begin{cases} L_{k+1}(x)=\dfrac{2k+1}{k+1}xL_k(x)-\dfrac{k}{k+1}L_{k-1}(x) \qquad k=1,2\ldots \\ \\ L_0(x)=1, \qquad L_1(x)=x. \end{cases}$$

For every $k=0,1\ldots$, $L_k\in\mathbb{P}_k$ and $(L_k,L_m)=\delta_{km}(k+1/2)^{-1}$ for $k,m=0,1,2,\ldots$. For any function $f\in\mathrm{L}^2(-1,1)$, its Legendre series takes the following form

$$Lf=\sum_{k=0}^{\infty}\widehat{f}_kL_k, \quad \text{with} \quad \widehat{f}_k=\left(k+\frac{1}{2}\right)^{-1}\int_{-1}^{1}f(x)L_k(x)dx.$$

Remark 10.1 (The Jacobi polynomials) The polynomials previously introduced belong to the wider family of Jacobi polynomials $\{J_k^{\alpha\beta},k=0,\ldots,n\}$, that are orthogonal with respect to the weight $w(x)=(1-x)^\alpha(1+x)^\beta$, for $\alpha,\beta>-1$. Indeed, setting $\alpha=\beta=0$ we recover the Legendre polynomials, while choosing $\alpha=\beta=-1/2$ gives the Chebyshev polynomials. ■

10.2 Gaussian Integration and Interpolation

Orthogonal polynomials play a crucial role in devising quadrature formulae with maximal degrees of exactness. Let $x_0,\ldots,x_n$ be $n+1$ given distinct points in the interval $[-1,1]$. For the approximation of the weighted integral

$I_w(f) = \int_{-1}^{1} f(x)w(x)dx$, being $f \in C^0([-1,1])$, we consider quadrature rules of the type

$$I_{n,w}(f) = \sum_{i=0}^{n} \alpha_i f(x_i) \tag{10.13}$$

where α_i are coefficients to be suitably determined. Obviously, both nodes and weights depend on n, however this dependence will be understood. Denoting by

$$E_{n,w}(f) = I_w(f) - I_{n,w}(f)$$

the error between the exact integral and its approximation (10.13), if $E_{n,w}(p) = 0$ for any $p \in \mathbb{P}_r$ (for a suitable $r \geq 0$) we shall say that formula (10.13) has degree of exactness r with respect to the weight w. This definition generalizes the one given for ordinary integration with weight $w = 1$.

Clearly, we can get a degree of exactness equal to (at least) n taking

$$I_{n,w}(f) = \int_{-1}^{1} \Pi_n f(x)w(x)dx$$

where $\Pi_n f \in \mathbb{P}_n$ is the Lagrange interpolating polynomial of the function f at the nodes $\{x_i, i = 0, \ldots, n\}$, given by (8.4). Therefore, (10.13) has degree of exactness at least equal to n taking

$$\alpha_i = \int_{-1}^{1} l_i(x)w(x)dx, \qquad i = 0, \ldots, n, \tag{10.14}$$

where $l_i \in \mathbb{P}_n$ is the i-th characteristic Lagrange polynomial such that $l_i(x_j) = \delta_{ij}$, for $i, j = 0, \ldots, n$.

The question that arises is whether suitable choices of the nodes exist such that the degree of exactness is greater than n, say, equal to $r = n + m$ for some $m > 0$. The answer to this question is furnished by the following theorem, due to Jacobi [Jac26].

Theorem 10.1 *For a given $m > 0$, the quadrature formula* (10.13) *has degree of exactness $n + m$ iff it is of interpolatory type and the nodal polynomial ω_{n+1}* (8.6) *associated with the nodes $\{x_i\}$ is such that*

$$\int_{-1}^{1} \omega_{n+1}(x)p(x)w(x)dx = 0, \qquad \forall p \in \mathbb{P}_{m-1}. \tag{10.15}$$

Proof. Let us prove that these conditions are sufficient. If $f \in \mathbb{P}_{n+m}$ then there exist a quotient $\pi_{m-1} \in \mathbb{P}_{m-1}$ and a remainder $q_n \in \mathbb{P}_n$, such that $f =$

$\omega_{n+1}\pi_{m-1} + q_n$. Since the degree of exactness of an interpolatory formula with $n+1$ nodes is equal to n (at least), we get

$$\sum_{i=0}^{n} \alpha_i q_n(x_i) = \int_{-1}^{1} q_n(x)w(x)dx = \int_{-1}^{1} f(x)w(x)dx - \int_{-1}^{1} \omega_{n+1}(x)\pi_{m-1}(x)w(x)dx.$$

As a consequence of (10.15), the last integral is null, thus

$$\int_{-1}^{1} f(x)w(x)dx = \sum_{i=0}^{n} \alpha_i q_n(x_i) = \sum_{i=0}^{n} \alpha_i f(x_i).$$

Since f is arbitrary, we conclude that $E_{n,w}(f) = 0$ for any $f \in \mathbb{P}_{n+m}$. Proving that the conditions are also necessary is an exercise left to the reader. ◇

Corollary 10.1 *The maximum degree of exactness of the quadrature formula* (10.13) *is* $2n+1$.

Proof. If this would not be true, one could take $m \geq n+2$ in the previous theorem. This, in turn, would allow us to choose $p = \omega_{n+1}$ in (10.15) and come to the conclusion that ω_{n+1} is identically zero, which is absurd. ◇

Setting $m = n+1$ (the maximum admissible value), from (10.15) we get that the nodal polynomial ω_{n+1} satisfies the relation

$$\int_{-1}^{1} \omega_{n+1}(x)p(x)w(x)dx = 0, \qquad \forall p \in \mathbb{P}_n.$$

Since ω_{n+1} is a polynomial of degree $n+1$ orthogonal to all the polynomials of lower degree, we conclude that ω_{n+1} is the only monic polynomial multiple of p_{n+1} (recall that $\{p_k\}$ is the system of orthogonal polynomials introduced in Section 10.1). In particular, its roots $\{x_j\}$ coincide with those of p_{n+1}, that is

$$p_{n+1}(x_j) = 0, \qquad j = 0, \ldots, n. \tag{10.16}$$

The abscissae $\{x_j\}$ are the *Gauss nodes* associated with the weight function $w(x)$. We can thus conclude that the quadrature formula (10.13) with coefficients and nodes given by (10.14) and (10.16), respectively, has degree of exactness $2n+1$, the maximum value that can be achieved using interpolatory quadrature formulae with $n+1$ nodes, and is called the *Gauss quadrature formula*.

Its weights are all positive and the nodes are *internal* to the interval $(-1, 1)$ (see, for instance, [CHQZ88], p. 56). However, it is often useful to also include the end points of the interval among the quadrature nodes. By

doing so, the Gauss formula with the highest degree of exactness is the one that employs as nodes the $n+1$ roots of the polynomial

$$\overline{\omega}_{n+1}(x) = p_{n+1}(x) + ap_n(x) + bp_{n-1}(x), \tag{10.17}$$

where the constants a and b are selected in such a way that $\overline{\omega}_{n+1}(-1) = \overline{\omega}_{n+1}(1) = 0$.

Denoting these roots by $\overline{x}_0 = -1, \overline{x}_1, \ldots, \overline{x}_n = 1$, the coefficients $\{\overline{\alpha}_i, i = 0, \ldots, n\}$ can then be obtained from the usual formulae (10.14), that is

$$\overline{\alpha}_i = \int_{-1}^{1} \bar{l}_i(x)w(x)dx, \qquad i = 0, \ldots, n,$$

where $\bar{l}_i \in \mathbb{P}_n$ is the i-th characteristic Lagrange polynomial such that $\bar{l}_i(\overline{x}_j) = \delta_{ij}$, for $i, j = 0, \ldots, n$. The quadrature formula

$$I_{n,w}^{GL}(f) = \sum_{i=0}^{n} \overline{\alpha}_i f(\overline{x}_i) \tag{10.18}$$

is called the *Gauss-Lobatto formula* with $n+1$ nodes, and has degree of exactness $2n-1$. Indeed, for any $f \in \mathbb{P}_{2n-1}$, there exist a polynomial $\pi_{n-2} \in \mathbb{P}_{n-2}$ and a remainder $q_n \in \mathbb{P}_n$ such that $f = \overline{\omega}_{n+1}\pi_{n-2} + q_n$.

The quadrature formula (10.18) has degree of exactness at least equal to n (being interpolatory with $n+1$ distinct nodes), thus we get

$$\sum_{j=0}^{n} \overline{\alpha}_j q_n(x_j) = \int_{-1}^{1} q_n(x)w(x)dx = \int_{-1}^{1} f(x)w(x)dx - \int_{-1}^{1} \overline{\omega}_{n+1}(x)\pi_{n-2}(x)w(x)dx.$$

From (10.17) we conclude that $\bar{\omega}_{n+1}$ is orthogonal to all the polynomials of degree $\leq n-2$, so that the last integral is null. Moreover, since $f(\overline{x}_j) = q_n(\overline{x}_j)$ for $j = 0, \ldots, n$, we conclude that

$$\int_{-1}^{1} f(x)w(x)dx = \sum_{i=0}^{n} \overline{\alpha}_i f(\overline{x}_i), \qquad \forall f \in \mathbb{P}_{2n-1}.$$

Denoting by $\Pi_{n,w}^{GL} f$ the polynomial of degree n that interpolates f at the nodes $\{\overline{x}_j, j = 0, \ldots, n\}$, we get

$$\Pi_{n,w}^{GL} f(x) = \sum_{i=0}^{n} f(\overline{x}_i)\bar{l}_i(x) \tag{10.19}$$

and thus $I_{n,w}^{GL}(f) = \int_{-1}^{1} \Pi_{n,w}^{GL} f(x)w(x)dx$.

Remark 10.2 In the special case where the Gauss-Lobatto quadrature is considered with respect to the Jacobi weight $w(x) = (1-x)^\alpha(1-x)^\beta$, with $\alpha, \beta > -1$, the internal nodes $\overline{x}_1, \ldots, \overline{x}_{n-1}$ can be identified as the roots of the polynomial $(J_n^{(\alpha,\beta)})'$, that is, the extremants of the n-th Jacobi polynomial $J_n^{(\alpha,\beta)}$ (see [CHQZ88], pp. 57-58). ■

The following convergence result holds for Gaussian integration (see [Atk89], Chapter 5)

$$\lim_{n\to+\infty} \left| \int_{-1}^{1} f(x)w(x)dx - \sum_{j=0}^{n} \alpha_j f(x_j) \right| = 0, \qquad \forall f \in C^0([-1,1]).$$

A similar result also holds for Gauss-Lobatto integration. If the integrand function is not only continuous, but also differentiable up to the order $p \geq 1$, we shall see that Gaussian integration converges with an order of infinitesimal with respect to $1/n$ that is larger when p is greater. In the forthcoming sections, the previous results will be specified in the cases of the Chebyshev and Legendre polynomials.

Remark 10.3 (Integration over an arbitrary interval) A quadrature formula with nodes ξ_j and coefficients β_j, $j = 0, \ldots, n$ over the interval $[-1,1]$ can be mapped on any interval $[a,b]$. Indeed, let $\varphi : [-1,1] \to [a,b]$ be the affine map $x = \varphi(\xi) = \frac{a+b}{2}\xi + \frac{b-a}{2}$. Then

$$\int_a^b f(x)dx = \frac{a+b}{2} \int_{-1}^{1} (f \circ \varphi)(\xi)d\xi.$$

Therefore, we can employ on the interval $[a,b]$ the quadrature formula with nodes $x_j = \varphi(\xi_j)$ and weights $\alpha_j = \frac{a+b}{2}\beta_j$. Notice that this formula maintains on the interval $[a,b]$ the same degree of exactness of the generating formula over $[-1,1]$. Indeed, assuming that

$$\int_{-1}^{1} p(\xi)d\xi = \sum_{j=0}^{n} p(\xi_j)\beta_j$$

for any polynomial p of degree r over $[-1,1]$ (for a suitable integer r), for any polynomial q of the same degree on $[a,b]$ we get

$$\sum_{j=0}^{n} q(x_j)\alpha_j = \frac{a+b}{2} \sum_{j=0}^{n} (q \circ \varphi)(\xi_j)\beta_j = \frac{a+b}{2} \int_{-1}^{1} (q \circ \varphi)(\xi)d\xi = \int_a^b q(x)dx,$$

having recalled that $(q \circ \varphi)(\xi)$ is a polynomial of degree r on $[-1,1]$. ■

10.3 Chebyshev Integration and Interpolation

If Gaussian quadratures are considered with respect to the Chebyshev weight $w(x) = (1-x^2)^{-1/2}$, Gauss nodes and coefficients are given by

$$x_j = -\cos\frac{(2j+1)\pi}{2(n+1)}, \quad \alpha_j = \frac{\pi}{n+1}, \quad 0 \le j \le n, \tag{10.20}$$

while Gauss-Lobatto nodes and weights are

$$\overline{x}_j = -\cos\frac{\pi j}{n}, \quad \overline{\alpha}_j = \frac{\pi}{d_j n}, \quad 0 \le j \le n, \quad n \ge 1, \tag{10.21}$$

where $d_0 = d_n = 2$ and $d_j = 1$ for $j = 1, \ldots, n-1$. Notice that the Gauss nodes (10.20) are, for a fixed $n \ge 0$, the zeros of the Chebyshev polynomial $T_{n+1} \in \mathbb{P}_{n+1}$, while, for $n \ge 1$, the internal nodes $\{\bar{x}_j,\ j = 1, \ldots, n-1\}$ are the zeros of T_n', as anticipated in Remark 10.2.

Denoting by $\Pi_{n,w}^{GL} f$ the polynomial of degree $n+1$ that interpolates f at the nodes (10.21), it can be shown that the interpolation error can be bounded as

$$\|f - \Pi_{n,w}^{GL} f\|_w \le C n^{-s} \|f\|_{s,w}, \qquad \text{for } s \ge 1, \tag{10.22}$$

where $\|\cdot\|_w$ is the norm in L_w^2 defined in (10.9), provided that for some $s \ge 1$ the function f has derivatives $f^{(k)}$ of order $k = 0, \ldots, s$ in L_w^2. In such a case

$$\|f\|_{s,w} = \left(\sum_{k=0}^{s} \|f^{(k)}\|_w^2 \right)^{\frac{1}{2}}. \tag{10.23}$$

Here and in the following, C is a constant independent of n that can assume different values at different places. In particular, for any continuous function f the following pointwise error estimate can be derived (see Exercise 3)

$$\|f(x) - \Pi_{n,w}^{GL} f(x)\|_\infty \le C n^{1/2-s} \|f\|_{s,w}. \tag{10.24}$$

Thus, $\Pi_{n,w}^{GL} f$ converges pointwise to f as $n \to \infty$, for any $f \in C^1([-1,1])$. The same kind of results (10.22) and (10.24) hold if $\Pi_{n,w}^{GL} f$ is replaced with the polynomial $\Pi_n^G f$ of degree n that interpolates f at the $n+1$ Gauss nodes x_j in (10.20). (For the proof of these results see, for instance, [CHQZ88], p. 298, or [QV94], p. 112). We have also the following result (see [Riv74], p.13)

$$\|f - \Pi_n^G f\|_\infty \le (1 + \Lambda_n) E_n^*(f), \quad \text{with } \Lambda_n \le \frac{2}{\pi}\log(n+1) + 1, \tag{10.25}$$

where $\forall n,\ E_n^*(f) = \inf_{p \in \mathbb{P}_n} \|f - p\|_\infty$ is the best approximation error for f in $\mathbb{P}_n$ and Λ_n is the Lebesgue constant associated with the Chebyshev

nodes (10.20). As far as the numerical integration error is concerned, let us consider, for instance, the Gauss-Lobatto quadrature rule (10.18) with nodes and weights given in (10.21). First of all, notice that

$$\int_{-1}^{1} f(x)(1-x^2)^{-1/2}dx = \lim_{n\to\infty} I_{n,w}^{GL}(f)$$

for any function f whose left integral is finite (see [Sze67], p. 342). If, moreover, $\|f\|_{s,w}$ is finite for some $s \geq 1$, we have

$$\left| \int_{-1}^{1} f(x)(1-x^2)^{-1/2}dx - I_{n,w}^{GL}(f) \right| \leq Cn^{-s}\|f\|_{s,w}. \tag{10.26}$$

This result follows from the more general one

$$|(f, v_n)_w - (f, v_n)_n| \leq Cn^{-s}\|f\|_{s,w}\|v_n\|_w, \qquad \forall v_n \in \mathbb{P}_n, \tag{10.27}$$

where the so-called *discrete scalar product* has been introduced

$$(f, g)_n = \sum_{j=0}^{n} \overline{\alpha}_j f(\overline{x}_j) g(\overline{x}_j) = I_{n,w}^{GL}(fg). \tag{10.28}$$

Actually, (10.26) follows from (10.27) setting $v_n \equiv 1$ and noticing that $\|v_n\|_w = \left(\int_{-1}^{1}(1-x^2)^{-1/2}dx\right)^{1/2} = \sqrt{\pi}$. Thanks to (10.26) we can thus conclude that the (Chebyshev) Gauss-Lobatto formula has degree of exactness $2n-1$ and order of accuracy (with respect to n^{-1}) equal to s, provided that $\|f\|_{s,w} < \infty$. Therefore, the order of accuracy is only limited by the regularity threshold s of the integrand function. Completely similar considerations can be drawn for (Chebyshev) Gauss formulae with $n+1$ nodes.

Let us finally determine the coefficients $\tilde{f}_k$, $k = 0, \ldots, n$, of the interpolating polynomial $\Pi_{n,w}^{GL} f$ at the $n+1$ Gauss-Lobatto nodes in the expansion with respect to the Chebyshev polynomials (10.10)

$$\Pi_{n,w}^{GL} f(x) = \sum_{k=0}^{n} \tilde{f}_k T_k(x). \tag{10.29}$$

Notice that $\Pi_{n,w}^{GL} f$ coincides with the discrete truncation of the Chebyshev series f_n^* defined in (10.4). Enforcing the equality $\Pi_{n,w}^{GL} f(\overline{x}_j) = f(\overline{x}_j)$, $j = 0, \ldots, n$, we find

$$f(\overline{x}_j) = \sum_{k=0}^{n} \cos\left(\frac{kj\pi}{n}\right) \tilde{f}_k, \qquad j = 0, \ldots, n. \tag{10.30}$$

Recalling the exactness of the Gauss-Lobatto quadrature, it can be checked that (see Exercise 2)

$$\tilde{f}_k = \frac{2}{nd_k}\sum_{j=0}^{n}\frac{1}{d_j}\cos\left(\frac{kj\pi}{n}\right)f(\overline{x}_j), \qquad k=0,\dots,n. \tag{10.31}$$

Relation (10.31) yields the discrete coefficients $\{\tilde{f}_k, k=0,\dots,n\}$ in terms of the nodal values $\{f(\overline{x}_j), j=0,\dots,n\}$. For this reason it is called the *Chebyshev discrete transform* (CDT) and, thanks to its trigonometric structure, it can be efficiently computed using the FFT algorithm (Fast Fourier transform) with a number of floating-point operations of the order of $n\log_2 n$ (see Section 10.9.2). Of course, (10.30) is the *inverse* of the CDT, and can be computed using the FFT.

10.4 Legendre Integration and Interpolation

As previously noticed, the Legendre weight is $w(x)\equiv 1$. For $n\geq 0$, the Gauss nodes and the related coefficients are given by

$$x_j \text{ zeros of } L_{n+1}(x),\ \alpha_j = \frac{2}{(1-x_j^2)[L'_{n+1}(x_j)]^2},\ j=0,\dots,n, \tag{10.32}$$

while the Gauss-Lobatto ones are, for $n\geq 1$

$$\overline{x}_0=-1,\ \overline{x}_n=1,\quad \overline{x}_j \text{ zeros of } L'_n(x),\quad j=1,\dots,n-1 \tag{10.33}$$

$$\overline{\alpha}_j = \frac{2}{n(n+1)}\frac{1}{[L_n(x_j)]^2}, \qquad j=0,\dots,n \tag{10.34}$$

where L_n is the n-th Legendre polynomial defined in (10.12). It can be checked that, for a suitable constant C independent of n,

$$\frac{2}{n(n+1)} \leq \overline{\alpha}_j \leq \frac{C}{n}, \qquad \forall j=0,\dots,n$$

(see [BM92], p. 76). Then, letting $\Pi_n^{GL}f$ be the polynomial of degree n that interpolates f at the $n+1$ nodes $\overline{x}_j$ given by (10.33), it can be proved that it fulfills the same error estimates as those reported in (10.22) and (10.24) in the case of the corresponding Chebyshev polynomial.

Of course, the norm $\|\cdot\|_w$ must here be replaced by the norm $\|\cdot\|_{\mathrm{L}^2(-1,1)}$, while $\|f\|_{s,w}$ becomes

$$\|f\|_s = \left(\sum_{k=0}^{s}\|f^{(k)}\|^2_{\mathrm{L}^2(-1,1)}\right)^{\frac{1}{2}}. \tag{10.35}$$

The same kinds of results are ensured if $\Pi_n^{GL} f$ is replaced by the polynomial of degree n that interpolates f at the $n+1$ nodes x_j given by (10.32).

Referring to the discrete scalar product defined in (10.28), but taking now the nodes and coefficients given by (10.33) and (10.34), we see that $(\cdot,\cdot)_n$ is an approximation of the usual scalar product $(\cdot,\cdot)$ of $\mathrm{L}^2(-1,1)$. Actually, the equivalent relation to (10.27) now reads

$$|(f,v_n)-(f,v_n)_n| \le Cn^{-s}\|f\|_s\|v_n\|_{\mathrm{L}^2(-1,1)}, \qquad \forall v_n \in \mathbb{P}_n \tag{10.36}$$

and holds for any $s \ge 1$ such that $\|f\|_s < \infty$. In particular, setting $v_n \equiv 1$, we get $\|v_n\| = \sqrt{2}$, and from (10.36) it follows that

$$\left| \int_{-1}^{1} f(x)dx - I_n^{GL}(f) \right| \le Cn^{-s}\|f\|_s \tag{10.37}$$

which demonstrates a convergence of the Gauss-Legendre-Lobatto quadrature formula to the exact integral of f with order of accuracy s with respect to n^{-1} provided that $\|f\|_s < \infty$. A similar result holds for the Gauss-Legendre quadrature formulae.

Example 10.1 Consider the approximate evaluation of the integral of $f(x) = |x|^{\alpha+\frac{3}{5}}$ over $[-1,1]$ for $\alpha = 0,1,2$. Notice that f has "piecewise" derivatives up to order $s = s(\alpha) = \alpha + 1$ in $\mathrm{L}^2(-1,1)$. Figure 10.1 shows the behavior of the error as a function of n for the Gauss-Legendre quadrature formula. According to (10.37), the convergence rate of the formula increases by one when α increases by one. •

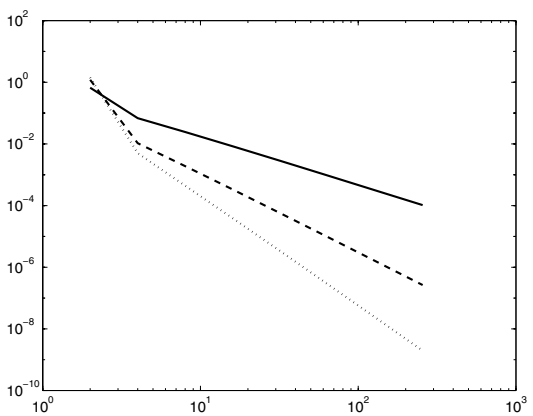

FIGURE 10.1. The quadrature error in logarithmic scale as a function of n in the case of a function with the first s derivatives in $\mathrm{L}^2(-1,1)$ for $s = 1$ (solid line), $s = 2$ (dashed line), $s = 3$ (dotted line)

The interpolating polynomial at the nodes (10.33) is given by

$$\Pi_n^{GL} f(x) = \sum_{k=0}^{n} \tilde{f}_k L_k(x). \tag{10.38}$$

Notice that also in this case $\Pi_n^{GL} f$ coincides with the discrete truncation of the Legendre series f_n^* defined in (10.4). Proceeding as in the previous section, we get

$$f(\overline{x}_j) = \sum_{k=0}^{n} \tilde{f}_k L_k(\overline{x}_j), \qquad j = 0, \ldots, n, \tag{10.39}$$

and also

$$\tilde{f}_k = \begin{cases} \dfrac{2k+1}{n(n+1)} \displaystyle\sum_{j=0}^{n} L_k(\overline{x}_j) \frac{1}{L_n^2(\overline{x}_j)} f(\overline{x}_j), & k = 0, \ldots, n-1, \\ \dfrac{1}{n+1} \displaystyle\sum_{j=0}^{n} \frac{1}{L_n(\overline{x}_j)} f(\overline{x}_j), & k = n \end{cases} \tag{10.40}$$

(see Exercise 6). Formulae (10.40) and (10.39) provide, respectively, the *discrete Legendre transform* (DLT) and its inverse.

10.5 Gaussian Integration over Unbounded Intervals

We consider integration on both half and on the whole of real axis. In both cases we use interpolatory Gaussian formulae whose nodes are the zeros of Laguerre and Hermite orthogonal polynomials, respectively.

The Laguerre polynomials. These are algebraic polynomials, orthogonal on the interval $[0, +\infty)$ with respect to the weight function $w(x) = e^{-x}$. They are defined by

$$\mathcal{L}_n(x) = e^x \frac{d^n}{dx^n}(e^{-x} x^n), \qquad n \geq 0,$$

and satisfy the following three-term recursive relation

$$\begin{cases} \mathcal{L}_{n+1}(x) = (2n + 1 - x)\mathcal{L}_n(x) - n^2 \mathcal{L}_{n-1}(x) & n \geq 0, \\ \mathcal{L}_{-1} = 0, \qquad \mathcal{L}_0 = 1. \end{cases}$$

For any function f, define $\varphi(x) = f(x)e^x$. Then, $I(f) = \int_0^\infty f(x)dx = \int_0^\infty e^{-x}\varphi(x)dx$, so that it suffices to apply to this last integral the Gauss-Laguerre quadratures, to get, for $n \geq 1$ and $f \in C^{2n}([0, +\infty))$

$$I(f) = \sum_{k=1}^{n} \alpha_k \varphi(x_k) + \frac{(n!)^2}{(2n)!} \varphi^{(2n)}(\xi), \qquad 0 < \xi < +\infty, \tag{10.41}$$

where the nodes x_k, for $k = 1, \ldots, n$, are the zeros of $\mathcal{L}_n$ and the weights are $\alpha_k = (n!)^2 x_k / [\mathcal{L}_{n+1}(x_k)]^2$. From (10.41), one concludes that Gauss-Laguerre formulae are exact for functions f of the type φe^{-x}, where $\varphi \in \mathbb{P}_{2n-1}$. In a generalized sense, we can then state that they have optimal degrees of exactness equal to $2n-1$.

Example 10.2 Using a Gauss-Laguerre quadrature formula with $n = 12$ to compute the integral in Example 9.12 we obtain the value 0.5997 with an absolute error with respect to exact integration equal to $2.96 \cdot 10^{-4}$. For the sake of comparison, the composite trapezoidal formula would require 277 nodes to obtain the same accuracy. •

The Hermite polynomials. These are orthogonal polynomials on the real line with respect to the weight function $w(x) = e^{-x^2}$. They are defined by

$$\mathcal{H}_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n}(e^{-x^2}), \qquad n \geq 0.$$

Hermite polynomials can be recursively generated as

$$\begin{cases} \mathcal{H}_{n+1}(x) = 2x\mathcal{H}_n(x) - 2n\mathcal{H}_{n-1}(x) & n \geq 0, \\ \mathcal{H}_{-1} = 0, \qquad \mathcal{H}_0 = 1. \end{cases}$$

As in the previous case, letting $\varphi(x) = f(x)e^{x^2}$, we have $I(f) = \int_{-\infty}^{\infty} f(x)dx = \int_{-\infty}^{\infty} e^{-x^2}\varphi(x)dx$. Applying to this last integral the Gauss-Hermite quadratures we obtain, for $n \geq 1$ and $f \in C^{2n}(\mathbb{R})$

$$I(f) = \int_{-\infty}^{\infty} e^{-x^2}\varphi(x)dx = \sum_{k=1}^{n} \alpha_k \varphi(x_k) + \frac{(n!)\sqrt{\pi}}{2^n(2n)!}\varphi^{(2n)}(\xi), \qquad \xi \in \mathbb{R}, \tag{10.42}$$

where the nodes x_k, for $k = 1, \ldots, n$, are the zeros of $\mathcal{H}_n$ and the weights are $\alpha_k = 2^{n+1}n!\sqrt{\pi}/[\mathcal{H}_{n+1}(x_k)]^2$. As for Gauss-Laguerre quadratures, the Gauss-Hermite rules also are exact for functions f of the form φe^{-x^2}, where $\varphi \in \mathbb{P}_{2n-1}$; therefore, they have optimal degrees of exactness equal to $2n-1$.

More details on the subject can be found in [DR75], pp. 173-174.

10.6 Programs for the Implementation of Gaussian Quadratures

Programs 82, 83 and 84 compute the coefficients $\{\alpha_k\}$ and $\{\beta_k\}$, introduced in (10.8), in the cases of the Legendre, Laguerre and Hermite polynomials. These programs are then called by Program 85 for the computation of nodes

and weights (10.32), in the case of the Gauss-Legendre formulae, and by Programs 86, 87 for computing nodes and weights in the Gauss-Laguerre and Gauss-Hermite quadrature rules (10.41) and (10.42). All the codings reported in this section are excerpts from the library `ORTHPOL` [Gau94].

Program 82 - coeflege : Coefficients of Legendre polynomials

```
function [a, b] = coeflege(n)
if (n <= 1), disp(' n must be > 1 '); return; end
for k=1:n, a(k)=0; b(k)=0; end; b(1)=2;
for k=2:n, b(k)=1/(4-1/(k-1)^2); end
```

Program 83 - coeflagu : Coefficients of Laguerre polynomials

```
function [a, b] = coeflagu(n)
if (n <= 1), disp(' n must be > 1 '); return; end
a=zeros(n,1); b=zeros(n,1); a(1)=1; b(1)=1;
for k=2:n, a(k)=2*(k-1)+1; b(k)=(k-1)^2; end
```

Program 84 - coefherm : Coefficients of Hermite polynomials

```
function [a, b] = coefherm(n)
if (n <= 1), disp(' n must be > 1 '); return; end
a=zeros(n,1); b=zeros(n,1); b(1)=sqrt(4.*atan(1.));
for k=2:n, b(k)=0.5*(k-1); end
```

Program 85 - zplege : Coefficients of Gauss-Legendre formulae

```
function [x,w]=zplege(n)
if (n <= 1), disp(' n must be > 1 '); return; end
[a,b]=coeflege(n);
JacM=diag(a)+diag(sqrt(b(2:n)),1)+diag(sqrt(b(2:n)),-1);
[w,x]=eig(JacM); x=diag(x); scal=2; w=w(1,:)'.^2*scal;
[x,ind]=sort(x); w=w(ind);
```

Program 86 - zplagu : Coefficients of Gauss-Laguerre formulae

```
function [x,w]=zplagu(n)
if (n <= 1), disp(' n must be > 1 '); return; end
[a,b]=coeflagu(n);
JacM=diag(a)+diag(sqrt(b(2:n)),1)+diag(sqrt(b(2:n)),-1);
[w,x]=eig(JacM); x=diag(x); w=w(1,:)'.^2;
```

Program 87 - zpherm : Coefficients of Gauss-Hermite formulae

```
function [x,w]=zpherm(n)
if (n <= 1), disp(' n must be > 1 '); return; end
```

```
[a,b]=coefherm(n);
JacM=diag(a)+diag(sqrt(b(2:n)),1)+diag(sqrt(b(2:n)),-1);
[w,x]=eig(JacM); x=diag(x); scal=sqrt(pi); w=w(1,:)'.^2*scal;
[x,ind]=sort(x); w=w(ind);
```

10.7 Approximation of a Function in the Least-Squares Sense

Given a function $f \in \mathrm{L}^2(a,b)$, we look for a polynomial r_n of degree $\leq n$ that satisfies

$$\|f - r_n\|_w = \min_{p_n \in \mathbb{P}_n} \|f - p_n\|_w,$$

where w is a fixed weight function in (a,b). Should it exist, r_n is called a *least-squares polynomial*. The name derives from the fact that, if $w \equiv 1$, r_n is the polynomial that minimizes the mean-square error $E = \|f - r_n\|_{\mathrm{L}^2(a,b)}$ (see Exercise 8).

As seen in Section 10.1, r_n coincides with the truncation f_n of order n of the Fourier series (see (10.2) and (10.3)). Depending on the choice of the weight $w(x)$, different least-squares polynomials arise with different convergence properties.

Analogous to Section 10.1, we can introduce the discrete truncation f_n^* (10.4) of the Chebyshev series (setting $p_k = T_k$) or the Legendre series (setting $p_k = L_k$). If the discrete scalar product induced by the Gauss-Lobatto quadrature rule (10.28) is used in (10.5) then the $\tilde{f}_k$'s coincide with the coefficients of the expansion of the interpolating polynomial $\Pi_{n,w}^{GL} f$ (see (10.29) in the Chebyshev case, or (10.38) in the Legendre case).

Consequently, $f_n^* = \Pi_{n,w}^{GL} f$, i.e., the discrete truncation of the (Chebyshev or Legendre) series of f turns out to coincide with the interpolating polynomial at the $n+1$ Gauss-Lobatto nodes. In particular, in such a case (10.6) is trivially satisfied, since $\|f - f_n^*\|_n = 0$.

10.7.1 Discrete Least-Squares Approximation

Several applications require representing in a synthetic way, using elementary functions, a large set of data that are available at a discrete level, for instance, the results of experimental measurements. This approximation process, often referred to as *data fitting*, can be satisfactorily solved using the discrete least-squares technique that can be formulated as follows.

Assume we are given $m+1$ pairs of data

$$\{(x_i, y_i),\ i = 0, \ldots, m\} \tag{10.43}$$

where y_i may represent, for instance, the value of a physical quantity measured at the position x_i. We assume that all the abscissae are distinct.

We look for a polynomial $p_n(x) = \sum_{i=0}^{n} a_i \varphi_i(x)$ such that

$$\sum_{j=0}^{m} w_j |p_n(x_j) - y_j|^2 \leq \sum_{j=0}^{m} w_j |q_n(x_j) - y_j|^2 \quad \forall q_n \in \mathbb{P}_n, \tag{10.44}$$

for suitable coefficients $w_j > 0$. If $n = m$ the polynomial p_n clearly coincides with the interpolating polynomial of degree n at the nodes $\{x_i\}$. Problem (10.44) is called a *discrete least-squares problem* since a discrete scalar product is involved, and is the discrete counterpart of the continuous least-squares problem. The solution p_n is therefore referred to as a *least-squares polynomial*. Notice that

$$|||q||| = \left\{ \sum_{j=0}^{m} w_j [q(x_j)]^2 \right\}^{1/2} \tag{10.45}$$

is an *essentially strict* seminorm on $\mathbb{P}_n$ (see, Exercise 7). By definition a discrete norm (or seminorm) $\|\cdot\|_*$ is essentially strict if $\|f+g\|_* = \|f\|_* + \|g\|_*$ implies there exist nonnull α, β such that $\alpha f(x_i) + \beta g(x_i) = 0$ for $i = 0, \ldots, m$. Since $|||\cdot|||$ is an essentially strict seminorm, problem (10.44) admits a unique solution (see, [IK66], Section 3.5). Proceeding as in Section 3.13, we find

$$\sum_{k=0}^{n} a_k \sum_{j=0}^{m} w_j \varphi_k(x_j) \varphi_i(x_j) = \sum_{j=0}^{m} w_j y_j \varphi_i(x_j), \qquad \forall i = 0, \ldots, n,$$

which is called a *system of normal equations*, and can be conveniently written in the form

$$\mathrm{B}^T \mathrm{B} \mathbf{a} = \mathrm{B}^T \mathbf{y}, \tag{10.46}$$

where B is the rectangular matrix $(m+1)\times(n+1)$ of entries $b_{ij} = \varphi_j(x_i)$, $i = 0, \ldots, m$, $j = 0, \ldots, n$, $\mathbf{a} \in \mathbb{R}^{n+1}$ is the vector of the unknown coefficients and $\mathbf{y} \in \mathbb{R}^{m+1}$ is the vector of data.

Notice that the system of normal equations obtained in (10.46) is of the same nature as that introduced in Section 3.13 in the case of overdetermined systems. Actually, if $w_j = 1$ for $j = 0, \ldots, m$, the above system can be regarded as the solution in the least-squares sense of the system

$$\sum_{k=0}^{n} a_k \varphi_k(x_i) = y_i, \qquad i = 0, 1, \ldots, m,$$

which would not admit a solution in the classical sense, since the number of rows is greater than the number of columns. In the case $n = 1$, the solution to (10.44) is a linear function, called *linear regression* for the data fitting of (10.43). The associated system of normal equations is

$$\sum_{k=0}^{1}\sum_{j=0}^{m} w_j\varphi_i(x_j)\varphi_k(x_j)a_k = \sum_{j=0}^{m} w_j\varphi_i(x_j)y_j, \qquad i = 0, 1.$$

Setting $(f, g)_m = \sum_{j=0}^{m} f(x_j)g(x_j)$ the previous system becomes

$$\begin{cases} (\varphi_0, \varphi_0)_m a_0 + (\varphi_1, \varphi_0)_m a_1 = (y, \varphi_0)_m, \\ (\varphi_0, \varphi_1)_m a_0 + (\varphi_1, \varphi_1)_m a_1 = (y, \varphi_1)_m, \end{cases}$$

where $y(x)$ is a function that takes the value y_i at the nodes x_i, $i = 0, \ldots, m$. After some algebra, we get this explicit form for the coefficients

$$a_0 = \frac{(y, \varphi_0)_m(\varphi_1, \varphi_1)_m - (y, \varphi_1)_m(\varphi_1, \varphi_0)_m}{(\varphi_1, \varphi_1)_m(\varphi_0, \varphi_0)_m - (\varphi_0, \varphi_1)_m^2},$$

$$a_1 = \frac{(y, \varphi_1)_m(\varphi_0, \varphi_0)_m - (y, \varphi_0)_m(\varphi_1, \varphi_0)_m}{(\varphi_1, \varphi_1)_m(\varphi_0, \varphi_0)_m - (\varphi_0, \varphi_1)_m^2}.$$

Example 10.3 As already seen in Example 8.2, small changes in the data can give rise to large variations on the interpolating polynomial of a given function f. This doesn't happen for the least-squares polynomial where m is much larger than n. As an example, consider the function $f(x) = sin(2\pi x)$ in $[-1, 1]$ and evaluate it at the 22 equally spaced nodes $x_i = 2i/21$, $i = 0, \ldots, 21$, setting $f_i = f(x_i)$. Then, suppose to add to the data f_i a random perturbation of the order of 10^{-3} and denote by p_5 and $\tilde{p}_5$ the least-squares polynomials of degree 5 approximating the data f_i and $\tilde{f}_i$, respectively. The maximum norm of the difference $p_5 - \tilde{p}_5$ over $[-1, 1]$ is of the order of 10^{-3}, i.e., it is of the same order as the perturbation on the data. For comparison, the same difference in the case of Lagrange interpolation is about equal to 2 as can be seen in Figure 10.2. •

10.8 The Polynomial of Best Approximation

Consider a function $f \in C^0([a, b])$. A polynomial $p_n^* \in \mathbb{P}_n$ is said to be the *polynomial of best approximation of f* if it satisfies

$$\|f - p_n^*\|_\infty = \min_{p_n \in \mathbb{P}_n} \|f - p_n\|_\infty, \quad \forall p_n \in \mathbb{P}_n \tag{10.47}$$

where $\|g\|_\infty = \max_{a \le x \le b} |g(x)|$. This problem is referred to as a *minimax approximation*, as we are looking for the minimum error measured in the maximum norm.

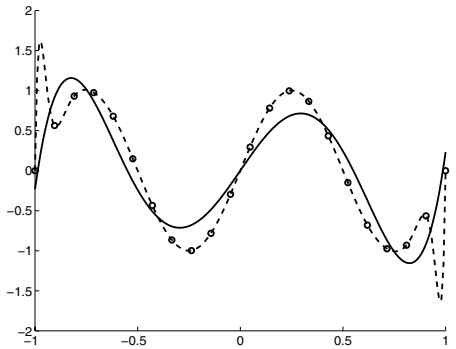

FIGURE 10.2. The perturbed data (circles), the associated least-squares polynomial of degree 5 (solid line) and the Lagrange interpolating polynomial (dashed line)

Property 10.1 (Chebyshev equioscillation theorem) *For any $n \geq 0$, the polynomial of best approximation p_n^* of f exists and is unique. Moreover, in $[a, b]$ there exist $n + 2$ points $x_0 < x_1 < \ldots < x_{n+1}$ such that*

$$f(x_j) - p_n^*(x_j) = \sigma(-1)^j E_n^*(f), \qquad j = 0, \ldots, n+1$$

with $\sigma = 1$ or $\sigma = -1$ depending on f and n, and $E_n^*(f) = \|f - p_n^*\|_\infty$.

(For the proof, see [Dav63], Chapter 7). As a consequence, there exist $n+1$ points $\tilde{x}_0 < \tilde{x}_1 < \ldots < \tilde{x}_n$, with $x_k < \tilde{x}_k < x_{k+1}$ for $k = 0, \ldots, n$, to be determined in $[a, b]$ such that

$$p_n^*(\tilde{x}_j) = f(\tilde{x}_j), \quad j = 0, 1, \ldots, n,$$

so that the best approximation polynomial is a polynomial of degree n that interpolates f at $n + 1$ unknown nodes.

The following result yields an estimate of $E_n^*(f)$ without explicitly computing p_n^* (we refer for the proof to [Atk89], Chapter 4).

Property 10.2 (de la Vallée-Poussin theorem) *Let $n \geq 0$ and let $x_0 < x_1 < \ldots < x_{n+1}$ be $n + 2$ points in $[a, b]$. If there exists a polynomial q_n of degree $\leq n$ such that*

$$f(x_j) - q_n(x_j) = (-1)^j e_j \quad j = 0, 1, \ldots, n+1$$

where all e_j have the same sign and are non null, then

$$\min_{0 \leq j \leq n+1} |e_j| \leq E_n^*(f).$$

We can now relate $E_n^*(f)$ with the interpolation error. Indeed,

$$\|f - \Pi_n f\|_\infty \leq \|f - p_n^*\|_\infty + \|p_n^* - \Pi_n f\|_\infty.$$

On the other hand, using the Lagrange representation of p_n^* we get

$$\|p_n^* - \Pi_n f\|_\infty = \|\sum_{i=0}^{n}(p_n^*(x_i) - f(x_i))l_i\|_\infty \leq \|p_n^* - f\|_\infty \|\sum_{i=0}^{n}|l_i|\|_\infty,$$

from which it follows

$$\|f - \Pi_n f\|_\infty \leq (1 + \Lambda_n)E_n^*(f),$$

where Λ_n is the Lebesgue constant (8.11) associated with the nodes $\{x_i\}$. Thanks to (10.25) we can conclude that the Lagrange interpolating polynomial on the Chebyshev nodes is a good approximation of p_n^*. The above results yield a characterization of the best approximation polynomial, but do not provide a constructive way for generating it. However, starting from the Chebyshev equioscillation theorem, it is possible to devise an algorithm, called the Remes algorithm, that is able to construct an arbitrarily good approximation of the polynomial p_n^* (see [Atk89], Section 4.7).

10.9 Fourier Trigonometric Polynomials

Let us apply the theory developed in the previous sections to a particular family of orthogonal polynomials which are no longer algebraic polynomials but rather trigonometric. The *Fourier polynomials* on $(0, 2\pi)$ are defined as

$$\varphi_k(x) = e^{ikx}, \qquad k = 0, \pm 1, \pm 2, \ldots$$

where i is the imaginary unit. These are complex-valued periodic functions with period equal to 2π. We shall use the notation $\mathrm{L}^2(0, 2\pi)$ to denote the complex-valued functions that are square integrable over $(0, 2\pi)$. Therefore

$$\mathrm{L}^2(0, 2\pi) = \left\{ f : (0, 2\pi) \to \mathbb{C} \text{ such that } \int_0^{2\pi} |f(x)|^2 dx < \infty \right\}$$

with scalar product and norm defined respectively by

$$(f, g) = \textstyle\int_0^{2\pi} f(x)\overline{g(x)}dx, \quad \|f\|_{\mathrm{L}^2(0,2\pi)} = \sqrt{(f, f)}.$$

If $f \in \mathrm{L}^2(0, 2\pi)$, its Fourier series is

$$Ff = \sum_{k=-\infty}^{\infty} \widehat{f}_k \varphi_k, \quad \text{with } \widehat{f}_k = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-ikx}dx = \frac{1}{2\pi}(f, \varphi_k). \quad (10.48)$$

If f is complex-valued we set $f(x) = \alpha(x) + i\beta(x)$ for $x \in [0, 2\pi]$, where $\alpha(x)$ is the real part of f and $\beta(x)$ is the imaginary one. Recalling that

$e^{-ikx} = \cos(kx) - i\sin(kx)$ and letting

$$a_k = \frac{1}{2\pi}\int_0^{2\pi} [\alpha(x)\cos(kx) + \beta(x)\sin(kx)]\,dx$$
$$b_k = \frac{1}{2\pi}\int_0^{2\pi} [-\alpha(x)\sin(kx) + \beta(x)\cos(kx)]\,dx,$$

the *Fourier coefficients* of the function f can be written as

$$\widehat{f}_k = a_k + ib_k \qquad \forall k = 0, \pm 1, \pm 2, \dots . \tag{10.49}$$

We shall assume henceforth that f is a real-valued function; in such a case $\widehat{f}_{-k} = \overline{\widehat{f}_k}$ for any k.

Let N be an even positive integer. Analogously to what was done in Section 10.1, we call the *truncation of order* N of the Fourier series the function

$$f_N^*(x) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \widehat{f}_k e^{ikx}.$$

The use of capital N instead of small n is to conform with the notation usually adopted in the analysis of discrete Fourier series (see [Bri74], [Wal91]). To simplify the notations we also introduce an index shift so that

$$f_N^*(x) = \sum_{k=0}^{N-1} \widehat{f}_k e^{i(k-\frac{N}{2})x},$$

where now

$$\widehat{f}_k = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-i(k-N/2)x}dx = \frac{1}{2\pi}(f, \widetilde{\varphi}_k),\ k = 0, \dots, N-1 \tag{10.50}$$

and $\widetilde{\varphi}_k = e^{-i(k-N/2)x}$. Denoting by

$$S_N = \text{span}\{\widetilde{\varphi}_k,\, 0 \le k \le N-1\},$$

if $f \in \mathrm{L}^2(0, 2\pi)$ its truncation of order N satisfies the following optimal approximation property in the least-squares sense

$$\|f - f_N^*\|_{\mathrm{L}^2(0,2\pi)} = \min_{g \in S_N} \|f - g\|_{\mathrm{L}^2(0,2\pi)}.$$

Set $h = 2\pi/N$ and $x_j = jh$, for $j = 0, \dots, N-1$, and introduce the following *discrete scalar product*

$$(f, g)_N = h\sum_{j=0}^{N-1} f(x_j)\overline{g(x_j)}. \tag{10.51}$$

Replacing $(f, \widetilde{\varphi}_k)$ in (10.50) with $(f, \widetilde{\varphi}_k)_N$, we get the *discrete Fourier coefficients* of the function f

$$\widetilde{f}_k = \frac{1}{N}\sum_{j=0}^{N-1} f(x_j)e^{-ikjh}e^{ij\pi} = \frac{1}{N}\sum_{j=0}^{N-1} f(x_j)W_N^{(k-\frac{N}{2})j} \tag{10.52}$$

for $k = 0, \ldots, N-1$, where

$$W_N = \exp\left(-i\frac{2\pi}{N}\right)$$

is the *principal root of order* N of unity. According to (10.4), the trigonometric polynomial

$$\Pi_N^F f(x) = \sum_{k=0}^{N-1} \widetilde{f}_k e^{i(k-\frac{N}{2})x} \tag{10.53}$$

is called the *discrete Fourier series of order* N of f.

Lemma 10.1 *The following property holds*

$$(\varphi_l, \varphi_j)_N = h\sum_{k=0}^{N-1} e^{-ik(l-j)h} = 2\pi\delta_{jl}, \qquad 0 \le l, j \le N-1, \tag{10.54}$$

where δ_{jl} *is the Kronecker symbol.*

Proof. For $l = j$ the result is immediate. Thus, assume $l \neq j$; we have that

$$\sum_{k=0}^{N-1} e^{-ik(l-j)h} = \frac{1 - \left(e^{-i(l-j)h}\right)^N}{1 - e^{-i(l-j)h}} = 0.$$

Indeed, the numerator is $1 - (\cos(2\pi(l-j)) - i\sin(2\pi(l-j))) = 1 - 1 = 0$, while the denominator cannot vanish. Actually, it vanishes iff $(j-l)h = 2\pi$, i.e., $j - l = N$, which is impossible. ◇

Thanks to Lemma 10.1, the trigonometric polynomial $\Pi_N^F f$ is the Fourier *interpolate* of f at the nodes x_j, that is

$$\Pi_N^F f(x_j) = f(x_j), \qquad j = 0, 1, \ldots, N-1.$$

Indeed, using (10.52) and (10.54) in (10.53) it follows that

$$\Pi_N^F f(x_j) = \sum_{k=0}^{N-1} \widetilde{f}_k e^{ikjh}e^{-ijh\frac{N}{2}} = \sum_{l=0}^{N-1} f(x_l)\left[\frac{1}{N}\sum_{k=0}^{N-1} e^{-ik(l-j)h}\right] = f(x_j).$$

Therefore, looking at the first and last equality, we get

$$f(x_j) = \sum_{k=0}^{N-1} \widetilde{f}_k e^{ik(j-\frac{N}{2})h} = \sum_{k=0}^{N-1} \widetilde{f}_k W_N^{-(j-\frac{N}{2})k}, \ j = 0, \dots, N-1. \quad (10.55)$$

The mapping $\{f(x_j)\} \to \{\widetilde{f}_k\}$ described by (10.52) is called the *Discrete Fourier Transform* (DFT), while the mapping (10.55) from $\{\widetilde{f}_k\}$ to $\{f(x_j)\}$ is called the *inverse transform* (IDFT). Both DFT and IDFT can be written in matrix form as $\{\widetilde{f}_k\} = \mathrm{T}\{f(x_j)\}$ and $\{f(x_j)\} = \mathrm{C}\{\widetilde{f}_k\}$ where $\mathrm{T} \in \mathbb{C}^{N\times N}$, C denotes the inverse of T and

$$T_{kj} = \frac{1}{N} W_N^{(k-\frac{N}{2})j}, \quad k, j = 0, \dots, N-1,$$
$$C_{jk} = W_N^{-(j-\frac{N}{2})k}, \quad j, k = 0, \dots, N-1.$$

A naive implementation of the matrix-vector computation in the DFT and IDFT would require N^2 operations. Using the FFT (*Fast Fourier Transform*) algorithm only $\mathcal{O}(N \log_2 N)$ flops are needed, provided that N is a power of 2, as will be explained in Section 10.9.2.

The function $\Pi_N^F f \in S_N$ introduced in (10.53) is the solution of the minimization problem $\|f - \Pi_N^F f\|_N \le \|f - g\|_N$ for any $g \in S_N$, where $\|\cdot\|_N = (\cdot,\cdot)_N^{1/2}$ is a discrete norm for S_N. In the case where f is periodic with all its derivatives up to order s ($s \ge 1$), an error estimate analogous to that for Chebyshev and Legendre interpolation holds

$$\|f - \Pi_N^F f\|_{\mathrm{L}^2(0,2\pi)} \le C N^{-s} \|f\|_s$$

and also

$$\max_{0\le x\le 2\pi} |f(x) - \Pi_N^F f(x)| \le C N^{1/2-s} \|f\|_s.$$

In a similar manner, we also have

$$|(f, v_N) - (f, v_N)_N| \le C N^{-s} \|f\|_s \|v_N\|$$

for any $v_N \in S_N$, and in particular, setting $v_N = 1$ we have the following error for the quadrature formula (10.51)

$$\left| \int_0^{2\pi} f(x)dx - h \sum_{j=0}^{N-1} f(x_j) \right| \le C N^{-s} \|f\|_s$$

(see for the proof [CHQZ88], Chapter 2).

Notice that $h \sum_{j=0}^{N-1} f(x_j)$ is nothing else than the composite trapezoidal rule for approximating the integral $\int_0^{2\pi} f(x)dx$. Therefore, such a formula

turns out to be extremely accurate when dealing with periodic and smooth integrands.

Programs 88 and 89 provide an implementation of the DFT and IDFT. The input parameter `f` is a string containing the function f to be transformed while `fc` is a vector of size N containing the values $\widetilde{f}_k$.

Program 88 - dft : Discrete Fourier transform

```
function fc = dft(N,f)
h = 2*pi/N; x=[0:h:2*pi*(1-1/N)]; fx = eval(f); wn = exp(-i*h);
for k=0:N-1,
   s = 0;
   for j=0:N-1
      s = s + fx(j+1)*wn^((k-N/2)*j);
   end
   fc (k+1) = s/N;
end
```

Program 89 - idft : Inverse discrete Fourier transform

```
function fv = idft(N,fc)
h  = 2*pi/N; wn = exp(-i*h);
for k=0:N-1
   s = 0;
   for j=0:N-1
      s = s + fc(j+1)*wn^(-k*(j-N/2));
   end
   fv (k+1) = s;
end
```

10.9.1 The Gibbs Phenomenon

Consider the discontinuous function $f(x) = x/\pi$ for $x \in [0, \pi]$ and equal to $x/\pi - 2$ for $x \in (\pi, 2\pi]$, and compute its DFT using Program 88. The interpolate $\Pi_N^F f$ is shown in Figure 10.3 (above) for $N = 8, 16, 32$. Notice the spurious oscillations around the point of discontinuity of f whose maximum amplitude, however, tends to a finite limit. The arising of these oscillations is known as *Gibbs phenomenon* and is typical of functions with isolated jump discontinuities; it affects the behavior of the truncated Fourier series not only in the neighborhood of the discontinuity but also over the entire interval, as can be clearly seen in the figure. The convergence rate of the truncated series for functions with jump discontinuities is linear in N^{-1} at every given non-singular point of the interval of definition of the function (see [CHQZ88], Section 2.1.4).

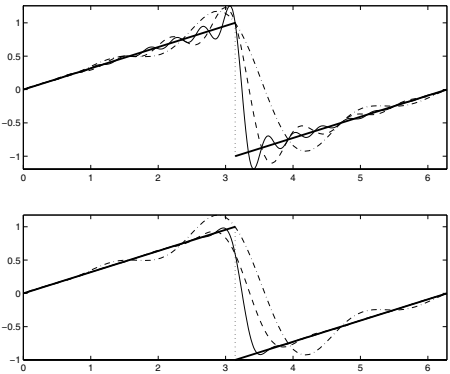

FIGURE 10.3. Above: Fourier interpolate of the sawtooth function (thick solid line) for $N = 8$ (dash-dotted line), 16 (dashed line) and 32 (thin solid line). Below: the same informations are plotted in the case of the Lanczos smoothing

Since the Gibbs phenomenon is related to the slow decay of the Fourier coefficients of a discontinuous function, smoothing procedures can be profitably employed to attenuate the higher-order Fourier coefficients. This can be done by multiplying each coefficient $\widetilde{f}_k$ by a factor σ_k such that σ_k is a decreasing function of k. An example is provided by the *Lanczos smoothing*

$$\sigma_k = \frac{\sin(2(k - N/2)(\pi/N))}{2(k - N/2)(\pi/N)}, \qquad k = 0, \ldots, N-1. \tag{10.56}$$

The effect of applying the Lanczos smoothing to the computation of the DFT of the above function f is represented in Figure 10.3 (below), which shows that the oscillations have almost completely disappeared.

For a deeper analysis of this subject we refer to [CHQZ88], Chapter 2.

10.9.2 The Fast Fourier Transform

As pointed out in the previous section, computing the discrete Fourier transform (DFT) or its inverse (IDFT) as a matrix-vector product, would require N^2 operations. In this section we illustrate the basic steps of the Cooley-Tukey algorithm [CT65], commonly known as Fast Fourier Transform (FFT). The computation of a DFT of order N is split into DFTs of order $p_0, \ldots, p_m$, where $\{p_i\}$ are the prime factors of N. If N is a power of 2, the computational cost has the order of $N \log_2 N$ flops.

A recursive algorithm to compute the DFT when N is a power of 2 is described in the following. Let $\mathbf{f} = (f_i)^T$, $i = 0, \ldots, N-1$ and set $p(x) = \frac{1}{N} \sum_{j=0}^{N-1} f_j x^j$. Then, computing the DFT of the vector $\mathbf{f}$ amounts to

evaluating $p(W_N^{k-\frac{N}{2}})$ for $k = 0, \ldots, N-1$. Let us introduce the polynomials

$$p_e(x) = \frac{1}{N}\left[f_0 + f_2 x + \ldots + f_{N-2} x^{\frac{N}{2}-1}\right],$$
$$p_o(x) = \frac{1}{N}\left[f_1 + f_3 x + \ldots + f_{N-1} x^{\frac{N}{2}-1}\right].$$

Notice that

$$p(x) = p_e(x^2) + x p_o(x^2)$$

from which it follows that the computation of the DFT of $\mathbf{f}$ can be carried out by evaluating the polynomials p_e and p_o at the points $W_N^{2(k-\frac{N}{2})}$, $k = 0, \ldots, N-1$. Since

$$W_N^{2(k-\frac{N}{2})} = W_N^{2k-N} = \exp\left(-i\frac{2\pi k}{N/2}\right)\exp(i2\pi) = W_{N/2}^k,$$

it turns out that we must evaluate p_e and p_o at the principal roots of unity of order $N/2$. In this manner the DFT of order N is rewritten in terms of two DFTs of order $N/2$; of course, we can recursively apply again this procedure to p_o and p_e. The process is terminated when the degree of the last generated polynomials is equal to one.

In Program 90 we propose a simple implementation of the FFT recursive algorithm. The input parameters are the vector `f` containing the `NN` values f_k, where `NN` is a power of 2.

Program 90 - fftrec : FFT algorithm in the recursive version

```
function [fftv]=fftrec(f,NN)
N = length(f);   w = exp(-2*pi*sqrt(-1)/N);
if N == 2
   fftv = f(1)+w.^[-NN/2:NN-1-NN/2]*f(2);
else
   a1 = f(1:2:N);   b1 = f(2:2:N);
   a2 = fftrec(a1,NN); b2 = fftrec(b1,NN);
   for k=-NN/2:NN-1-NN/2
      fftv(k+1+NN/2) = a2(k+1+NN/2) + b2(k+1+NN/2)*w^k;
   end
end
```

Remark 10.4 A FFT procedure can also be set up when N is not a power of 2. The simplest approach consists of adding some zero samples to the original sequence $\{f_i\}$ in such a way to obtain a total number of $\tilde{N} = 2^p$ values. This technique, however, does not necessarily yield the correct result. Therefore, an effective alternative is based on partitioning the Fourier matrix C into subblocks of smaller size. Practical FFT implementations can handle both strategies (see, for instance, the `fft` package available in MATLAB). ■

10.10 Approximation of Function Derivatives

A problem which is often encountered in numerical analysis is the approximation of the derivative of a function $f(x)$ on a given interval $[a, b]$. A natural approach to it consists of introducing in $[a, b]$ $n + 1$ nodes $\{x_k,\ k = 0, \ldots, n\}$, with $x_0 = a$, $x_n = b$ and $x_{k+1} = x_k + h$, $k = 0, \ldots, n-1$ where $h = (b - a)/n$. Then, we approximate $f'(x_i)$ using the nodal values $f(x_k)$ as

$$h \sum_{k=-m}^{m} \alpha_k u_{i-k} = \sum_{k=-m'}^{m'} \beta_k f(x_{i-k}), \tag{10.57}$$

where $\{\alpha_k\}$, $\{\beta_k\} \in \mathbb{R}$ are $m + m' + 1$ coefficients to be determined and u_k is the desired approximation to $f'(x_k)$.

A non negligible issue in the choice of scheme (10.57) is the computational efficiency. Regarding this concern, it is worth noting that, if $m \neq 0$, determining the values $\{u_i\}$ requires the solution of a linear system.

The set of nodes which are involved in constructing the derivative of f at a certain node, is called a *stencil*. The band of the matrix associated with system (10.57) increases as the stencil gets larger.

10.10.1 Classical Finite Difference Methods

The simplest way to generate a formula like (10.57) consists of resorting to the definition of the derivative. If $f'(x_i)$ exists, then

$$f'(x_i) = \lim_{h \to 0^+} \frac{f(x_i + h) - f(x_i)}{h}. \tag{10.58}$$

Replacing the limit with the incremental ratio, with h finite, yields the approximation

$$u_i^{FD} = \frac{f(x_{i+1}) - f(x_i)}{h}, \qquad 0 \leq i \leq n - 1. \tag{10.59}$$

Relation (10.59) is a special instance of (10.57) setting $m = 0$, $\alpha_0 = 1$, $m' = 1$, $\beta_{-1} = 1$, $\beta_0 = -1$, $\beta_1 = 0$.
The right side of (10.59) is called the *forward finite difference* and the approximation that is being used corresponds to replacing $f'(x_i)$ with the slope of the straight line passing through the points $(x_i, f(x_i))$ and $(x_{i+1}, f(x_{i+1}))$, as shown in Figure 10.4.

To estimate the error that is made, it suffices to expand f in Taylor's series, obtaining

$$f(x_{i+1}) = f(x_i) + hf'(x_i) + \frac{h^2}{2} f''(\xi_i) \qquad \text{with } \xi_i \in (x_i, x_{i+1}).$$

We assume henceforth that f has the required regularity, so that

$$f'(x_i) - u_i^{FD} = -\frac{h}{2} f''(\xi_i). \tag{10.60}$$

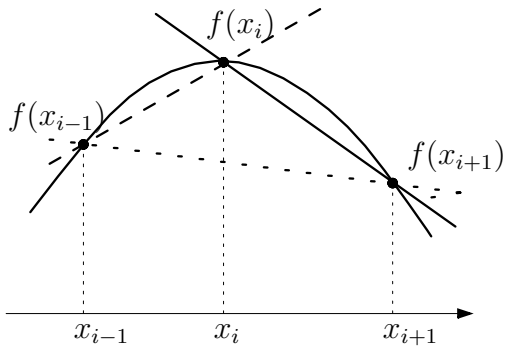

FIGURE 10.4. Finite difference approximation of $f'(x_i)$: backward (solid line), forward (pointed line) and centred (dashed line)

Obviously, instead of (10.58) we could employ a centred incremental ratio, obtaining the following approximation

$$u_i^{CD} = \frac{f(x_{i+1}) - f(x_{i-1})}{2h}, \qquad 1 \leq i \leq n-1. \tag{10.61}$$

Scheme (10.61) is a special instance of (10.57) setting $m = 0$, $\alpha_0 = 1$, $m' = 1$, $\beta_{-1} = 1/2$, $\beta_0 = 0$, $\beta_1 = -1/2$.

The right side of (10.61) is called the *centred finite difference* and geometrically amounts to replacing $f'(x_i)$ with the slope of the straight line passing through the points $(x_{i-1}, f(x_{i-1}))$ and $(x_{i+1}, f(x_{i+1}))$ (see Figure 10.4). Resorting again to Taylor's series, we get

$$f'(x_i) - u_i^{CD} = -\frac{h^2}{6} f'''(\xi_i). \tag{10.62}$$

Formula (10.61) thus provides a second-order approximation to $f'(x_i)$ with respect to h.

Finally, with a similar procedure, we can derive a *backward finite difference* scheme, where

$$u_i^{BD} = \frac{f(x_i) - f(x_{i-1})}{h}, \qquad 1 \leq i \leq n, \tag{10.63}$$

which is affected by the following error

$$f'(x_i) - u_i^{BD} = \frac{h}{2} f''(\xi_i). \tag{10.64}$$

The values of the parameters in (10.57) are $m = 0$, $\alpha_0 = 1$, $m' = 1$ and $\beta_{-1} = 0$, $\beta_0 = 1$, $\beta_1 = -1$.

Higher-order schemes, as well as finite difference approximations of higher-order derivatives of f, can be constructed using Taylor's expansions of higher order. A remarkable example is the approximation of f''; if $f \in C^4([a,b])$ we easily get

$$\begin{aligned} f''(x_i) &= \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} \\ &\quad -\frac{h^2}{24}\left(f^{(4)}(x_i + \theta_i h) + f^{(4)}(x_i - \omega_i h)\right), \qquad 0 < \theta_i, \omega_i < 1. \end{aligned}$$

The following *centred finite difference* scheme can thus be derived

$$u_i'' = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2}, \qquad 1 \le i \le n-1 \tag{10.65}$$

which is affected by the error

$$f''(x_i) - u_i'' = -\frac{h^2}{24}\left(f^{(4)}(x_i + \theta_i h) + f^{(4)}(x_i - \omega_i h)\right). \tag{10.66}$$

Formula (10.65) provides a second-order approximation to $f''(x_i)$ with respect to h.

10.10.2 Compact Finite Differences

More accurate approximations are provided by using the following formula (which we call *compact differences*)

$$\alpha u_{i-1} + u_i + \alpha u_{i+1} = \frac{\beta}{2h}(f_{i+1} - f_{i-1}) + \frac{\gamma}{4h}(f_{i+2} - f_{i-2}) \tag{10.67}$$

for $i = 2, \ldots, n-1$. We have set, for brevity, $f_i = f(x_i)$.

The coefficients α, β and γ are to be determined in such a way that the relations (10.67) yield values u_i that approximate $f'(x_i)$ up to the highest order with respect to h. For this purpose, the coefficients are selected in such a way as to minimize the *consistency error* (see Section 2.2)

$$\begin{aligned} \sigma_i(h) &= \alpha f_{i-1}^{(1)} + f_i^{(1)} - \alpha f_{i+1}^{(1)} \\ &\quad -\left(\frac{\beta}{2h}(f_{i+1} - f_{i-1}) + \frac{\gamma}{4h}(f_{i+2} - f_{i-2})\right) \end{aligned} \tag{10.68}$$

which comes from "forcing" f to satisfy the numerical scheme (10.67). For brevity, we set $f_i^{(k)} = f^{(k)}(x_i)$, $k = 1, 2, \ldots$.

Precisely, assuming that $f \in C^5([a, b])$ and expanding it in a Taylor's series around x_i, we find

$$f_{i\pm1} = f_i \pm h f_i^{(1)} + \tfrac{h^2}{2} f_i^{(2)} \pm \tfrac{h^3}{6} f_i^{(3)} + \tfrac{h^4}{24} f_i^{(4)} \pm \tfrac{h^5}{120} f_i^{(5)} + \mathcal{O}(h^6),$$

$$f_{i\pm1}^{(1)} = f_i^{(1)} \pm h f_i^{(2)} + \tfrac{h^2}{2} f_i^{(3)} \pm \tfrac{h^3}{6} f_i^{(4)} + \tfrac{h^4}{24} f_i^{(5)} + \mathcal{O}(h^5).$$

Substituting into (10.68) we get

$$\sigma_i(h) = (2\alpha + 1) f_i^{(1)} + \alpha \frac{h^2}{2} f_i^{(3)} + \alpha \frac{h^4}{12} f_i^{(5)} - (\beta + \gamma) f_i^{(1)}$$

$$- \frac{h^2}{2} \left(\frac{\beta}{6} + \frac{2\gamma}{3} \right) f_i^{(3)} - \frac{h^4}{60} \left(\frac{\beta}{2} + 8\gamma \right) f_i^{(5)} + \mathcal{O}(h^6).$$

Second-order methods are obtained by equating to zero the coefficient of $f_i^{(1)}$, i.e., if $2\alpha + 1 = \beta + \gamma$, while we obtain schemes of order 4 by equating to zero also the coefficient of $f_i^{(3)}$, yielding $6\alpha = \beta + 4\gamma$ and finally, methods of order 6 are obtained by setting to zero also the coefficient of $f_i^{(5)}$, i.e., $10\alpha = \beta + 16\gamma$.

The linear system formed by these last three equations has a nonsingular matrix. Thus, there exists a unique scheme of order 6 that corresponds to the following choice of the parameters

$$\alpha = 1/3, \quad \beta = 14/9, \quad \gamma = 1/9, \tag{10.69}$$

while there exist infinitely many methods of second and fourth order. Among these infinite methods, a popular scheme has coefficients $\alpha = 1/4$, $\beta = 3/2$ and $\gamma = 0$. Schemes of higher order can be generated at the expense of furtherly expanding the computational stencil.

Traditional finite difference schemes correspond to setting $\alpha = 0$ and allow for computing explicitly the approximant of the first derivative of f at a node, in contrast with compact schemes which are required in any case to solve a linear system of the form $\mathrm{A}\mathbf{u} = \mathrm{B}\mathbf{f}$ (where the notation has the obvious meaning).

To make the system solvable, it is necessary to provide values to the variables u_i with $i < 0$ and $i > n$. A particularly favorable instance is that where f is a periodic function of period $b - a$, in which case $u_{i+n} = u_i$ for any $i \in \mathbb{Z}$. In the nonperiodic case, system (10.67) must be supplied by suitable relations at the nodes near the boundary of the approximation interval. For example, the first derivative at x_0 can be computed using the relation

$$u_0 + \alpha u_1 = \frac{1}{h}(A f_1 + B f_2 + C f_3 + D f_4),$$

and requiring that

$$A = -\frac{3+\alpha+2D}{2}, \quad B = 2+3D, \quad C = -\frac{1-\alpha+6D}{2},$$

in order for the scheme to be at least second-order accurate (see [Lel92] for the relations to enforce in the case of higher-order methods). Finally, we notice that, for any given order of accuracy, compact schemes have a stencil smaller than the one of standard finite differences.

Program 91 provides an implementation of the compact finite difference schemes (10.67) for the approximation of the derivative of a given function f which is assumed to be periodic on the interval $[a, b)$. The input parameters `alpha`, `beta` and `gamma` contain the coefficients of the scheme, `a` and `b` are the endpoints of the interval, `f` is a string containing the expression of f and `n` denotes the number of subintervals in which $[a, b]$ is partitioned. The output vectors `u` and `x` contain the computed approximate values u_i and the node coordinates. Notice that setting `alpha`=`gamma`=0 and `beta`=1 we recover the centered finite difference approximation (10.61).

Program 91 - compdiff : Compact difference schemes

```
function [u, x] = compdiff(alpha,beta,gamma,a,b,n,f)
h=(b-a)/(n+1); x=[a:h:b]; fx = eval(f);
A   = eye(n+2)+alpha*diag(ones(n+1,1),1)+alpha*diag(ones(n+1,1),-1);
rhs = 0.5*beta/h*(fx(4:n+1)-fx(2:n-1))+0.25*gamma/h*(fx(5:n+2)-fx(1:n-2));
if gamma == 0
   rhs=[0.5*beta/h*(fx(3)-fx(1)), rhs, 0.5*beta/h*(fx(n+2)-fx(n))];
   A(1,1:n+2) = zeros(1,n+2);
   A(1,1) = 1; A(1,2)=alpha; A(1,n+1)=alpha;
   rhs=[0.5*beta/h*(fx(2)-fx(n+1)), rhs];
   A(n+2,1:n+2) = zeros(1,n+2);
   A(n+2,n+2) = 1; A(n+2,n+1)=alpha; A(n+2,2)=alpha;
   rhs=[rhs, 0.5*beta/h*(fx(2)-fx(n+1))];
else
   rhs=[0.5*beta/h*(fx(3)-fx(1))+0.25*gamma/h*(fx(4)-fx(n+1)), rhs];
   A(1,1:n+2) = zeros(1,n+2);
   A(1,1) = 1; A(1,2)=alpha; A(1,n+1)=alpha;
   rhs=[0.5*beta/h*(fx(2)-fx(n+1))+0.25*gamma/h*(fx(3)-fx(n)), rhs];
   rhs=[rhs,0.5*beta/h*(fx(n+2)-fx(n))+0.25*gamma/h*(fx(2)-fx(n-1))];
   A(n+2,1:n+2) = zeros(1,n+2);
   A(n+2,n+2) = 1; A(n+2,n+1)=alpha; A(n+2,2)=alpha;
   rhs=[rhs,0.5*beta/h*(fx(2)-fx(n+1))+0.25*gamma/h*(fx(3)-fx(n))];
end
u = A \ rhs';
return
```

Example 10.4 Let us consider the approximate evaluation of the derivative of the function $f(x) = \sin(x)$ on the interval $[0, 2\pi]$. Figure 10.5 shows the loga-

rithm of the maximum nodal errors for the second-order centered finite difference scheme (10.61) and of the fourth and sixth-order compact difference schemes introduced above, as a function of $p = \log(n)$. •

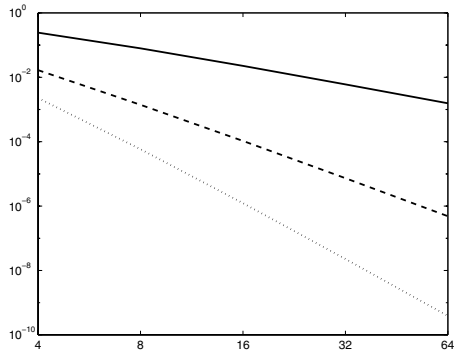

FIGURE 10.5. Maximum nodal errors for the second-order centred finite difference scheme (solid line) and for the fourth (dashed line) and sixth-order (dotted line) compact difference schemes as functions of $p = \log(n)$

Another nice feature of compact schemes is that they maximize the range of *well-resolved waves* as we are going to explain. Assume that f is a real and periodic function on $[0, 2\pi]$, that is, $f(0) = f(2\pi)$. Using the same notation as in Section 10.9, we let N be an even positive integer and set $h = 2\pi/N$. Then replace f by its truncated Fourier series

$$f_N^*(x) = \sum_{k=-N/2}^{N/2-1} \widehat{f}_k e^{ikx}.$$

Since the function f is real-valued, $\widehat{f}_k = \overline{\widehat{f}}_{-k}$ for $k = 1, \ldots, N/2$ and $\widehat{f}_0 = \overline{\widehat{f}}_0$. For sake of convenience, introduce the *normalized wave number* $w_k = kh = 2\pi k/N$ and perform a scaling of the coordinates setting $s = x/h$. As a consequence, we get

$$f_N^*(x(s)) = \sum_{k=-N/2}^{N/2-1} \widehat{f}_k e^{iksh} = \sum_{k=-N/2}^{N/2-1} \widehat{f}_k e^{iw_k s}. \tag{10.70}$$

Taking the first derivative of (10.70) with respect to s yields a function whose Fourier coefficients are $\widehat{f}'_k = iw_k \widehat{f}_k$. We can thus estimate the approximation error on $(f_N^*)'$ by comparing the exact coefficients $\widehat{f}'_k$ with the corresponding ones obtained by an approximate derivative, in particular, by comparing the exact wave number w_k with the approximate one, say $w_{k,app}$.

Let us neglect the subscript k and perform the comparison over the whole interval $[0, \pi)$ where w_k is varying. It is clear that methods based on the Fourier expansion have $w_{app} = w$ if $w \neq \pi$ ($w_{app} = 0$ if $w = \pi$). The family of schemes (10.67) is instead characterized by the wave number

$$w_{app}(z) = \frac{a\sin(z) + (b/2)\sin(2z) + (c/3)\sin(3z)}{1 + 2\alpha\cos(z) + 2\beta\cos(2z)}, \qquad z \in [0, \pi)$$

(see [Lel92]). Figure 10.6 displays a comparison among wave numbers of several schemes, of compact and non compact type.

The range of values for which the wave number computed by the numerical scheme adequately approximates the exact wave number, is the set of *well-resolved* waves. As a consequence, if w_{min} is the smallest well-resolved wave, the difference $1 - w_{min}/\pi$ represents the fraction of waves that are unresolved by the numerical scheme. As can be seen in Figure 10.6, the standard finite difference schemes approximate correctly the exact wave number only for small wave numbers.

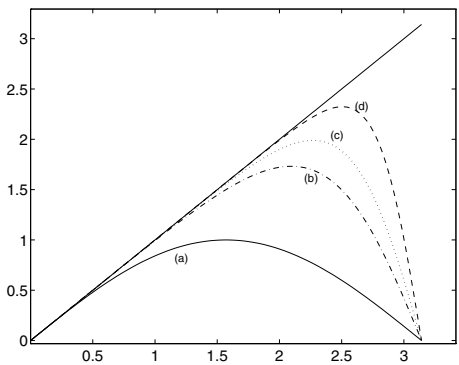

FIGURE 10.6. Computed wave numbers for centred finite differences (10.61) (a) and for compact schemes of fourth (b), sixth (c) and tenth (d) order, compared with the exact wave number (the straight line). On the x axis the normalized coordinate s is represented

10.10.3 Pseudo-Spectral Derivative

An alternative way for numerical differentiation consists of approximating the first derivative of a function f with the exact first derivative of the polynomial $\Pi_n f$ interpolating f at the nodes $\{x_0, \ldots, x_n\}$.

Exactly as happens for Lagrange interpolation, using equally spaced nodes does not yield stable approximations to the first derivative of f for n large. For this reason, we limit ourselves to considering the case where the nodes are nonuniformly distributed according to the Gauss-Lobatto-Chebyshev formula.

For simplicity, assume that $I = [a, b] = [-1, 1]$ and for $n \geq 1$, take in I the Gauss-Lobatto-Chebyshev nodes as in (10.21). Then, consider the Lagrange interpolating polynomial $\Pi_{n,w}^{GL} f$, introduced in Section 10.3. We define the *pseudo-spectral derivative* of $f \in C^0(I)$ to be the derivative of the polynomial $\Pi_{n,w}^{GL} f$

$$\mathcal{D}_n f = (\Pi_{n,w}^{GL} f)' \in \mathbb{P}_{n-1}(I).$$

The error made in replacing f' with $\mathcal{D}_n f$ is of *exponential* type, that is, it only depends on the smoothness of the function f. More precisely, there exists a constant $C > 0$ independent of n such that

$$\|f' - \mathcal{D}_n f\|_w \leq C n^{1-m} \|f\|_{m,w}, \tag{10.71}$$

for any $m \geq 2$ such that the norm $\|f\|_{m,w}$, introduced in (10.23), is finite. Recalling (10.19) and using (10.27) yields

$$(\mathcal{D}_n f)(\bar{x}_i) = \sum_{j=0}^{n} f(\bar{x}_j) \bar{l}_j'(\bar{x}_i), \qquad i = 0, \ldots, n, \tag{10.72}$$

so that the pseudo-spectral derivative at the interpolation nodes can be computed knowing only the nodal values of f and of $\bar{l}_j'$. These values can be computed once for all and stored in a matrix $D \in \mathbb{R}^{(n+1)\times(n+1)}$: $D_{ij} = \bar{l}_j'(\bar{x}_i)$ for $i, j = 0, ..., n$, called a *pseudo-spectral differentiation matrix*.

Relation (10.72) can thus be cast in matrix form as $\mathbf{f}' = D\mathbf{f}$, letting $\mathbf{f} = [f(\bar{x}_i)]$ and $\mathbf{f}' = [(\mathcal{D}_n f)(\bar{x}_i)]$ for $i = 0, ..., n$.
The entries of D have the following explicit form (see [CHQZ88], p. 69)

$$D_{lj} = \begin{cases} \dfrac{d_l}{d_j} \dfrac{(-1)^{l+j}}{\bar{x}_l - \bar{x}_j}, & l \neq j, \\ \dfrac{-\bar{x}_j}{2(1 - \bar{x}_j^2)}, & 1 \leq l = j \leq n-1, \\ -\dfrac{2n^2+1}{6}, & l = j = 0, \\ \dfrac{2n^2+1}{6}, & l = j = n, \end{cases} \tag{10.73}$$

where the coefficients d_l have been defined in Section 10.3 (see also Example 5.13 concerning the approximation of the multiple eigenvalue $\lambda = 0$ of D). To compute the pseudo-spectral derivative of a function f over the generic interval $[a, b]$, we only have to resort to the change of variables considered in Remark 10.3.

The second-order pseudo-spectral derivative can be computed as the product of the matrix D and the vector $\mathbf{f}'$, that is, $\mathbf{f}'' = D\mathbf{f}'$, or by directly applying matrix D^2 to the vector $\mathbf{f}$.

10.11 Transforms and Their Applications

In this section we provide a short introduction to the most relevant integral transforms and discuss their basic analytical and numerical properties.

10.11.1 The Fourier Transform

Definition 10.1 Let $\mathrm{L}^1(\mathbb{R})$ denote the space of real or complex functions defined on the real line such that

$$\int_{-\infty}^{\infty} |f(t)|\, dt < +\infty.$$

For any $f \in \mathrm{L}^1(\mathbb{R})$ its Fourier transform is a complex-valued function $F = \mathcal{F}[f]$ defined as

$$F(\nu) = \int_{-\infty}^{\infty} f(t) e^{-i2\pi\nu t}\, dt.$$

■

Should the independent variable t denote time, then ν would have the meaning of frequency. Thus, the Fourier transform is a mapping that to a function of time (typically, real-valued) associates a complex-valued function of frequency.
The following result provides the conditions under which an inversion formula exists that allows us to recover the function f from its Fourier transform F (for the proof see [Rud83], p. 199).

Property 10.3 (inversion theorem) *Let f be a given function in $\mathrm{L}^1(\mathbb{R})$, $F \in \mathrm{L}^1(\mathbb{R})$ be its Fourier transform and g be the function defined by*

$$g(t) = \int_{-\infty}^{\infty} F(\nu) e^{i2\pi\nu t}\, d\nu, \qquad t \in \mathbb{R}. \tag{10.74}$$

Then $g \in C^0(\mathbb{R})$, with $\lim_{|x|\to\infty} g(x) = 0$, and $f(t) = g(t)$ almost everywhere in $\mathbb{R}$ (i.e., for any t unless possibly a set of zero measure).

The integral at right hand side of (10.74) is to be meant in the *Cauchy principal value sense*, i.e., we let

$$\int_{-\infty}^{\infty} F(\nu) e^{i2\pi\nu t}\, d\nu = \lim_{a\to\infty} \int_{-a}^{a} F(\nu) e^{i2\pi\nu t}\, d\nu$$

and we call it the *inverse Fourier transform* or *inversion formula of the Fourier transform*. This mapping that associates to the complex function F the generating function f will be denoted by $\mathcal{F}^{-1}[F]$, i.e., $F = \mathcal{F}[f]$ iff $f = \mathcal{F}^{-1}[F]$.
Let us briefly summarize the main properties of the Fourier transform and its inverse.

1. $\mathcal{F}$ and $\mathcal{F}^{-1}$ are *linear* operators, i.e.

$$\begin{aligned} &\mathcal{F}[\alpha f + \beta g] = \alpha\mathcal{F}[f] + \beta\mathcal{F}[g], && \forall \alpha, \beta \in \mathbb{C}, \\ &\mathcal{F}^{-1}[\alpha F + \beta G] = \alpha\mathcal{F}^{-1}[F] + \beta\mathcal{F}^{-1}[G], && \forall \alpha, \beta \in \mathbb{C}; \end{aligned} \tag{10.75}$$

2. *scaling*: if α is any nonzero real number and f_α is the function $f_\alpha(t) = f(\alpha t)$, then

$$\mathcal{F}[f_\alpha] = \frac{1}{|\alpha|} F_{\frac{1}{\alpha}}$$

where $F_{\frac{1}{\alpha}}(\nu) = F(\nu/\alpha)$;

3. *duality*: let $f(t)$ be a given function and $F(\nu)$ be its Fourier transform. Then the function $g(t) = F(-t)$ has a Fourier transform given by $f(\nu)$. Thus, once an associated function-transform pair is found, another dual pair is automatically generated. An application of this property is provided by the pair $r(t)$-$\mathcal{F}[r]$ in Example 10.5;

4. *parity*: if $f(t)$ is a real even function then $F(\nu)$ is real and even, while if $f(t)$ is a real and odd function then $F(\nu)$ is imaginary and odd. This property allows one to work only with nonnegative frequencies;

5. *convolution and product*: for any given functions $f, g \in \mathrm{L}^1(\mathbb{R})$, we have

$$\mathcal{F}[f * g] = \mathcal{F}[f]\mathcal{F}[g], \quad \mathcal{F}[fg] = F * G, \tag{10.76}$$

where the *convolution integral* of two functions ϕ and ψ is given by

$$(\phi * \psi)(t) = \int_{-\infty}^{\infty} \phi(\tau)\psi(t - \tau)\, d\tau. \tag{10.77}$$

Example 10.5 We provide two examples of the computation of the Fourier transforms of functions that are typically encountered in signal processing.
Let us first consider the *square wave (or rectangular)* function $r(t)$ defined as

$$r(t) = \begin{cases} A & \text{if } -\frac{T}{2} \le t \le \frac{T}{2}, \\ 0 & \text{otherwise,} \end{cases}$$

where T and A are two given positive numbers. Its Fourier transform $\mathcal{F}[r]$ is the function

$$F(\nu) = \int_{-T/2}^{T/2} Ae^{-i2\pi\nu t}\, dt = AT\frac{\sin(\pi\nu T)}{\pi\nu T}, \qquad \nu \in \mathbb{R}$$

where AT is the area of the rectangular function.

Let us consider the *sawtooth* function

$$s(t) = \begin{cases} \dfrac{2At}{T} & \text{if } -\frac{T}{2} \le t \le \frac{T}{2}, \\ 0 & \text{otherwise,} \end{cases}$$

whose DFT is shown in Figure 10.3 and whose Fourier transform $\mathcal{F}[s]$ is the function

$$F(\nu) = i\,\frac{AT}{\pi\nu T}\left[\cos(\pi\nu T) - \frac{\sin(\pi\nu T)}{\pi\nu T}\right], \qquad \nu \in \mathbb{R}$$

and is purely imaginary since s is an odd real function. Notice also that the functions r and s have a finite support whereas their transforms have an infinite support (see Figure 10.7). In signal theory this corresponds to saying that the transform has an infinite *bandwidth*. •

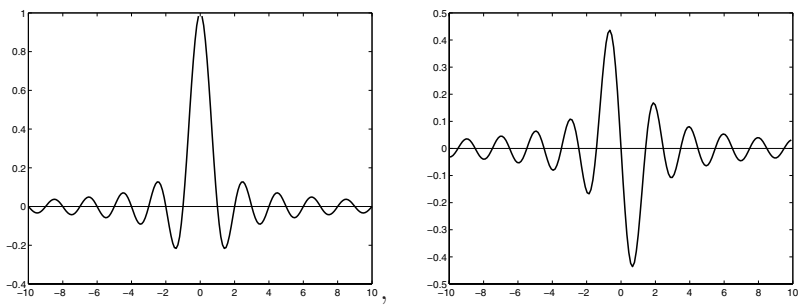

FIGURE 10.7. Fourier transforms of the rectangular (left) and the sawtooth (right) functions

Example 10.6 The Fourier transform of a sinusoidal function is of paramount interest in signal and communication systems. To start with, consider the constant function $f(t) = A$ for a given $A \in \mathbb{R}$. Since it has an *infinite* time duration its Fourier transform $\mathcal{F}[A]$ is the function

$$F(\nu) = \lim_{a\to\infty} \int_{-a}^{a} Ae^{-i2\pi\nu t}\, dt = A \lim_{a\to\infty} \frac{\sin(2\pi\nu a)}{\pi\nu},$$

where the integral above is again the Cauchy principal value of the corresponding integral over $(-\infty, \infty)$. It can be proved that the limit exists and is unique *in the sense of distributions* (see Section 12.4) yielding

$$F(\nu) = A\delta(\nu), \tag{10.78}$$

where δ is the so-called Dirac mass, i.e., a distribution that satisfies

$$\int_{-\infty}^{\infty} \delta(\xi)\phi(\xi)\, d\xi = \phi(0) \tag{10.79}$$

for any function ϕ continuous at the origin. From (10.78) we see that the transform of a function with infinite time duration has a finite bandwidth.

Let us now consider the computation of the Fourier transform of the function $f(t) = A\cos(2\pi\nu_0 t)$ where ν_0 is a fixed frequency. Recalling Euler's formula

$$\cos(\theta) = \frac{e^{i\theta} + e^{-i\theta}}{2}, \qquad \theta \in \mathbb{R},$$

and applying (10.78) twice we get

$$\mathcal{F}[A\cos(2\pi\nu_0 t)] = \frac{A}{2}\delta(\nu - \nu_0) + \frac{A}{2}\delta(\nu + \nu_0),$$

which shows that the spectrum of a sinusoidal function with frequency ν_0 is centred around $\pm\nu_0$ (notice that the transform is even and real since the same holds for the function $f(t)$). •

It is worth noting that in real-life there do not exist functions (i.e. signals) with infinite duration or bandwidth. Actually, if $f(t)$ is a function whose value may be considered as "negligible" outside of some interval (t_a, t_b), then we can assume that the *effective duration* of f is the length $\Delta t = t_b - t_a$. In a similar manner, if $F(\nu)$ is the Fourier transform of f and it happens that $F(\nu)$ may be considered as "negligible" outside of some interval (ν_a, ν_b), then the *effective bandwidth* of f is $\Delta\nu = \nu_b - \nu_a$. Referring to Figure 10.7, we clearly see that the effective bandwidth of the rectangular function can be taken as $(-10, 10)$.

10.11.2 (Physical) Linear Systems and Fourier Transform

Mathematically speaking, a physical linear system (LS) can be regarded as a linear operator S that enjoys the linearity property (10.75). Denoting by $i(t)$ and $u(t)$ an admissible input function for S and its corresponding output function respectively, the LS can be represented as $u(t) = S(i(t))$ or $S : i \to u$. A special category of LS are the so-called *shift invariant* (or time-invariant) linear systems (ILS) which satisfy the property

$$S(i(t - t_0)) = u(t - t_0), \qquad \forall t_0 \in \mathbb{R}$$

and for any admissible input function i.

Let S be an ILS system and let f and g be two admissible input functions for S with $w = S(g)$. An immediate consequence of the linearity and shift-invariance is that

$$S((f * g)(t)) = (f * S(g))(t) = (f * w)(t) \tag{10.80}$$

where $*$ is the convolution operator defined in (10.77).

Assume we take as input function the *impulse function* $\delta(t)$ introduced in the previous section and denote by $h(t) = S(\delta(t))$ the corresponding output through S (usually referred to as the system *impulse response function*). Property (10.79) implies that for any function ϕ, $(\phi * \delta)(t) = \phi(t)$, so that, recalling (10.80) and taking $\phi(t) = i(t)$ we have

$$u(t) = S(i(t)) = S(i * \delta)(t) = (i * S(\delta))(t) = (i * h)(t).$$

Thus, S can be completely described through its impulse response function. Equivalently, we can pass to the frequency domain by means of the first relation in (10.76) obtaining

$$U(\nu) = I(\nu)H(\nu), \tag{10.81}$$

where I, U and H are the Fourier transforms of $i(t)$, $u(t)$ and $h(t)$, respectively; H is the so-called system *transfer function*.

Relation (10.81) plays a central role in the analysis of linear time-invariant systems as it is simpler to deal with the system transfer function than with the corresponding impulse response function, as demonstrated in the following example.

Example 10.7 (ideal low-pass filter) An ideal low-pass filter is an ILS characterized by the transfer function

$$H(\nu) = \begin{cases} 1, & \text{if } |\nu| \leq \nu_0/2, \\ 0, & \text{otherwise.} \end{cases}$$

Using the duality property, the impulse response function $\mathcal{F}^{-1}[H]$ is

$$h(t) = \nu_0 \frac{\sin(\pi\nu_0 t)}{\pi\nu_0 t}.$$

Given an input signal $i(t)$ with Fourier transform $I(\nu)$, the corresponding output $u(t)$ has a spectrum given by (10.81)

$$I(\nu)H(\nu) = \begin{cases} I(\nu), & \text{if } |\nu| \leq \nu_0/2, \\ 0 & \text{otherwise.} \end{cases}$$

The effect of the filter is to cut off the input frequencies that lie outside the *window* $|\nu| \leq \nu_0/2$. •

The input/output functions $i(t)$ and $u(t)$ usually denote *signals* and the linear system described by $H(\nu)$ is typically a communication system. Therefore, as pointed out at the end of Section 10.11.1, we are legitimated in assuming that both $i(t)$ and $u(t)$ have a finite effective duration. In

particular, referring to $i(t)$ we suppose $i(t) = 0$ if $t \notin [0, T_0)$. Then, the computation of the Fourier transform of $i(t)$ yields

$$I(\nu) = \int_0^{T_0} i(t) e^{-i2\pi\nu t}\, dt.$$

Letting $\Delta t = T_0/n$ for $n \geq 1$ and approximating the integral above by the composite trapezoidal formula (9.14), we get

$$\tilde{I}(\nu) = \Delta t \sum_{k=0}^{n-1} i(k\Delta t) e^{-i2\pi\nu k\Delta t}.$$

It can be proved (see, e.g., [Pap62]) that $\tilde{I}(\nu)/\Delta t$ is the Fourier transform of the so-called *sampled signal*

$$i_s(t) = \sum_{k=-\infty}^{\infty} i(k\Delta t)\delta(t - k\Delta t),$$

where $\delta(t - k\Delta t)$ is the Dirac mass at $k\Delta t$. Then, using the convolution and the duality properties of the Fourier transform, we get

$$\tilde{I}(\nu) = \sum_{j=-\infty}^{\infty} I\left(\nu - \frac{j}{\Delta t}\right), \tag{10.82}$$

which amounts to replacing $I(\nu)$ by its periodic repetition with period $1/\Delta t$. Let $\mathcal{J}_{\Delta t} = [-\frac{1}{2\Delta t}, \frac{1}{2\Delta t}]$; then, it suffices to compute (10.82) for $\nu \in \mathcal{J}_{\Delta t}$. This can be done numerically by introducing a uniform discretization of $\mathcal{J}_{\Delta t}$ with frequency step $\nu_0 = 1/(m\Delta t)$ for $m \geq 1$. By doing so, the computation of $\tilde{I}(\nu)$ requires evaluating the following $m+1$ discrete Fourier transforms (DFT)

$$\tilde{I}(j\nu_0) = \Delta t \sum_{k=0}^{n-1} i(k\Delta t) e^{-i2\pi j\nu_0 k\Delta t}, \quad j = -\tfrac{m}{2}, \ldots, \tfrac{m}{2}.$$

For an efficient computation of each DFT in the formula above it is crucial to use the FFT algorithm described in Section 10.9.2.

10.11.3 The Laplace Transform

The Laplace transform can be employed to solve ordinary differential equations with constant coefficients as well as partial differential equations.

Definition 10.2 Let $f \in \mathrm{L}^1_{loc}([0,\infty))$ i.e., $f \in \mathrm{L}^1([0,T])$ for any $T > 0$. Let $s = \sigma + i\omega$ be a complex variable. The Laplace integral of f is defined

as

$$\int_0^\infty f(t)e^{-st}\,dt = \lim_{T\to\infty}\int_0^T f(t)e^{-st}\,dt.$$

If this integral exists for some s, it turns out to be a function of s; then, the *Laplace transform* $\mathcal{L}[f]$ of f is the function

$$L(s) = \int_0^\infty f(t)e^{-st}\,dt.$$

■

The following relation between Laplace and Fourier transforms holds

$$L(s) = F(e^{-\sigma t}\tilde{f}(t)),$$

where $\tilde{f}(t) = f(t)$ if $t \geq 0$ while $\tilde{f}(t) = 0$ if $t < 0$.

Example 10.8 The Laplace transform of the unit step function $f(t) = 1$ if $t > 0$, $f(t) = 0$ otherwise, is given by

$$L(s) = \int_0^\infty e^{-st}\,dt = \frac{1}{s}.$$

We notice that the Laplace integral exists if $\sigma > 0$. •

In Example 10.8 the convergence region of the Laplace integral is the half-plane $\{\mathrm{Re}(s) > 0\}$ of the complex field. This property is quite general, as stated by the following result.

Property 10.4 *If the Laplace transform exists for $s = \bar{s}$ then it exists for all s with $\mathrm{Re}(s) > \mathrm{Re}(\bar{s})$. Moreover, let E be the set of the real parts of s such that the Laplace integral exists and denote by λ the infimum of E. If λ happens to be finite, the Laplace integral exists in the half-plane $\mathrm{Re}(s) > \lambda$. If $\lambda = -\infty$ then it exists for all $s \in \mathbb{C}$; λ is called the abscissa of convergence.*

We recall that the Laplace transform enjoys properties completely analogous to those of the Fourier transform. The inverse Laplace transform is denoted formally as $\mathcal{L}^{-1}$ and is such that

$$f(t) = \mathcal{L}^{-1}[L(s)].$$

Example 10.9 Let us consider the ordinary differential equation $y'(t)+ay(t) = g(t)$ with $y(0) = y_0$. Multiplying by e^{st}, integrating between 0 and ∞ and passing to the Laplace transform, yields

$$sY(s) - y_0 + aY(s) = G(s). \tag{10.83}$$

Should $G(s)$ be easily computable, (10.83) would furnish $Y(s)$ and then, by applying the inverse Laplace transform, the generating function $y(t)$. For instance, if $g(t)$ is the unit step function, we obtain

$$y(t) = \mathcal{L}^{-1}\left\{\frac{1}{a}\left[\frac{1}{s} - \frac{1}{s+a}\right] + \frac{y_0}{s+a}\right\} = \frac{1}{a}(1 - e^{-at}) + y_0 e^{-at}.$$

•

For an extensive presentation and analysis of the Laplace transform see, e.g., [Tit37]. In the next section we describe a discrete version of the Laplace transform, known as the Z-transform.

10.11.4 The Z-Transform

Definition 10.3 Let f be a given function, defined for any $t \geq 0$, and $\Delta t > 0$ be a given time step. The function

$$Z(z) = \sum_{n=0}^{\infty} f(n\Delta t) z^{-n}, \qquad z \in \mathbb{C} \tag{10.84}$$

is called the Z-transform of the sequence $\{f(n\Delta t)\}$ and is denoted by $\mathcal{Z}[f(n\Delta t)]$. ■

The parameter Δt is the *sampling time step* of the sequence of samples $f(n\Delta t)$. The infinite sum (10.84) converges if

$$|z| > R = \limsup_{n\to\infty} \sqrt[n]{|f(n\Delta t)|}.$$

It is possible to deduce the Z-transform from the Laplace transform as follows. Denoting by $f_0(t)$ the piecewise constant function such that $f_0(t) = f(n\Delta t)$ for $t \in (n\Delta t, (n+1)\Delta t)$, the Laplace transform $\mathcal{L}[f_0]$ of f_0 is the function

$$\begin{aligned} L(s) &= \int_0^{\infty} f_0(t) e^{-st}\, dt = \sum_{n=0}^{\infty} \int_{n\Delta t}^{(n+1)\Delta t} e^{-st} f(n\Delta t)\, dt \\ &= \sum_{n=0}^{\infty} f(n\Delta t) \frac{e^{-ns\Delta t} - e^{-(n+1)s\Delta t}}{s} = \left(\frac{1 - e^{-s\Delta t}}{s}\right) \sum_{n=0}^{\infty} f(n\Delta t) e^{-ns\Delta t}. \end{aligned}$$

The *discrete Laplace transform* $\mathcal{Z}^d[f_0]$ of f_0 is the function

$$Z^d(s) = \sum_{n=0}^{\infty} f(n\Delta t)e^{-ns\Delta t}.$$

Then, the Z-transform of the sequence $\{f(n\Delta t),\ n = 0, \dots, \infty\}$ coincides with the discrete Laplace transform of f_0 up to the change of variable $z = e^{-s\Delta t}$. The Z-transform enjoys similar properties (linearity, scaling, convolution and product) to those already seen in the continuous case.
The inverse Z-transform is denoted by $\mathcal{Z}^{-1}$ and is defined as

$$f(n\Delta t) = \mathcal{Z}^{-1}[Z(z)].$$

The practical computation of $\mathcal{Z}^{-1}$ can be carried out by resorting to classical techniques of complex analysis (for example, using the Laurent formula or the Cauchy theorem for residual integral evaluation) coupled with an extensive use of tables (see, e.g., [Pou96]).

10.12 The Wavelet Transform

This technique, originally developed in the area of signal processing, has successively been extended to many different branches of approximation theory, including the solution of differential equations. It is based on the so-called wavelets, which are functions generated by an elementary wavelet through traslations and dilations. We shall limit ourselves to a brief introduction of univariate wavelets and their transform in both the continuous and discrete cases referring to [DL92], [Dau88] and to the references cited therein for a detailed presentation and analysis.

10.12.1 The Continuous Wavelet Transform

Any function

$$h_{s,\tau}(t) = \frac{1}{\sqrt{s}} h\left(\frac{t-\tau}{s}\right), \qquad t \in \mathbb{R} \tag{10.85}$$

that is obtained from a reference function $h \in \mathrm{L}^2(\mathbb{R})$ by means of traslations by a *traslation factor* τ and dilations by a positive *scaling factor* s is called a *wavelet*. The function h is called an *elementary wavelet*.
Its Fourier transform, written in terms of $\omega = 2\pi\nu$, is

$$H_{s,\tau}(\omega) = \sqrt{s}H(s\omega)e^{-i\omega\tau}, \tag{10.86}$$

where i denotes the imaginary unit and $H(\omega)$ is the Fourier transform of the elementary wavelet. A dilation t/s $(s > 1)$ in the real domain produces

therefore a contraction $s\omega$ in the frequency domain. Therefore, the factor $1/s$ plays the role of the frequency ν in the Fourier transform (see Section 10.11.1). In wavelets theory s is usually referred to as the *scale*. Formula (10.86) is known as the filter of the wavelet transform.

Definition 10.4 Given a function $f \in \mathrm{L}^2(\mathbb{R})$, its continuous *wavelet transform* $W_f = \mathcal{W}[f]$ is a decomposition of $f(t)$ onto a wavelet basis $\{h_{s,\tau}(t)\}$, that is

$$W_f(s,\tau) = \int_{-\infty}^{\infty} f(t)\bar{h}_{s,\tau}(t)\ dt, \tag{10.87}$$

where the overline bar denotes complex conjugate. ■

When t denotes the time-variable, the wavelet transform of $f(t)$ is a function of the two variables s (scale) and τ (time shift); as such, it is a representation of f in the time-scale space and is usually referred to as *time-scale joint representation* of f. The time-scale representation is the analogue of the time-frequency representation introduced in the Fourier analysis. This latter representation has an intrinsic limitation: the product of the resolution in time Δt and the resolution in frequency $\Delta\omega$ must satisfy the following constraint (Heisenberg inequality)

$$\Delta t \Delta\omega \geq \frac{1}{2} \tag{10.88}$$

which is the counterpart of the Heisenberg uncertainty principle in quantum mechanics. This inequality states that a signal cannot be represented as a point in the time-frequency space. We can only determine its position within a rectangle of area $\Delta t \Delta\omega$ in the time-frequency space.

The wavelet transform (10.87) can be rewritten in terms of the Fourier transform $F(\omega)$ of f as

$$W_f(s,\tau) = \frac{\sqrt{s}}{2\pi} \int_{-\infty}^{\infty} F(\omega)\bar{H}(s\omega)e^{i\omega\tau}\ d\omega,$$

which shows that the wavelets transform is a bank of wavelet filters characterized by different scales. More precisely, if the scale is small the wavelet is concentrated in time and the wavelet transform provides a detailed description of $f(t)$ (which is the signal). Conversely, if the scale is large, the wavelet transform is able to resolve only the large-scale details of f. Thus, the wavelet transform can be regarded as a bank of *multiresolution filters*.

The theoretical properties of this transform do not depend on the particular elementary wavelet that is considered. Hence, specific bases of wavelets can be derived for specific applications. Some examples of elementary wavelets are reported below.

Example 10.10 (Haar wavelets) These functions can be obtained by choosing as the elementary wavelet the Haar function defined as

$$h(x) = \begin{cases} 1 & \text{if } x \in (0, \frac{1}{2}), \\ -1 & \text{if } x \in (\frac{1}{2}, 1), \\ 0 & \text{otherwise.} \end{cases}$$

Its Fourier transform is the complex-valued function

$$H(\omega) = 4ie^{-i\omega/2} \left(1 - \cos(\frac{\omega}{2})\right) /\omega,$$

which has symmetric module with respect to the origin (see Figure 10.8). The bases that are obtained from this wavelet are not used in practice due to their ineffective localization properties in the frequency domain. •

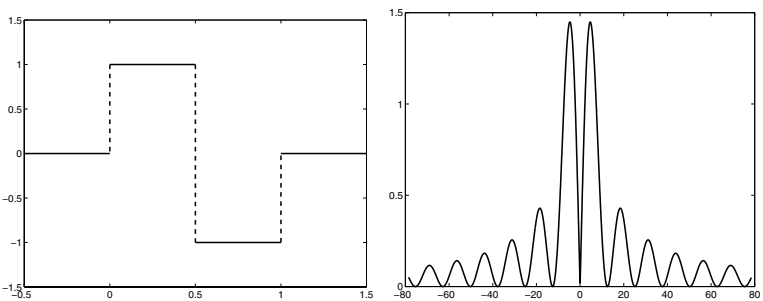

FIGURE 10.8. The Haar wavelet (left) and the module of its Fourier transform (right)

Example 10.11 (Morlet wavelets) The *Morlet wavelet* is defined as follows (see [MMG87])

$$h(x) = e^{i\omega_0 x} e^{-x^2/2}.$$

Thus, it is a complex-valued function whose real part has a real positive Fourier transform, symmetric with respect to the origin, given by

$$H(\omega) = \sqrt{\pi} \left[e^{-(\omega-\omega_0)^2/2} + e^{-(\omega+\omega_0)^2/2}\right].$$

•

We point out that the presence of the dilation factor allows for the wavelets to easily handle possible discontinuities or singularities in f. Indeed, using the multi-resolution analysis, the signal, properly divided into frequency bandwidths, can be processed at each frequency by suitably tuning up the scale factor of the wavelets.

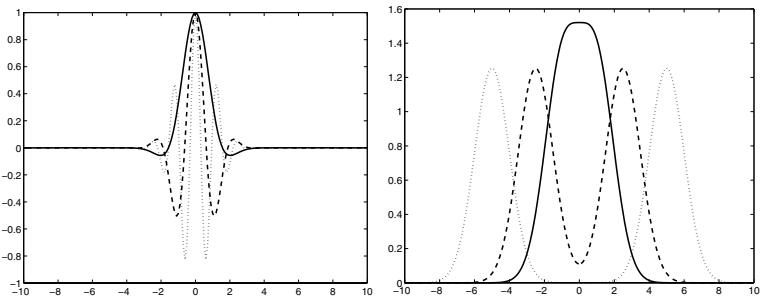

FIGURE 10.9. The real part of the Morlet wavelet (left) and the real part of the corresponding Fourier transforms (right) for $\omega_0 = 1$ (solid line), $\omega_0 = 2.5$ (dashed line) and $\omega_0 = 5$ (dotted line)

Recalling what was already pointed out in Section 10.11.1, the time localization of the wavelet gives rise to a filter with infinite bandwidth. In particular, defining the bandwidth $\Delta\omega$ of the wavelet filter as

$$\Delta\omega = \left(\int_{-\infty}^{\infty} \omega^2 |H(\omega)|^2 \, d\omega / \int_{-\infty}^{\infty} |H(\omega)|^2 \, d\omega \right)^2 ,$$

then the bandwidth of the wavelet filter with scale equal to s is

$$\Delta\omega_s = \left(\int_{-\infty}^{\infty} \omega^2 |H(s\omega)|^2 \, d\omega / \int_{-\infty}^{\infty} |H(s\omega)|^2 \, d\omega \right)^2 = \frac{1}{s} \Delta\omega.$$

Consequently, the *quality factor* Q of the wavelet filter, defined as the inverse of the bandwidth of the filter, is independent of s since

$$Q = \frac{1/s}{\Delta\omega_s} = \Delta\omega$$

provided that (10.88) holds. At low frequencies, corresponding to large values of s, the wavelet filter has a small bandwidth and a large temporal width (called *window*) with a low resolution. Conversely, at high frequencies the filter has a large bandwidth and a small temporal window with a high resolution. Thus, the resolution furnished by the wavelet analysis increases with the frequency of the signal. This property of *adaptivity* makes the wavelets a crucial tool in the analysis of unsteady signals or signals with fast transients for which the standard Fourier analysis turns out to be ineffective.

10.12.2 Discrete and Orthonormal Wavelets

The continuous wavelet transform maps a function of one variable into a bidimensional representation in the time-scale domain. In many applications

this description is excessively rich. Resorting to the discrete wavelets is an attempt to represent a function using a finite (and small) number of parameters.

A *discrete wavelet* is a continuous wavelet that is generated by using discrete scale and translation factors. For $s_0 > 1$, denote by $s = s_0^j$ the scale factors; the dilation factors usually depend on the scale factors by setting $\tau = k\tau_0 s_0^j$, $\tau_0 \in \mathbb{R}$. The corresponding discrete wavelet is

$$h_{j,k}(t) = s_0^{-j/2} h(s_0^{-j}(t - k\tau_0 s_0^j)) = s_0^{-j/2} h(s_0^{-j}t - k\tau_0).$$

The scale factor s_0^j corresponds to the magnification or the resolution of the observation, while the translation factor τ_0 is the location where the observations are made. If one looks at very small details, the magnification must be large, which corresponds to large negative index j. In this case the step of translation is small and the wavelet is very concentrated around the observation point. For large and positive j, the wavelet is spread out and large translation steps are used.

The behavior of the discrete wavelets depends on the steps s_0 and τ_0. When s_0 is close to 1 and τ_0 is small, the discrete wavelets are close to the continuous ones. For a fixed scale s_0 the localization points of the discrete wavelets along the scale axis are logarithmic as $\log s = j \log s_0$. The choice $s_0 = 2$ corresponds to the dyadic sampling in frequency. The discrete time-step is $\tau_0 s_0^j$ and, typically, $\tau_0 = 1$. Hence, the time-sampling step is a function of the scale and along the time axis the localization points of the wavelet depend on the scale.

For a given function $f \in \mathrm{L}^1(\mathbb{R})$, the corresponding discrete wavelet transform is

$$W_f(j,k) = \int_{-\infty}^{\infty} f(t)\bar{h}_{j,k}(t)\, dt.$$

It is possible to introduce an orthonormal wavelet basis using discrete dilation and traslation factors, i.e.

$$\int_{-\infty}^{\infty} h_{i,j}\bar{h}_{k,l}(t)\, dt = \delta_{ik}\delta jl, \qquad \forall i,j,k,l \in \mathbb{Z}.$$

With an orthogonal wavelet basis, an arbitrary function f can be reconstructed by the expansion

$$f(t) = A \sum_{j,k\in\mathbb{Z}} W_f(j,k) h_{j,k}(t),$$

where A is a constant that does not depend on f.

As of the computational standpoint, the wavelet discrete transform can be implemented at even a cheaper cost than the FFT algorithm for computing the Fourier transform.

10.13 Applications

In this section we apply the theory of orthogonal polynomials to solve two problems arising in quantum physics. In the first example we deal with Gauss-Laguerre quadratures, while in the second case the Fourier analysis and the FFT are considered.

10.13.1 Numerical Computation of Blackbody Radiation

The monochromatic energy density $\mathcal{E}(\nu)$ of blackbody radiation as a function of frequency ν is expressed by the following law

$$\mathcal{E}(\nu) = \frac{8\pi h}{c^3} \frac{\nu^3}{e^{h\nu/K_B T} - 1},$$

where h is the Planck constant, c is the speed of light, K_B is the Boltzmann constant and T is the absolute temperature of the blackbody (see, for instance, [AF83]).

To compute the total density of monochromatic energy that is emitted by the blackbody (that is, the emitted energy per unit volume) we must evaluate the integral

$$E = \int_0^\infty \mathcal{E}(\nu) d\nu = \alpha T^4 \int_0^\infty \frac{x^3}{e^x - 1} dx,$$

where $x = h\nu/K_B T$ and $\alpha = (8\pi K_B^4)/(ch)^3 \simeq 1.16 \cdot 10^{-16}\,[J][K^{-4}][m^{-3}]$. We also let $f(x) = x^3/(e^x - 1)$ and $I(f) = \int_0^\infty f(x)dx$.

To approximate $I(f)$ up to a previously fixed absolute error $\leq \delta$, we compare method 1. introduced in Section 9.8.3 with Gauss-Laguerre quadratures.

In the case of method 1. we proceed as follows. For any $a > 0$ we let $I(f) = \int_0^a f(x)dx + \int_a^\infty f(x)dx$ and try to find a function ϕ such that

$$\int_a^\infty f(x)dx \leq \int_a^\infty \phi(x)dx \leq \frac{\delta}{2}, \tag{10.89}$$

the integral $\int_a^\infty \phi(x)dx$ being "easy" to compute. Once the value of a has been found such that (10.89) is fulfilled, we compute the integral $I_1(f) = \int_0^a f(x)dx$ using for instance the adaptive Cavalieri-Simpson formula introduced in Section 9.7.2 and denoted in the following by AD.

A natural choice of a bounding function for f is $\phi(x) = Kx^3 e^{-x}$, for a suitable constant $K > 1$. Thus, we have $K \geq e^x/(e^x - 1)$, for any $x > 0$, that is, letting $x = a$, $K = e^a/(e^a - 1)$. Substituting back into (10.89) yields

$$\int_a^\infty f(x)dx \leq \frac{a^3 + 3a^2 + 6a + 6}{e^a - 1} = \eta(a) \leq \frac{\delta}{2}.$$

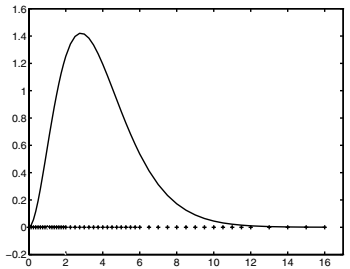

FIGURE 10.10. Distribution of quadrature nodes and graph of the integrand function

Letting $\delta = 10^{-3}$, we see that (10.89) is satisfied by taking $a \simeq 16$. Program 77 for computing $I_1(f)$ with the AD method, setting `hmin`$=10^{-3}$ and `tol`$=5 \cdot 10^{-4}$, yields the approximate value $I_1 \simeq 6.4934$ with a number of (nonuniform) partitions equal to 25.

The distribution of the quadrature nodes produced by the adaptive algorithm is plotted in Figure 10.10. Globally, using method 1. yields an approximation of $I(f)$ equal to 6.4984. Table 10.1 shows, for sake of comparison, some approximate values of $I(f)$ obtained using the Gauss-Laguerre formulae with the number of nodes varying between 2 to 20. Notice that, taking $n = 4$ nodes, the accuracy of the two computational procedures is roughly equivalent.

n	$I_n(f)$
2	6.413727469517582
3	6.481130171540022
4	6.494535639802632
5	6.494313365790864
10	6.493939967652101
15	6.493939402671590
20	6.493939402219742

TABLE 10.1. Approximate evaluation of $I(f) = \int_0^\infty x^3/(e^x - 1)dx$ with Gauss-Laguerre quadratures

10.13.2 Numerical Solution of Schrödinger Equation

Let us consider the following differential equation arising in quantum mechanics known as the *Schrödinger equation*

$$i\frac{\partial \psi}{\partial t} = -\frac{\hbar}{2m}\frac{\partial^2 \psi}{\partial x^2}, \qquad x \in \mathbb{R} \qquad t > 0. \tag{10.90}$$

The symbols i and $\hbar$ denote the imaginary unit and the reduced Planck constant, respectively. The complex-valued function $\psi = \psi(x, t)$, the solu-

tion of (10.90), is called a *wave function* and the quantity $|\psi(x,t)|^2$ defines the probability density in the space x of a free electron of mass m at time t (see [FRL55]).
The corresponding Cauchy problem may represent a physical model for describing the motion of an electron in a cell of an infinite lattice (for more details see, e.g., [AF83]).

Consider the initial condition $\psi(x,0) = w(x)$, where w is the step function that takes the value $1/\sqrt{2b}$ for $|x| \leq b$ and is zero for $|x| > b$, with $b = a/5$, and where $2a$ represents the inter-ionic distance in the lattice. Therefore, we are searching for periodic solutions, with period equal to $2a$.

Solving problem (10.90) can be carried out using Fourier analysis as follows. We first write the Fourier series of w and ψ (for any $t > 0$)

$$\begin{aligned} w(x) &= \sum_{k=-N/2}^{N/2-1} \widehat{w}_k e^{i\pi kx/a}, \qquad \widehat{w}_k = \frac{1}{2a}\int_{-a}^{a} w(x)e^{-i\pi kx/a}dx, \\ \psi(x,t) &= \sum_{k=-N/2}^{N/2-1} \widehat{\psi}_k(t)e^{i\pi kx/a}, \quad \widehat{\psi}_k(t) = \frac{1}{2a}\int_{-a}^{a} \psi(x,t)e^{-i\pi kx/a}dx. \end{aligned} \tag{10.91}$$

Then, we substitute back (10.91) into (10.90), obtaining the following Cauchy problem for the Fourier coefficients $\widehat{\psi}_k$, for $k = -N/2, \ldots, N/2-1$

$$\begin{cases} \widehat{\psi}_k'(t) = -i\dfrac{\hbar}{2m}\left(\dfrac{k\pi}{a}\right)^2 \widehat{\psi}_k(t), \\ \widehat{\psi}_k(0) = \widetilde{\widehat{w}}_k. \end{cases} \tag{10.92}$$

The coefficients $\{\widetilde{\widehat{w}}_k\}$ have been computed by regularizing the coefficients $\{\widehat{w}_k\}$ of the step function w using the Lanczos smoothing (10.56) in order to avoid the Gibbs phenomenon arising around the discontinuities of w (see Section 10.9.1).
After solving (10.92), we finally get, recalling (10.91), the following expression for the wave function

$$\psi_N(x,t) = \sum_{k=-N/2}^{N/2-1} \widetilde{\widehat{w}}_k e^{-iE_k t/\hbar} e^{i\pi kx/a}, \tag{10.93}$$

where the coefficients $E_k = (k^2\pi^2\hbar^2)/(2ma^2)$ represent, from the physical standpoint, the energy levels that the electron may assume in its motion within the potential well.
To compute the coefficients $\widehat{w}_k$ (and, as a consequence, $\widetilde{\widehat{w}}_k$), we have used the MATLAB intrinsic function `fft` (see Section 10.9.2), employing $N = 2^6 = 64$ points and letting $a = 10\ \mathring{A} = 10^{-9}[m]$. Time analysis has been carried out up to $T = 10\,[s]$, with time steps of $1\,[s]$; in all the reported

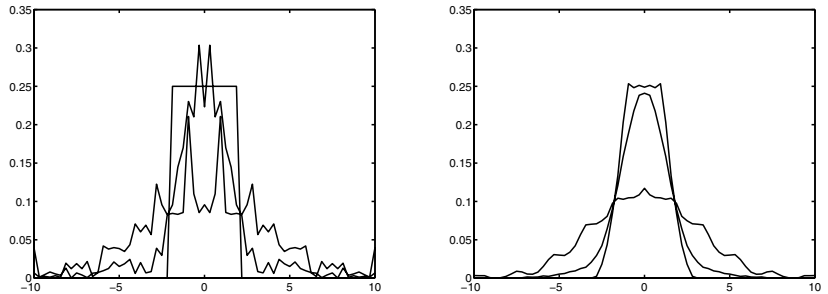

FIGURE 10.11. Probability density $|\psi(x,t)|^2$ at $t = 0,\ 2,\ 5\,[s]$, corresponding to a step function as initial datum: solution without filtering (left), with Lanczos filtering (right)

graphs, the x-axis is measured in $[\mathring{A}]$, while the y-axes are respectively in units of $10^5\,[m^{-1/2}]$ and $10^{10}\,[m^{-1}]$.

In Figure 10.11 we draw the probability density $|\psi(x,t)|^2$ at $t = 0,\ 2$ and $5\,[s]$. The result obtained without the regularizing procedure above is shown on the left, while the same calculation with the "filtering" of the Fourier coefficients is reported on the right. The second plot demonstrates the smoothing effect on the solution by the regularization, at the price of a slight enlargement of the step-like initial probability distribution.

Finally, it is interesting to apply Fourier analysis to solve problem (10.90) starting from a smooth initial datum. For this, we choose an initial probability density w of Gaussian form such that $\|w\|_2 = 1$. The solution $|\psi(x,t)|^2$, this time computed without regularization, is shown in Figure 10.12, at $t = 0,\ 2,\ 5,\ 7,\ 9[s]$. Notice the absence of spurious oscillations with respect to the previous case.

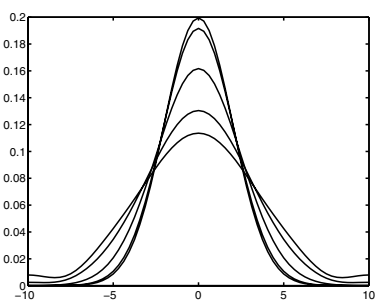

FIGURE 10.12. Probability density $|\psi(x,t)|^2$ at $t = 0,\ 2,\ 5,\ 7,\ 9[s]$, corresponding to an initial datum with Gaussian form

10.14 Exercises

1. Prove the three-term relation (10.11).
   [*Hint*: set $x = \cos(\theta)$, for $0 \le \theta \le \pi$.]

2. Prove (10.31).
   [*Hint*: first prove that $\|v_n\|_n = (v_n, v_n)^{1/2}$, $\|T_k\|_n = \|T_k\|_w$ for $k < n$ and $\|T_n\|_n^2 = 2\|T_n\|_w^2$ (see [QV94], formula (4.3.16)). Then, the thesis follows from (10.29) multiplying by T_l $(l \neq k)$ and taking $(\cdot,\cdot)_n$.]

3. Prove (10.24) after showing that $\|(f - \Pi_n^{GL} f)'\|_\omega \le Cn^{1-s}\|f\|_{s,\omega}$.
   [*Hint*: use the Gagliardo-Nirenberg inequality

$$\max_{-1\le x\le 1} |f(x)| \le \|f\|^{1/2}\|f'\|^{1/2}$$

   valid for any $f \in \mathrm{L}^2$ with $f' \in \mathrm{L}^2$. Next, use the relation that has been just shown to prove (10.24).]

4. Prove that the discrete seminorm $\|f\|_n = (f, f)_n^{1/2}$ is a norm for $\mathbb{P}_n$.

5. Compute weights and nodes of the following quadrature formulae

$$\int_a^b w(x)f(x)dx = \sum_{i=0}^{n} \omega_i f(x_i),$$

   in such a way that the order is maximum, setting

$$\begin{array}{lll} \omega(x) = \sqrt{x}, & a = 0, \quad b = 1, & n = 1; \\ \omega(x) = 2x^2 + 1, & a = -1, \quad b = 1, & n = 0; \\ \omega(x) = \left\{ \begin{array}{ll} 2 & \text{if } 0 < x \le 1, \\ 1 & \text{if } -1 \le x \le 0 \end{array} \right. & a = -1, \quad b = 1, \quad n = 1. \end{array}$$

   [*Solution*: for $\omega(x) = \sqrt{x}$, the nodes $x_1 = \frac{5}{9} + \frac{2}{9}\sqrt{10/7}$, $x_2 = \frac{5}{9} - \frac{2}{9}\sqrt{10/7}$ are obtained, from which the weights can be computed (order 3); for $\omega(x) = 2x^2 + 1$, we get $x_1 = 3/5$ and $\omega_1 = 5/3$ (order 1); for $\omega(x) = 2x^2 + 1$, we have $x_1 = \frac{1}{22} + \frac{1}{22}\sqrt{155}$, $x_2 = \frac{1}{22} - \frac{1}{22}\sqrt{155}$ (order 3).]

6. Prove (10.40).
   [*Hint*: notice that $(\Pi_n^{GL} f, L_j)_n = \sum_k f_k^*(L_k, L_j)_n = \ldots$, distinguishing the case $j < n$ from the case $j = n$.]

7. Show that $|||\cdot|||$, defined in (10.45), is an essentially strict seminorm.
   [*Solution* : use the Cauchy-Schwarz inequality (1.14) to check that the triangular inequality is satisfied. This proves that $|||\cdot|||$ is a seminorm. The second part of the exercise follows after a direct computation.]

8. Consider in an interval $[a, b]$ the nodes

$$x_j = a + \left(j - \frac{1}{2}\right)\left(\frac{b-a}{m}\right) \quad j = 1, 2, \ldots, m$$

for $m \geq 1$. They are the midpoints of m equally spaced intervals in $[a, b]$. Let f be a given function; prove that the least-squares polynomial r_n with respect to the weight $w(x) = 1$ minimizes the error average, defined as

$$E = \lim_{m \to \infty} \left\{ \frac{1}{m} \sum_{j=1}^{m} [f(x_j) - r_n(x_j)]^2 \right\}^{1/2}.$$

9. Consider the function

$$F(a_0, a_1, \ldots, a_n) = \int_0^1 \left[f(x) - \sum_{j=0}^{n} a_j x^j \right]^2 dx$$

and determine the coefficients $a_0, a_1, \ldots, a_n$ in such a way that F is minimized. Which kind of linear system is obtained?

[*Hint*: enforce the conditions $\partial F / \partial a_i = 0$ with $i = 0, 1, \ldots, n$. The matrix of the final linear system is the Hilbert matrix (see Example 3.2, Chapter 3) which is strongly ill-conditioned.]

11

Numerical Solution of Ordinary Differential Equations

In this chapter we deal with the numerical solutions of the Cauchy problem for ordinary differential equations (henceforth abbreviated by ODEs). After a brief review of basic notions about ODEs, we introduce the most widely used techniques for the numerical approximation of scalar equations. The concepts of consistency, convergence, zero-stability and absolute stability will be addressed. Then, we extend our analysis to systems of ODEs, with emphasis on *stiff* problems.

11.1 The Cauchy Problem

The Cauchy problem (also known as the initial-value problem) consists of finding the solution of an ODE, in the scalar or vector case, given suitable initial conditions. In particular, in the scalar case, denoting by I an interval of $\mathbb{R}$ containing the point t_0, the Cauchy problem associated with a first order ODE reads:

find a real-valued function $y \in C^1(I)$, such that

$$\begin{cases} y'(t) = f(t, y(t)), & t \in I, \\ y(t_0) = y_0, \end{cases} \tag{11.1}$$

where $f(t, y)$ is a given real-valued function in the strip $S = I \times (-\infty, +\infty)$, which is continuous with respect to both variables. Should f depend on t only through y, the differential equation is called *autonomous*.

Most of our analysis will be concerned with one single differential equation (scalar case). The extension to the case of systems of first-order ODEs will be addressed in Section 11.9.
If f is continuous with respect to t, then the solution to (11.1) satisfies

$$y(t) - y_0 = \int_{t_0}^{t} f(\tau, y(\tau))d\tau. \tag{11.2}$$

Conversely, if y is defined by (11.2), then it is continuous in I and $y(t_0) = y_0$. Moreover, since y is a primitive of the continuous function $f(\cdot, y(\cdot))$, $y \in C^1(I)$ and satisfies the differential equation $y'(t) = f(t, y(t))$.

Thus, if f is continuous the Cauchy problem (11.1) is equivalent to the integral equation (11.2). We shall see later on how to take advantage of this equivalence in the numerical methods.

Let us now recall two existence and uniqueness results for (11.1).

1. **Local existence and uniqueness**.

   Suppose that $f(t, y)$ is locally Lipschitz continuous at (t_0, y_0) with respect to y, that is, there exist two neighborhoods, $J \subseteq I$ of t_0 of width r_J, and Σ of y_0 of width r_Σ, and a constant $L > 0$, such that

   $$|f(t, y_1) - f(t, y_2)| \leq L|y_1 - y_2| \quad \forall t \in J,\ \forall y_1, y_2 \in \Sigma. \tag{11.3}$$

   Then, the Cauchy problem (11.1) admits a unique solution in a neighborhood of t_0 with radius r_0 with $0 < r_0 < \min(r_J, r_\Sigma/M, 1/L)$, where M is the maximum of $|f(t, y)|$ on $J \times \Sigma$. This solution is called the *local solution*.

   Notice that condition (11.3) is automatically satisfied if f has continuous derivative with respect to y: indeed, in such a case it suffices to choose L as the maximum of $|\partial f(t, y)/\partial y|$ in $\overline{J \times \Sigma}$.

2. **Global existence and uniqueness**. The problem admits a unique *global solution* if one can take $J = I$ and $\Sigma = \mathbb{R}$ in (11.3), that is, if f is *uniformly Lipschitz continuous* with respect to y.

In view of the stability analysis of the Cauchy problem, we consider the following problem

$$\begin{cases} z'(t) = f(t, z(t)) + \delta(t), & t \in I, \\ z(t_0) = y_0 + \delta_0, \end{cases} \tag{11.4}$$

where $\delta_0 \in \mathbb{R}$ and δ is a continuous function on I. Problem (11.4) is derived from (11.1) by perturbing both the initial datum y_0 and the source function f. Let us now characterize the sensitivity of the solution z to those perturbations.

Definition 11.1 ([Hah67], [Ste71] or [PS91]). Let I be a bounded set. The Cauchy problem (11.1) is *stable in the sense of Liapunov* (or *stable*) on I if, for any perturbation $(\delta_0, \delta(t))$ satisfying

$$|\delta_0| < \varepsilon, \qquad |\delta(t)| < \varepsilon \quad \forall t \in I,$$

with $\varepsilon > 0$ sufficiently small to guarantee that the solution to the perturbed problem (11.4) does exist, then

$$\exists C > 0 \,\text{independent of } \varepsilon \text{ such that} \quad |y(t) - z(t)| < C\varepsilon, \qquad \forall t \in I. \tag{11.5}$$

If I has no upperly bound we say that (11.1) is *asymptotically stable* if, as well as being Liapunov stable in any bounded interval I, the following limit also holds

$$|y(t) - z(t)| \to 0, \qquad \text{for } t \to +\infty. \tag{11.6}$$

■

The requirement that the Cauchy problem is stable is equivalent to requiring that it is well-posed in the sense stated in Chapter 2.

The uniform Lipschitz-continuity of f with respect to y suffices to ensure the stability of the Cauchy problem. Indeed, letting $w(t) = z(t) - y(t)$, we have

$$w'(t) = f(t, z(t)) - f(t, y(t)) + \delta(t).$$

Therefore,

$$w(t) = \delta_0 + \int_{t_0}^{t} [f(s, z(s)) - f(s, y(s))]\, ds + \int_{t_0}^{t} \delta(s) ds, \qquad \forall t \in I.$$

Thanks to previous assumptions, it follows that

$$|w(t)| \le (1 + |t - t_0|)\, \varepsilon + L \int_{t_0}^{t} |w(s)| ds.$$

Applying the Gronwall lemma (which we include below for the reader's ease) yields

$$|w(t)| \le (1 + |t - t_0|)\, \varepsilon e^{L|t - t_0|}, \qquad \forall t \in I$$

and, thus, (11.5) with $C = (1 + K_I)e^{LK_I}$ where $K_I = \max_{t \in I} |t - t_0|$.

Lemma 11.1 (Gronwall) *Let p be an integrable function nonnegative on the interval $(t_0, t_0 + T)$, and let g and φ be two continuous functions on $[t_0, t_0 + T]$, g being nondecreasing. If φ satisfies the inequality*

$$\varphi(t) \le g(t) + \int_{t_0}^{t} p(\tau)\varphi(\tau) d\tau, \qquad \forall t \in [t_0, t_0 + T],$$

then

$$\varphi(t) \leq g(t) \exp\left(\int_{t_0}^{t} p(\tau)d\tau\right), \qquad \forall t \in [t_0, t_0 + T].$$

For the proof, see, for instance, [QV94], Lemma 1.4.1.

The constant C that appears in (11.5) could be very large and, in general, depends on the upper extreme of the interval I, as in the proof above. For that reason, the property of asymptotic stability is more suitable for describing the behavior of the *dynamical system* (11.1) as $t \to +\infty$ (see [Arn73]).

As is well-known, only a restricted number of nonlinear ODEs can be solved in closed form (see, for instance, [Arn73]). Moreover, even when this is possible, it is not always a straightforward task to find an explicit expression of the solution; for example, consider the (very simple) equation $y' = (y-t)/(y+t)$, whose solution is only implicitly defined by the relation $(1/2)\log(t^2 + y^2) + \tan^{-1}(y/t) = C$, where C is a constant depending on the initial condition.

For this reason we are interested in numerical methods, since these can be applied to any ODE under the sole condition that it admits a unique solution.

11.2 One-Step Numerical Methods

Let us address the numerical approximation of the Cauchy problem (11.1). Fix $0 < T < +\infty$ and let $I = (t_0, t_0 + T)$ be the integration interval and, correspondingly, for $h > 0$, let $t_n = t_0 + nh$, with $n = 0, 1, 2, \ldots, N_h$, be the sequence of discretization nodes of I into subintervals $I_n = [t_n, t_{n+1}]$. The width h of such subintervals is called the *discretization stepsize*. Notice that N_h is the maximum integer such that $t_{N_h} \leq t_0 + T$. Let u_j be the approximation at node t_j of the exact solution $y(t_j)$; this solution will be henceforth shortly denoted by y_j. Similarly, f_j denotes the value $f(t_j, u_j)$. We obviously set $u_0 = y_0$.

Definition 11.2 A numerical method for the approximation of problem (11.1) is called a *one-step* method if $\forall n \geq 0$, u_{n+1} depends only on u_n. Otherwise, the scheme is called a *multistep* method. ■

For now, we focus our attention on one-step methods. Here are some of them:

1. **forward Euler method**

$$u_{n+1} = u_n + hf_n; \tag{11.7}$$

2. **backward Euler method**

$$u_{n+1} = u_n + hf_{n+1}. \tag{11.8}$$

In both cases, y' is approximated through a finite difference: forward and backward differences are used in (11.7) and (11.8), respectively. Both finite differences are first-order approximations of the first derivative of y with respect to h (see Section 10.10.1).

3. **trapezoidal (or Crank-Nicolson) method**

$$u_{n+1} = u_n + \frac{h}{2}\left[f_n + f_{n+1}\right]. \tag{11.9}$$

This method stems from approximating the integral on the right side of (11.2) by the trapezoidal quadrature rule (9.11).

4. **Heun method**

$$u_{n+1} = u_n + \frac{h}{2}[f_n + f(t_{n+1}, u_n + hf_n)]. \tag{11.10}$$

This method can be derived from the trapezoidal method substituting $f(t_{n+1}, u_n + hf(t_n, u_n))$ for $f(t_{n+1}, u_{n+1})$ in (11.9) (i.e., using the forward Euler method to compute u_{n+1}).

In this last case, we notice that the aim is to transform an *implicit* method into an *explicit* one. Addressing this concern, we recall the following.

Definition 11.3 (explicit and implicit methods) A method is called *explicit* if u_{n+1} can be computed directly in terms of (some of) the previous values u_k, $k \leq n$. A method is said to be *implicit* if u_{n+1} depends implicitly on itself through f. ■

Methods (11.7) and (11.10) are explicit, while (11.8) and (11.9) are implicit. These latter require at each time step to solving a nonlinear problem if f depends nonlinearly on the second argument.

A remarkable example of one-step methods are the Runge-Kutta methods, which will be analyzed in Section 11.8.

11.3 Analysis of One-Step Methods

Any one-step explicit method for the approximation of (11.1) can be cast in the concise form

$$u_{n+1} = u_n + h\Phi(t_n, u_n, f_n; h), \quad 0 \leq n \leq N_h - 1, \quad u_0 = y_0, \tag{11.11}$$

where $\Phi(\cdot,\cdot,\cdot;\cdot)$ is called an *increment function*. Letting as usual $y_n = y(t_n)$, analogously to (11.11) we can write

$$y_{n+1} = y_n + h\Phi(t_n, y_n, f(t_n, y_n); h) + \varepsilon_{n+1}, \quad 0 \le n \le N_h - 1, \tag{11.12}$$

where ε_{n+1} is the residual arising at the point t_{n+1} when we pretend that the exact solution "satisfies" the numerical scheme. Let us write the residual as

$$\varepsilon_{n+1} = h\tau_{n+1}(h).$$

The quantity $\tau_{n+1}(h)$ is called the *local truncation error* (LTE) at the node t_{n+1}. We thus define the *global truncation error* to be the quantity

$$\tau(h) = \max_{0 \le n \le N_h - 1} |\tau_{n+1}(h)|$$

Notice that $\tau(h)$ depends on the solution y of the Cauchy problem (11.1).

The forward Euler's method is a special instance of (11.11), where

$$\Phi(t_n, u_n, f_n; h) = f_n,$$

while to recover Heun's method we must set

$$\Phi(t_n, u_n, f_n; h) = \frac{1}{2}\left[f_n + f(t_n + h, u_n + hf_n)\right].$$

A one-step explicit scheme is fully characterized by its increment function Φ. This function, in all the cases considered thus far, is such that

$$\lim_{h\to 0} \Phi(t_n, y_n, f(t_n, y_n); h) = f(t_n, y_n), \quad \forall t_n \ge t_0 \tag{11.13}$$

Property (11.13), together with the obvious relation $y_{n+1} - y_n = hy'(t_n) + \mathcal{O}(h^2)$, $\forall n \ge 0$, allows one to obtain from (11.12) that $\lim_{h\to 0} \tau_n(h) = 0$, $0 \le n \le N_h - 1$. In turn, this condition ensures that

$$\lim_{h\to 0} \tau(h) = 0$$

which expresses the *consistency* of the numerical method (11.11) with the Cauchy problem (11.1). In general, a method is said to be *consistent* if its LTE is infinitesimal with respect to h. Moreover, a scheme has *order* p if, $\forall t \in I$, the solution $y(t)$ of the Cauchy problem (11.1) fulfills the condition

$$\tau(h) = \mathcal{O}(h^p) \quad \text{for } h \to 0. \tag{11.14}$$

Using Taylor expansions, as was done in Section 11.2, it can be proved that the forward Euler method has order 1, while the Heun method has order 2 (see Exercises 1 and 2).

11.3.1 The Zero-Stability

Let us formulate a requirement analogous to the one for Liapunov stability (11.5), specifically for the numerical scheme. If (11.5) is satisfied with a constant C independent of h, we shall say that the numerical problem is *zero-stable*. Precisely:

Definition 11.4 (zero-stability of one-step methods) The numerical method (11.11) for the approximation of problem (11.1) is *zero-stable* if

$$\exists h_0 > 0,\ \exists C > 0 :\ \forall h \in (0, h_0],\ |z_n^{(h)} - u_n^{(h)}| \leq C\varepsilon,\ 0 \leq n \leq N_h, \quad (11.15)$$

where $z_n^{(h)}$, $u_n^{(h)}$ are respectively the solutions of the problems

$$\begin{cases} z_{n+1}^{(h)} = z_n^{(h)} + h\left[\Phi(t_n, z_n^{(h)}, f(t_n, z_n^{(h)}); h) + \delta_{n+1}\right], \\ z_0 = y_0 + \delta_0, \end{cases} \quad (11.16)$$

$$\begin{cases} u_{n+1}^{(h)} = u_n^{(h)} + h\Phi(t_n, u_n^{(h)}, f(t_n, u_n^{(h)}); h), \\ u_0 = y_0, \end{cases} \quad (11.17)$$

for $0 \leq n \leq N_h - 1$, under the assumption that $|\delta_k| \leq \varepsilon$, $0 \leq k \leq N_h$. ■

Zero-stability thus requires that, in a bounded interval, (11.15) holds for any value $h \leq h_0$. This property deals, in particular, with the behavior of the numerical method in the limit case $h \to 0$ and this justifies the name of *zero*-stability. This latter is therefore a distinguishing property of the numerical method itself, not of the Cauchy problem (which, indeed, is stable due to the uniform Lipschitz continuity of f). Property (11.15) ensures that the numerical method has a weak sensitivity with respect to small changes in the data and is thus stable in the sense of the general definition given in Chapter 2.

Remark 11.1 The constant C in (11.15) is independent of h (and thus of N_h), but it can depend on the width T of the integration interval I. Actually, (11.15) does not exclude *a priori* the constant C from being an unbounded function of T. ■

The request that a numerical method be stable arises, before anything else, from the need of keeping under control the (unavoidable) errors introduced by the finite arithmetic of the computer. Indeed, if the numerical method were not zero-stable, the rounding errors made on y_0 as well as in the process of computing $f(t_n, u_n)$ would make the computed solution completely useless.

Theorem 11.1 (Zero-stability) *Consider the explicit one-step method* (11.11) *for the numerical solution of the Cauchy problem* (11.1). *Assume that the increment function* Φ *is Lipschitz continuous with respect to the second argument, with constant* Λ *independent of* h *and of the nodes* $t_j \in [t_0, t_0 + T]$, *that is*

$$\exists h_0 > 0,\ \exists \Lambda > 0 : \forall h \in (0, h_0]$$

$$|\Phi(t_n, u_n^{(h)}, f(t_n, u_n^{(h)}); h) - \Phi(t_n, z_n^{(h)}, f(t_n, z_n^{(h)}); h)| \leq \Lambda |u_n^{(h)} - z_n^{(h)}|,\ 0 \leq n \leq N_h. \tag{11.18}$$

Then, method (11.11) *is zero-stable.*

Proof. Setting $w_j^{(h)} = z_j^{(h)} - u_j^{(h)}$, by subtracting (11.17) from (11.16) we obtain, for $j = 0, \ldots, N_h - 1$,

$$w_{j+1}^{(h)} = w_j^{(h)} + h\left[\Phi(t_j, z_j^{(h)}, f(t_j, z_j^{(h)}); h) - \Phi(t_j, u_j^{(h)}, f(t_j, u_j^{(h)}); h)\right] + h\delta_{j+1}.$$

Summing over j gives, for $n = 1, \ldots, N_h$,

$$w_n^{(h)} = w_0^{(h)} + h\sum_{j=0}^{n-1} \delta_{j+1} + h\sum_{j=0}^{n-1} \left(\Phi(t_j, z_j^{(h)}, f(t_j, z_j^{(h)}); h) - \Phi(t_j, u_j^{(h)}, f(t_j, u_j^{(h)}); h)\right),$$

so that, by (11.18)

$$|w_n^{(h)}| \leq |w_0| + h\sum_{j=0}^{n-1} |\delta_{j+1}| + h\Lambda \sum_{j=0}^{n-1} |w_j^{(h)}|, \qquad 1 \leq n \leq N_h. \tag{11.19}$$

Applying the discrete Gronwall lemma, given below, we obtain

$$|w_n^{(h)}| \leq (1 + hn)\, \varepsilon e^{nh\Lambda}, \qquad 1 \leq n \leq N_h.$$

Then (11.15) follows from noticing that $hn \leq T$ and setting $C = (1 + T)\, e^{\Lambda T}$. $\diamond$

Notice that zero-stability implies the boundedness of the solution when f is linear with respect to the second argument.

Lemma 11.2 (discrete Gronwall) *Let* k_n *be a nonnegative sequence and* φ_n *a sequence such that*

$$\begin{cases} \varphi_0 \leq g_0 \\ \varphi_n \leq g_0 + \displaystyle\sum_{s=0}^{n-1} p_s + \sum_{s=0}^{n-1} k_s \phi_s, \qquad n \geq 1. \end{cases}$$

If $g_0 \geq 0$ and $p_n \geq 0$ for any $n \geq 0$, then

$$\varphi_n \leq \left(g_0 + \sum_{s=0}^{n-1} p_s \right) \exp\left(\sum_{s=0}^{n-1} k_s \right), \qquad n \geq 1.$$

For the proof, see, for instance, [QV94], Lemma 1.4.2. In the specific case of the Euler method, checking the property of zero-stability can be done directly using the Lipschitz continuity of f (we refer the reader to the end of Section 11.3.2). In the case of multistep methods, the analysis will lead to the verification of a purely algebraic property, the so-called *root condition* (see Section 11.6.3).

11.3.2 Convergence Analysis

Definition 11.5 A method is said to be *convergent* if

$$\forall n = 0, \ldots, N_h, \qquad |u_n - y_n| \leq C(h)$$

where $C(h)$ is an infinitesimal with respect to h. In that case, it is said to be *convergent with order p* if $\exists \mathcal{C} > 0$ such that $C(h) = \mathcal{C}h^p$. ■

We can prove the following theorem.

Theorem 11.2 (Convergence) *Under the same assumptions as in Theorem* 11.1*, we have*

$$|y_n - u_n| \leq (|y_0 - u_0| + nh\tau(h))\, e^{nh\Lambda}, \qquad 1 \leq n \leq N_h. \tag{11.20}$$

Therefore, if the consistency assumption (11.13) *holds and $|y_0 - u_0| \to 0$ as $h \to 0$, then the method is convergent. Moreover, if $|y_0 - u_0| = \mathcal{O}(h^p)$ and the method has order p, then it is also convergent with order p.*

Proof. Setting $w_j = y_j - u_j$, subtracting (11.11) from (11.12) and proceeding as in the proof of the previous theorem yields inequality (11.19), with the understanding that

$$w_0 = y_0 - u_0, \text{ and } \delta_{j+1} = \tau_{j+1}(h).$$

The estimate (11.20) is then obtained by applying again the discrete Gronwall lemma. From the fact that $nh \leq T$ and $\tau(h) = \mathcal{O}(h^p)$, we can conclude that $|y_n - u_n| \leq Ch^p$ with C depending on T and Λ but not on h. ◇

A consistent and zero-stable method is thus convergent. This property is known as the *Lax-Richtmyer theorem* or *equivalence theorem* (the converse: "a convergent method is zero-stable" being obviously true). This theorem, which is proven in [IK66], was already advocated in Section 2.2.1 and is a central result in the analysis of numerical methods for ODEs (see [Dah56] or

[Hen62] for linear multistep methods, [But66], [MNS74] for a wider classes of methods). It will be considered again in Section 11.5 for the analysis of multistep methods.

We carry out in detail the convergence analysis in the case of the forward Euler method, without resorting to the discrete Gronwall lemma. In the first part of the proof we assume that any operation is performed in exact arithmetic and that $u_0 = y_0$.

Denote by $e_{n+1} = y_{n+1} - u_{n+1}$ the error at node t_{n+1} with $n = 0, 1, \ldots$ and notice that

$$e_{n+1} = (y_{n+1} - u^*_{n+1}) + (u^*_{n+1} - u_{n+1}), \tag{11.21}$$

where $u^*_{n+1} = y_n + hf(t_n, y_n)$ is the solution obtained after one step of the forward Euler method starting from the initial datum y_n (see Figure 11.1). The first addendum in (11.21) accounts for the consistency error, the second one for the cumulation of these errors. Then

$$y_{n+1} - u^*_{n+1} = h\tau_{n+1}(h), \quad u^*_{n+1} - u_{n+1} = e_n + h\left[f(t_n, y_n) - f(t_n, u_n)\right].$$

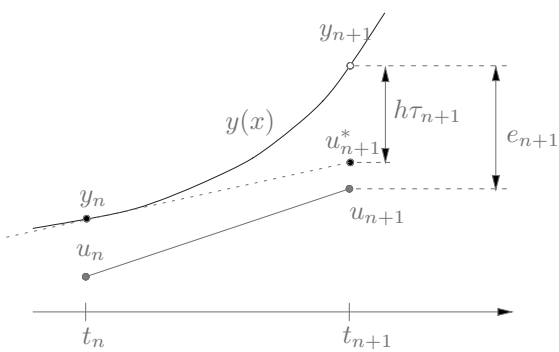

FIGURE 11.1. Geometrical interpretation of the local and global truncation errors at node t_{n+1} for the forward Euler method

As a consequence,

$$|e_{n+1}| \le h|\tau_{n+1}(h)| + |e_n| + h|f(t_n, y_n) - f(t_n, u_n)| \le h\tau(h) + (1 + hL)|e_n|,$$

L being the Lipschitz constant of f. By recursion on n, we find

$$\begin{aligned} |e_{n+1}| &\le \left[1 + (1 + hL) + \ldots + (1 + hL)^n\right] h\tau(h) \\ &= \frac{(1 + hL)^{n+1} - 1}{L}\tau(h) \le \frac{e^{L(t_{n+1} - t_0)} - 1}{L}\tau(h). \end{aligned}$$

The last inequality follows from noticing that $1+hL \leq e^{hL}$ and $(n+1)h = t_{n+1} - t_0$.

On the other hand, if $y \in C^2(I)$, the LTE for the forward Euler method is (see Section 10.10.1)

$$\tau_{n+1}(h) = \frac{h}{2}y''(\xi), \quad \xi \in (t_n, t_{n+1}),$$

and thus, $\tau(h) \leq (M/2)h$, where $M = \max_{\xi \in I} |y''(\xi)|$. In conclusion,

$$|e_{n+1}| \leq \frac{e^{L(t_{n+1}-t_0)} - 1}{L} \frac{M}{2} h, \quad \forall n \geq 0, \tag{11.22}$$

from which it follows that the global error tends to zero with the same order as the local truncation error.

If also the rounding errors are accounted for, we can assume that the solution $\bar{u}_{n+1}$, actually computed by the forward Euler method at time t_{n+1}, is such that

$$\bar{u}_0 = y_0 + \zeta_0, \quad \bar{u}_{n+1} = \bar{u}_n + hf(t_n, \bar{u}_n) + \zeta_{n+1}, \tag{11.23}$$

having denoted the rounding error by ζ_j, for $j \geq 0$.
Problem (11.23) is an instance of (11.16), provided that we identify ζ_{n+1} and $\bar{u}_n$ with $h\delta_{n+1}$ and $z_n^{(h)}$ in (11.16), respectively. Combining Theorems 11.1 and 11.2 we get, instead of (11.22), the following error estimate

$$|y_{n+1} - \bar{u}_{n+1}| \leq e^{L(t_{n+1}-t_0)} \left[|\zeta_0| + \frac{1}{L} \left(\frac{M}{2} h + \frac{\zeta}{h} \right) \right],$$

where $\zeta = \max_{1 \leq j \leq n+1} |\zeta_j|$. The presence of rounding errors does not allow, therefore, to conclude that as $h \to 0$, the error goes to zero. Actually, there exists an optimal (non null) value of h, h_{opt}, for which the error is minimized. For $h < h_{opt}$, the rounding error dominates the truncation error and the global error increases.

11.3.3 The Absolute Stability

The property of *absolute stability* is in some way specular to zero-stability, as far as the roles played by h and I are concerned. Heuristically, we say that a numerical method is absolutely stable if, *for h fixed*, u_n remains bounded as $t_n \to +\infty$. This property, thus, deals with the asymptotic behavior of u_n, as opposed to a zero-stable method for which, for a fixed integration interval, u_n remains bounded as $h \to 0$.
For a precise definition, consider the linear Cauchy problem (that from now on, we shall refer to as the *test problem*)

$$\begin{cases} y'(t) = \lambda y(t), & t > 0, \\ y(0) = 1, \end{cases} \tag{11.24}$$

with $\lambda \in \mathbb{C}$, whose solution is $y(t) = e^{\lambda t}$. Notice that $\lim_{t \to +\infty} |y(t)| = 0$ if $\mathrm{Re}(\lambda) < 0$.

Definition 11.6 A numerical method for approximating (11.24) is *absolutely stable* if

$$|u_n| \longrightarrow 0 \quad \text{as} \quad t_n \longrightarrow +\infty. \tag{11.25}$$

Let h be the discretization stepsize. The numerical solution u_n of (11.24) obviously depends on h and λ. The *region of absolute stability* of the numerical method is the subset of the complex plane

$$\mathcal{A} = \{ z = h\lambda \in \mathbb{C} : \ (11.25) \ \text{is satisfied} \}. \tag{11.26}$$

Thus, $\mathcal{A}$ is the set of the values of the product $h\lambda$ for which the numerical method furnishes solutions that decay to zero as t_n tends to infinity. ■

Let us check whether the one-step methods introduced previously are absolutely stable.

1. *Forward Euler method*: applying (11.7) to problem (11.24) yields $u_{n+1} = u_n + h\lambda u_n$ for $n \geq 0$, with $u_0 = 1$. Proceeding recursively on n we get

$$u_n = (1 + h\lambda)^n, \qquad n \geq 0.$$

Therefore, condition (11.25) is satisfied iff $|1 + h\lambda| < 1$, that is, if $h\lambda$ lies within the unit circle with center at $(-1, 0)$ (see Figure 11.3). This amounts to requiring that

$$h\lambda \in \mathbb{C}^- \quad \text{and} \quad 0 < h < -\frac{2\mathrm{Re}(\lambda)}{|\lambda|^2} \tag{11.27}$$

where

$$\mathbb{C}^- = \{ z \in \mathbb{C} : \ \mathrm{Re}(z) < 0 \}.$$

Example 11.1 For the Cauchy problem $y'(x) = -5y(x)$ for $x > 0$ and $y(0) = 1$, condition (11.27) implies $0 < h < 2/5$. Figure 11.2 (left) shows the behavior of the computed solution for two values of h which do not fulfill this condition, while on the right we show the solutions for two values of h that do. Notice that in this second case the oscillations, if present, damp out as t grows. •

2. *Backward Euler method*: proceeding as before, we get this time

$$u_n = \frac{1}{(1 - h\lambda)^n}, \qquad n \geq 0.$$

The absolute stability property (11.25) is satisfied *for any value of* $h\lambda$ that does not belong to the unit circle of center $(1, 0)$ (see Figure 11.3, right).

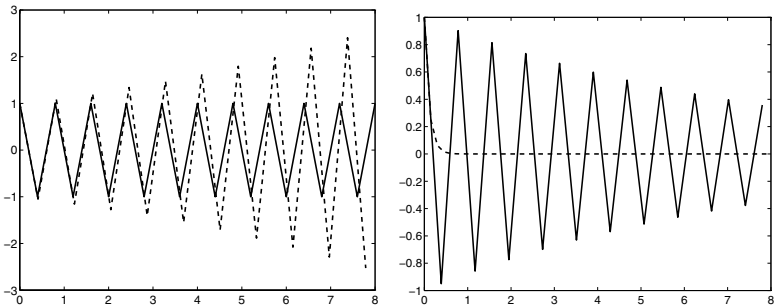

FIGURE 11.2. Left: computed solutions for $h = 0.41 > 2/5$ (dashed line) and $h = 2/5$ (solid line). Notice how, in the limiting case $h = 2/5$, the oscillations remain unmodified as t grows. Right: two solutions are reported for $h = 0.39$ (solid line) and $h = 0.15$ (dashed line)

Example 11.2 The numerical solution given by the backward Euler method in the case of Example 11.1 does not exhibit any oscillation for any value of h. On the other hand, the same method, if applied to the problem $y'(t) = 5y(t)$ for $t > 0$ and with $y(0) = 1$, computes a solution that decays *anyway* to zero as $t \to \infty$ if $h > 2/5$, despite the fact that the exact solution of the Cauchy problem tends to infinity. •

3. *Trapezoidal (or Crank-Nicolson) method*: we get

$$u_n = \left[\left(1 + \frac{1}{2}\lambda h\right) / \left(1 - \frac{1}{2}\lambda h\right)\right]^n, \qquad n \geq 0,$$

hence (11.25) is fulfilled for any $h\lambda \in \mathbb{C}^-$.

4. *Heun's method*: applying (11.10) to problem (11.24) and proceeding by recursion on n, we obtain

$$u_n = \left[1 + h\lambda + \frac{(h\lambda)^2}{2}\right]^n, \qquad n \geq 0.$$

As shown in Figure 11.3 the region of absolute stability of Heun's method is larger than the corresponding one of Euler's method. However, its restriction to the real axis is the same.
We say that a method is *A-stable* if $\mathcal{A} \cap \mathbb{C}^- = \mathbb{C}^-$, i.e., if for $\mathrm{Re}(\lambda) < 0$, condition (11.25) is satisfied for all values of h.

The backward Euler and Crank-Nicolson methods are A-stable, while the forward Euler and Heun methods are conditionally stable.

Remark 11.2 Notice that the implicit one-step methods examined so far are *unconditionally absolutely stable*, while explicit schemes are *condition-*

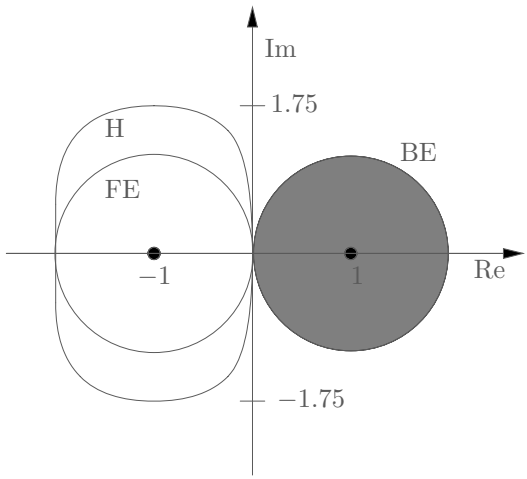

FIGURE 11.3. Regions of absolute stability for the forward (FE) and backward Euler (BE) methods and for Heun's method (H). Notice that the region of absolute stability of the BE method lies outside the unit circle of center $(1, 0)$ (shaded area)

ally absolutely stable. This is, however, not a general rule: in fact, there exist implicit unstable or only conditionally stable schemes. On the contrary, there are no explicit unconditionally absolutely stable schemes [Wid67]. ■

11.4 Difference Equations

For any integer $k \geq 1$, an equation of the form

$$u_{n+k} + \alpha_{k-1} u_{n+k-1} + \ldots + \alpha_0 u_n = \varphi_{n+k}, \quad n = 0, 1, \ldots \tag{11.28}$$

is called a *linear difference equation* of order k. The coefficients $\alpha_0 \neq 0$, $\alpha_1, \ldots, \alpha_{k-1}$ may or may not depend on n. If, for any n, the right side φ_{n+k} is equal to zero, the equation is said *homogeneous*, while if the α_j's are independent of n it is called *linear difference equation with constant coefficients*.

Difference equations arise for instance in the discretization of ordinary differential equations. Regarding this, we notice that all the numerical methods examined so far end up with equations like (11.28). More generally, equations like (11.28) are encountered when quantities are defined through linear recursive relations. Another relevant application is concerned with the discretization of boundary value problems (see Chapter 12). For further details on the subject, we refer to Chapters 2 and 5 of [BO78] and to Chapter 6 of [Gau97].

Any sequence $\{u_n,\, n = 0, 1, \dots\}$ of values that satisfy (11.28) is called a *solution* to the equation (11.28). Given k *initial values* $u_0, \dots, u_{k-1}$, it is always possible to construct a solution of (11.28) by computing (sequentially)

$$u_{n+k} = [\varphi_{n+k} - (\alpha_{k-1}u_{n+k-1} + \dots + \alpha_0 u_n)], \quad n = 0, 1, \dots$$

However, our interest is to find an expression of the solution u_{n+k} which depends only on the coefficients and on the initial values.

We start by considering the *homogeneous case with constant coefficients*,

$$u_{n+k} + \alpha_{k-1}u_{n+k-1} + \dots + \alpha_0 u_n = 0, \quad n = 0, 1, \dots \tag{11.29}$$

and associate with (11.29) the *characteristic polynomial* $\Pi \in \mathbb{P}_k$ defined as

$$\Pi(r) = r^k + \alpha_{k-1}r^{k-1} + \dots + \alpha_1 r + \alpha_0. \tag{11.30}$$

Denoting its roots by r_j, $j = 0, \dots, k-1$, any sequence of the form

$$\left\{r_j^n,\, n = 0, 1, \dots\right\}, \qquad \text{for } j = 0, \dots, k-1 \tag{11.31}$$

is a solution of (11.29), since

$$\begin{aligned} &r_j^{n+k} + \alpha_{k-1}r_j^{n+k-1} + \dots + \alpha_0 r_j^n \\ &= r_j^n \left(r_j^k + \alpha_{k-1}r_j^{k-1} + \dots + \alpha_0\right) = r_j^n \Pi(r_j) = 0. \end{aligned}$$

We say that the k sequences defined in (11.31) are the *fundamental solutions* of the homogeneous equation (11.29). Any sequence of the form

$$u_n = \gamma_0 r_0^n + \gamma_1 r_1^n + \dots + \gamma_{k-1} r_{k-1}^n, \qquad n = 0, 1, \dots \tag{11.32}$$

is still a solution to (11.29), since it is a linear equation.
The coefficients $\gamma_0, \dots, \gamma_{k-1}$ can be determined by imposing the k initial conditions $u_0, \dots, u_{k-1}$. Moreover, it can be proved that if all the roots of Π are simple, then *all* the solutions of (11.29) can be cast in the form (11.32).

This last statement no longer holds if there are roots of Π with multiplicity greater than 1. If, for a certain j, the root r_j has multiplicity $m \geq 2$, in order to obtain a system of fundamental solutions that generate all the solutions of (11.29), it suffices to replace the corresponding fundamental solution $\left\{r_j^n, n = 0, 1, \dots\right\}$ with the m sequences

$$\left\{r_j^n,\, n = 0, 1, \dots\right\},\ \left\{nr_j^n,\, n = 0, 1, \dots\right\},\ \dots,\ \left\{n^{m-1}r_j^n,\, n = 0, 1, \dots\right\}.$$

More generally, assuming that $r_0, \dots, r_{k'}$ are distinct roots of Π, with multiplicities equal to $m_0, \dots, m_{k'}$, respectively, we can write the solution of (11.29) as

$$u_n = \sum_{j=0}^{k'} \left(\sum_{s=0}^{m_j - 1} \gamma_{sj} n^s \right) r_j^n, \qquad n = 0, 1, \dots . \tag{11.33}$$

Notice that even in presence of complex conjugate roots one can still obtain a real solution (see Exercise 3).

Example 11.3 For the difference equation $u_{n+2}-u_n = 0$, we have $\Pi(r) = r^2-1$, then $r_0 = -1$ and $r_1 = 1$, therefore the solution is given by $u_n = \gamma_{00}(-1)^n + \gamma_{01}$. In particular, enforcing the initial conditions u_0 and u_1 gives $\gamma_{00} = (u_0 - u_1)/2$, $\gamma_{01} = (u_0 + u_1)/2$. •

Example 11.4 For the difference equation $u_{n+3} - 2u_{n+2} - 7u_{n+1} - 4u_n = 0$, $\Pi(r) = r^3 - 2r^2 - 7r - 4$. Its roots are $r_0 = -1$ (with multiplicity 2), $r_1 = 4$ and the solution is $u_n = (\gamma_{00} + n\gamma_{10})(-1)^n + \gamma_{01}4^n$. Enforcing the initial conditions we can compute the unknown coefficients as the solution of the following linear system

$$\begin{cases} \gamma_{00} + \gamma_{01} & = u_0, \\ -\gamma_{00} - \gamma_{10} + 4\gamma_{01} & = u_1, \\ \gamma_{00} + 2\gamma_{10} + 16\gamma_{01} & = u_2 \end{cases}$$

that yields $\gamma_{00} = (24u_0 - 2u_1 - u_2)/25$, $\gamma_{10} = (u_2 - 3u_1 - 4u_0)/5$ and $\gamma_{01} = (2u_1 + u_0 + u_2)/25$. •

The expression (11.33) is of little practical use since it does not outline the dependence of u_n on the k initial conditions. A more convenient representation is obtained by introducing a new set $\left\{\psi_j^{(n)},\ n = 0, 1, \dots\right\}$ of fundamental solutions that satisfy

$$\psi_j^{(i)} = \delta_{ij}, \quad i, j = 0, 1, \dots, k-1. \tag{11.34}$$

Then, the solution of (11.29) subject to the initial conditions $u_0, \dots, u_{k-1}$ is given by

$$u_n = \sum_{j=0}^{k-1} u_j \psi_j^{(n)}, \qquad n = 0, 1, \dots . \tag{11.35}$$

The new fundamental solutions $\left\{\psi_j^{(n)},\ n = 0, 1, \dots\right\}$ can be represented in terms of those in (11.31) as follows

$$\psi_j^{(n)} = \sum_{m=0}^{k-1} \beta_{j,m} r_m^n \quad \text{for } j = 0, \dots, k-1,\ n = 0, 1, \dots \tag{11.36}$$

By requiring (11.34), we obtain the k linear systems

$$\sum_{m=0}^{k-1} \beta_{j,m} r_m^i = \delta_{ij}, \qquad i, j = 0, \dots, k-1,$$

whose matrix form is

$$\mathrm{R}\mathbf{b}_j = \mathbf{e}_j, \qquad j = 0, \dots, k-1. \tag{11.37}$$

Here $\mathbf{e}_j$ denotes the unit vector of $\mathbb{R}^k$, $\mathrm{R} = (r_{im}) = (r_m^i)$ and $\mathbf{b}_j = (\beta_{j,0}, \dots, \beta_{j,k-1})^T$. If all r_j's are simple roots of Π, the matrix R is nonsingular (see Exercise 5).

The general case where Π has $k'+1$ distinct roots $r_0, \dots, r_{k'}$ with multiplicities $m_0, \dots, m_{k'}$ respectively, can be dealt with by replacing in (11.36) $\{r_j^n, \quad n = 0, 1, \dots\}$ with $\{r_j^n n^s, \quad n = 0, 1, \dots\}$, where $j = 0, \dots, k'$ and $s = 0, \dots, m_j - 1$.

Example 11.5 We consider again the difference equation of Example 11.4. Here we have $\{r_0^n,\ nr_0^n,\ r_1^n,\ n = 0, 1, \dots\}$ so that the matrix R becomes

$$\mathrm{R} = \begin{bmatrix} r_0^0 & 0 & r_2^0 \\ r_0^1 & r_0^1 & r_2^1 \\ r_0^2 & 2r_0^2 & r_2^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ -1 & -1 & 4 \\ 1 & 2 & 16 \end{bmatrix}.$$

Solving the three systems (11.37) yields

$$\begin{aligned} \psi_0^{(n)} &= \frac{24}{25}(-1)^n - \frac{4}{5}n(-1)^n + \frac{1}{25}4^n, \\ \psi_1^{(n)} &= -\frac{2}{25}(-1)^n - \frac{3}{5}n(-1)^n + \frac{2}{25}4^n, \\ \psi_2^{(n)} &= -\frac{1}{25}(-1)^n + \frac{1}{5}n(-1)^n + \frac{1}{25}4^n, \end{aligned}$$

from which it can be checked that the solution $u_n = \sum_{j=0}^{2} u_j \psi_j^{(n)}$ coincides with the one already found in Example 11.4. •

Now we return to the case of *nonconstant coefficients* and consider the following homogeneous equation

$$u_{n+k} + \sum_{j=1}^{k} \alpha_{k-j}(n) u_{n+k-j} = 0, \qquad n = 0, 1, \dots . \tag{11.38}$$

The goal is to transform it into an ODE by means of a function F, called the *generating function* of the equation (11.38). F depends on the real variable t and is derived as follows. We require that the n-th coefficient of the Taylor series of F around $t = 0$ can be written as $\gamma_n u_n$, for some unknown constant γ_n, so that

$$F(t) = \sum_{n=0}^{\infty} \gamma_n u_n t^n. \tag{11.39}$$

The coefficients $\{\gamma_n\}$ are unknown and must be determined in such a way that

$$\sum_{j=0}^{k} c_j F^{(k-j)}(t) = \sum_{n=0}^{\infty} \left[u_{n+k} + \sum_{j=1}^{k} \alpha_{k-j}(n) u_{n+k-j} \right] t^n, \tag{11.40}$$

where c_j are suitable unknown constants not depending on n. Note that owing to (11.39) we obtain the ODE

$$\sum_{j=0}^{k} c_j F^{(k-j)}(t) = 0$$

to which we must add the initial conditions $F^{(j)}(0) = \gamma_j u_j$ for $j = 0, \dots, k-1$. Once F is available, it is simple to recover u_n through the definition of F itself.

Example 11.6 Consider the difference equation

$$(n+2)(n+1)u_{n+2} - 2(n+1)u_{n+1} - 3u_n = 0, \quad n = 0, 1, \dots \tag{11.41}$$

with the initial conditions $u_0 = u_1 = 2$. We look for a generating function of the form (11.39). By term-to-term derivation of the series, we get

$$F'(t) = \sum_{n=0}^{\infty} \gamma_n n u_n t^{n-1}, \quad F''(t) = \sum_{n=0}^{\infty} \gamma_n n(n-1) u_n t^{n-2},$$

and, after some algebra, we find

$$F'(t) = \sum_{n=0}^{\infty} \gamma_n n u_n t^{n-1} = \sum_{n=0}^{\infty} \gamma_{n+1}(n+1) u_{n+1} t^n,$$
$$F''(t) = \sum_{n=0}^{\infty} \gamma_n n(n-1) u_n t^{n-2} = \sum_{n=0}^{\infty} \gamma_{n+2}(n+2)(n+1) u_{n+2} t^n.$$

As a consequence, (11.40) becomes

$$\sum_{n=0}^{\infty} (n+1)(n+2) u_{n+2} t^n - 2\sum_{n=0}^{\infty} (n+1) u_{n+1} t^n - 3\sum_{n=0}^{\infty} u_n t^n$$
$$= c_0 \sum_{n=0}^{\infty} \gamma_{n+2}(n+2)(n+1) u_{n+2} t^n + c_1 \sum_{n=0}^{\infty} \gamma_{n+1}(n+1) u_{n+1} t^n + c_2 \sum_{n=0}^{\infty} \gamma_n u_n t^n,$$

so that, equating both sides, we find

$$\gamma_n = 1 \ \forall n \geq 0, \quad c_0 = 1, \ c_1 = -2, \ c_2 = -3.$$

We have thus associated with the difference equation the following ODE with constant coefficients

$$F''(t) - 2F'(t) - 3F(t) = 0,$$

with the initial condition $F(0) = F'(0) = 2$. The n-th coefficient of the solution $F(t) = e^{3t} + e^{-t}$ is

$$\frac{1}{n!} F^{(n)}(0) = \frac{1}{n!} \left[(-1)^n + 3^n\right],$$

so that $u_n = (1/n!)\left[(-1)^n + 3^n\right]$ is the solution of (11.41). •

The *nonhomogeneous case* (11.28) can be tackled by searching for solutions of the form

$$u_n = u_n^{(0)} + u_n^{(\varphi)},$$

where $u_n^{(0)}$ is the solution of the associated homogeneous equation and $u_n^{(\varphi)}$ is a particular solution of the nonhomogeneous equation. Once the solution of the homogeneous equation is available, a general technique to obtain the solution of the nonhomogeneous equation is based on the method of variation of parameters, combined with a reduction of the order of the difference equation (see [BO78]).

In the special case of difference equations with constant coefficients, with φ_n of the form $c^n Q(n)$, where c is a constant and Q is a polynomial of degree p with respect to the variable n, a possible approach is that of *undetermined coefficients*, where one looks for a particular solution that depends on some undetermined constants and has a known form for some classes of right sides φ_n. It suffices to look for a particular solution of the form

$$u_n^{(\varphi)} = c^n(b_p n^p + b_{p-1} n^{p-1} + \ldots + b_0),$$

where $b_p, \ldots, b_0$ are constants to be determined in such a way that $u_n^{(\varphi)}$ is actually a solution of (11.28).

Example 11.7 Consider the difference equation $u_{n+3} - u_{n+2} + u_{n+1} - u_n = 2^n n^2$. The particular solution is of the form $u_n = 2^n(b_2 n^2 + b_1 n + b_0)$. Substituting this solution into the equation, we find $5b_2 n^2 + (36b_2 + 5b_1)n + (58b_2 + 18b_1 + 5b_0) = n^2$, from which, recalling the principle of identity for polynomials, one gets $b_2 = 1/5$, $b_1 = -36/25$ and $b_0 = 358/125$. •

Analogous to the homogeneous case, it is possible to express the solution of (11.28) as

$$u_n = \sum_{j=0}^{k-1} u_j \psi_j^{(n)} + \sum_{l=k}^{n} \varphi_l \psi_{k-1}^{(n-l+k-1)}, \qquad n = 0, 1, \ldots \tag{11.42}$$

where we define $\psi_{k-1}^{(i)} \equiv 0$ for all $i < 0$ and $\varphi_j \equiv 0$ for all $j < k$.

11.5 Multistep Methods

Let us now introduce some examples of multistep methods (shortly, MS).

Definition 11.7 (q-steps methods) A q-step method ($q \geq 1$) is one which, $\forall n \geq q - 1$, u_{n+1} depends on u_{n+1-q}, but not on the values u_k with $k < n + 1 - q$. ■

A well-known *two-step* explicit method can be obtained by using the centered finite difference (10.61) to approximate the first order derivative in (11.1). This yields the *midpoint method*

$$u_{n+1} = u_{n-1} + 2hf_n, \qquad n \geq 1 \tag{11.43}$$

where $u_0 = y_0$, u_1 is to be determined and f_k denotes the value $f(t_k, u_k)$.

An example of an implicit two-step scheme is the *Simpson method*, obtained from (11.2) with $t_0 = t_{n-1}$ and $t = t_{n+1}$ and by using the Cavalieri-Simpson quadrature rule to approximate the integral of f

$$u_{n+1} = u_{n-1} + \frac{h}{3}[f_{n-1} + 4f_n + f_{n+1}], \qquad n \geq 1 \tag{11.44}$$

where $u_0 = y_0$, and u_1 is to be determined.

From these examples, it is clear that a multistep method requires q initial values $u_0, \ldots, u_{q-1}$ for "taking off". Since the Cauchy problem provides only one datum (u_0), one way to assign the remaining values consists of resorting to explicit one-step methods of high order. An example is given by Heun's method (11.10), other examples are provided by the Runge-Kutta methods, which will be introduced in Section 11.8.

In this section we deal with *linear multistep methods*

$$u_{n+1} = \sum_{j=0}^{p} a_j u_{n-j} + h\sum_{j=0}^{p} b_j f_{n-j} + hb_{-1}f_{n+1}, \ n = p, p+1, \ldots \tag{11.45}$$

which are $p+1$-step methods, $p \geq 0$. For $p = 0$, we recover one-step methods.

The coefficients a_j, b_j are real and fully identify the scheme; they are such that $a_p \neq 0$ or $b_p \neq 0$. If $b_{-1} \neq 0$ the scheme is implicit, otherwise it is explicit.

We can reformulate (11.45) as follows

$$\sum_{s=0}^{p+1} \alpha_s u_{n+s} = h\sum_{s=0}^{p+1} \beta_s f(t_{n+s}, u_{n+s}), \ n = 0, 1, \ldots, N_h - (p+1) \tag{11.46}$$

having set $\alpha_{p+1} = 1$, $\alpha_s = -a_{p-s}$ for $s = 0, \ldots, p$ and $\beta_s = b_{p-s}$ for $s = 0, \ldots, p+1$. Relation (11.46) is a special instance of the linear difference equation (11.28), where we set $k = p+1$ and $\varphi_{n+j} = h\beta_j f(t_{n+j}, u_{n+j})$, for $j = 0, \ldots, p+1$.

Also for MS methods we can characterize consistency in terms of the local truncation error, according to the following definition.

Definition 11.8 The local truncation error (LTE) $\tau_{n+1}(h)$ introduced by the multistep method (11.45) at t_{n+1} (for $n \geq p$) is defined through the

following relation

$$h\tau_{n+1}(h) = y_{n+1} - \left[\sum_{j=0}^{p} a_j y_{n-j} + h\sum_{j=-1}^{p} b_j y'_{n-j}\right], \qquad n \geq p, \quad (11.47)$$

where $y_{n-j} = y(t_{n-j})$ and $y'_{n-j} = y'(t_{n-j})$ for $j = -1, \ldots, p$. ■

Analogous to one-step methods, the quantity $h\tau_{n+1}(h)$ is the residual generated at t_{n+1} if we pretend that the exact solution "satisfies" the numerical scheme. Letting $\tau(h) = \max_n |\tau_n(h)|$, we have the following definition.

Definition 11.9 (Consistency) The multistep method (11.45) is consistent if $\tau(h) \to 0$ as $h \to 0$. Moreover, if $\tau(h) = \mathcal{O}(h^q)$, for some $q \geq 1$, then the method is said to have order q. ■

A more precise characterization of the LTE can be given by introducing the following linear operator $\mathcal{L}$ associated with the linear MS method (11.45)

$$\mathcal{L}[w(t); h] = w(t+h) - \sum_{j=0}^{p} a_j w(t-jh) - h\sum_{j=-1}^{p} b_j w'(t-jh), \quad (11.48)$$

where $w \in C^1(I)$ is an arbitrary function. Notice that the LTE is exactly $\mathcal{L}[y(t_n); h]$. If we assume that w is sufficiently smooth and expand $w(t-jh)$ and $w'(t-jh)$ about $t-ph$, we obtain

$$\mathcal{L}[w(t); h] = C_0 w(t-ph) + C_1 h w^{(1)}(t-ph) + \ldots + C_k h^k w^{(k)}(t-ph) + \ldots$$

Consequently, if the MS method has order q and $y \in C^{q+1}(I)$, we obtain

$$\tau_{n+1}(h) = C_{q+1} h^{q+1} y^{(q+1)}(t_{n-p}) + \mathcal{O}(h^{q+2}).$$

The term $C_{q+1}h^{q+1}y^{(q+1)}(t_{n-p})$ is the so-called *principal local truncation error* (PLTE) while C_{q+1} is the error constant. The PLTE is widely employed in devising adaptive strategies for MS methods (see [Lam91], Chapter 3).

Program 92 provides an implementation of the multistep method in the form (11.45) for the solution of a Cauchy problem on the interval (t_0, T). The input parameters are: the column vector `a` containing the $p+1$ coefficients a_i; the column vector `b` containing the $p+2$ coefficients b_i; the discretization stepsize h; the vector of initial data `u0` at the corresponding time instants `t0`; the macros `fun` and `dfun` containing the functions f and $\partial f/\partial y$. If the MS method is implicit, a tolerance `tol` and a maximum number of admissible iterations `itmax` must be provided. These two parameters monitor the convergence of Newton's method that is employed to solve the

nonlinear equation (11.45) associated with the MS method. In output the code returns the vectors `u` and `t` containing the computed solution at the time instants `t`.

Program 92 - multistep : Linear multistep methods

```
function [t,u] = multistep (a,b,tf,t0,u0,h,fun,dfun,tol,itmax)
y = u0;  t = t0;  f = eval (fun); p = length(a) - 1; u = u0;
nt = fix((tf - t0 (1) )/h);
for k = 1:nt
   lu=length(u);
   G = a' *u (lu:-1:lu-p)+ h * b(2:p+2)' * f(lu:-1:lu-p);
   lt = length(t0); t0 = [t0; t0(lt)+h]; unew = u (lu);
   t = t0 (lt+1); err = tol + 1; it = 0;
   while (err > tol) & (it <= itmax)
     y = unew; den = 1 - h * b (1) * eval(dfun);
     fnew = eval (fun);
     if den == 0
        it = itmax + 1;
     else
        it = it + 1;
        unew = unew - (unew - G - h * b (1) * fnew)/den;
        err = abs (unew - y);
     end
   end
   u = [u; unew]; f = [f; fnew];
end
t = t0;
```

In the forthcoming sections we examine some families of multistep methods.

11.5.1 Adams Methods

These methods are derived from the integral form (11.2) through an approximate evaluation of the integral of f between t_n and t_{n+1}. We suppose that the discretization nodes are equally spaced, i.e., $t_j = t_0 + jh$, with $h > 0$ and $j \geq 1$, and then we integrate, instead of f, its interpolating polynomial on $p+1$ distinct nodes. The resulting schemes are thus *consistent* by construction and have the following form

$$u_{n+1} = u_n + h \sum_{j=-1}^{p} b_j f_{n-j}, \qquad n \geq p. \tag{11.49}$$

The interpolation nodes can be either:

1. $t_n, t_{n-1}, \ldots, t_{n-p}$ (in this case $b_{-1} = 0$ and the resulting method is explicit);

   or

2. $t_{n+1}, t_n, \dots, t_{n-p+1}$ (in this case $b_{-1} \neq 0$ and the scheme is implicit).

The *implicit* schemes are called *Adams-Moulton* methods, while the *explicit* ones are called *Adams-Bashforth* methods.

Adams-Bashforth methods (AB)

Taking $p = 0$ we recover the forward Euler method, since the interpolating polynomial of degree zero at node t_n is given by $\Pi_0 f = f_n$. For $p = 1$, the linear interpolating polynomial at the nodes t_{n-1} and t_n is

$$\Pi_1 f(t) = f_n + (t - t_n)\frac{f_{n-1} - f_n}{t_{n-1} - t_n}.$$

Since $\Pi_1 f(t_n) = f_n$ and $\Pi_1 f(t_{n+1}) = 2f_n - f_{n-1}$, we get

$$\int_{t_n}^{t_{n+1}} \Pi_1 f(t) = \frac{h}{2}\left[\Pi_1 f(t_n) + \Pi_1 f(t_{n+1})\right] = \frac{h}{2}\left[3f_n - f_{n-1}\right].$$

Therefore, the two-step AB method is

$$u_{n+1} = u_n + \frac{h}{2}\left[3f_n - f_{n-1}\right]. \tag{11.50}$$

With a similar procedure, if $p = 2$, we find the three-step AB method

$$u_{n+1} = u_n + \frac{h}{12}\left[23f_n - 16f_{n-1} + 5f_{n-2}\right],$$

while for $p = 3$ we get the four-step AB scheme

$$u_{n+1} = u_n + \frac{h}{24}\left(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}\right).$$

In general, q-step Adams-Bashforth methods have order q. The error constants C^*_{q+1} of these methods are collected in Table 11.1.

Adams-Moulton methods (AM)

If $p = -1$, the Backward Euler scheme is recovered, while if $p = 0$, we construct the linear polynomial interpolating f at the nodes t_n and t_{n+1} to recover the Crank-Nicolson scheme (11.9). In the case of the two-step method, the polynomial of degree 2 interpolating f at the nodes t_{n-1}, t_n, t_{n+1} is generated, yielding the following scheme

$$u_{n+1} = u_n + \frac{h}{12}\left[5f_{n+1} + 8f_n - f_{n-1}\right]. \tag{11.51}$$

The methods corresponding to $p = 3$ and 4 are respectively given by

$$u_{n+1} = u_n + \frac{h}{24}\left(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}\right)$$

$$u_{n+1} = u_n + \frac{h}{720}\left(251f_{n+1} + 646f_n - 264f_{n-1} + 106f_{n-2} - 19f_{n-3}\right).$$

The q-step Adams-Moulton methods have order $q+1$ and their error constants C_{q+1} are summarized in Table 11.1.

q	C^*_{q+1}	C_{q+1}	q	C^*_{q+1}	C_{q+1}
1	$\frac{1}{2}$	$-\frac{1}{2}$	3	$\frac{3}{8}$	$-\frac{1}{24}$
2	$\frac{5}{12}$	$-\frac{1}{12}$	4	$\frac{251}{720}$	$-\frac{19}{720}$

TABLE 11.1. Error constants for Adams-Bashforth methods (having order q) and Adams-Moulton methods (having order $q+1$)

11.5.2 BDF Methods

The so-called *backward differentiation formulae* (henceforth denoted by BDF) are implicit MS methods derived from a complementary approach to the one followed for the Adams methods. In fact, for the Adams methods we have resorted to numerical integration for the source function f, whereas in BDF methods we directly approximate the value of the first derivative of y at node t_{n+1} through the first derivative of the polynomial interpolating y at the $p+1$ nodes $t_{n+1}, t_n, \ldots, t_{n-p+1}$.

By doing so, we get schemes of the form

$$u_{n+1} = \sum_{j=0}^{p} a_j u_{n-j} + hb_{-1}f_{n+1}$$

with $b_{-1} \neq 0$. Method (11.8) represents the most elementary example, corresponding to the coefficients $a_0 = 1$ and $b_{-1} = 1$.

We summarize in Table 11.2 the coefficients of BDF methods that are zero-stable. In fact, we shall see in Section 11.6.3 that only for $p \leq 5$ are BDF methods zero-stable (see [Cry73]).

11.6 Analysis of Multistep Methods

Analogous to what has been done for one-step methods, in this section we provide algebraic conditions that ensure consistency and stability of multistep methods.

p	a_0	a_1	a_2	a_3	a_4	a_5	b_{-1}
0	1	0	0	0	0	0	1
1	$\frac{4}{3}$	$-\frac{1}{3}$	0	0	0	0	$\frac{2}{3}$
2	$\frac{18}{11}$	$-\frac{9}{11}$	$\frac{2}{11}$	0	0	0	$\frac{6}{11}$
3	$\frac{48}{25}$	$-\frac{36}{25}$	$\frac{16}{25}$	$-\frac{3}{25}$	0	0	$\frac{12}{25}$
4	$\frac{300}{137}$	$-\frac{300}{137}$	$\frac{200}{137}$	$-\frac{75}{137}$	$\frac{12}{137}$	0	$\frac{60}{137}$
5	$\frac{360}{147}$	$-\frac{450}{147}$	$\frac{400}{147}$	$-\frac{225}{147}$	$\frac{72}{147}$	$-\frac{10}{147}$	$\frac{60}{137}$

TABLE 11.2. Coefficients of zero-stable BDF methods for $p = 0, 1, \ldots, 5$

11.6.1 Consistency

The property of consistency of a multistep method introduced in Definition 11.9 can be verified by checking that the coefficients satisfy certain algebraic equations, as stated in the following theorem.

Theorem 11.3 *The multistep method* (11.45) *is consistent iff the following algebraic relations among the coefficients are satisfied*

$$\sum_{j=0}^{p} a_j = 1, \quad -\sum_{j=0}^{p} j a_j + \sum_{j=-1}^{p} b_j = 1. \tag{11.52}$$

Moreover, if $y \in C^{q+1}(I)$ for some $q \geq 1$, where y is the solution of the Cauchy problem (11.1)*, then the method is of order q iff* (11.52) *holds and the following additional conditions are satisfied*

$$\sum_{j=0}^{p} (-j)^i a_j + i \sum_{j=-1}^{p} (-j)^{i-1} b_j = 1, \ i = 2, \ldots, q.$$

Proof. Expanding y and f in a Taylor series yields, for any $n \geq p$

$$y_{n-j} = y_n - j h y'_n + \mathcal{O}(h^2), \qquad f_{n-j} = f_n + \mathcal{O}(h). \tag{11.53}$$

Plugging these values back into the multistep scheme and neglecting the terms in h of order higher than 1 gives

$$\begin{aligned}
&y_{n+1} - \sum_{j=0}^{p} a_j y_{n-j} - h \sum_{j=-1}^{p} b_j f_{n-j} \\
&= y_{n+1} - \sum_{j=0}^{p} a_j y_n + h \sum_{j=0}^{p} j a_j y'_n - h \sum_{j=-1}^{p} b_j f_n - \mathcal{O}(h^2) \left(\sum_{j=0}^{p} a_j - \sum_{j=-1}^{p} b_j \right) \\
&= y_{n+1} - \sum_{j=0}^{p} a_j y_n - h y'_n \left(-\sum_{j=0}^{p} j a_j + \sum_{j=-1}^{p} b_j \right) - \mathcal{O}(h^2) \left(\sum_{j=0}^{p} a_j - \sum_{j=-1}^{p} b_j \right)
\end{aligned}$$

where we have replaced y'_n by f_n. From the definition (11.47) we then obtain

$$h\tau_{n+1}(h) = y_{n+1} - \sum_{j=0}^{p} a_j y_n - h y'_n \left(-\sum_{j=0}^{p} j a_j + \sum_{j-1}^{p} b_j \right) - \mathcal{O}(h^2) \left(\sum_{j=0}^{p} a_j - \sum_{j=-1}^{p} b_j \right)$$

hence the local truncation error is

$$\tau_{n+1}(h) = \frac{y_{n+1} - y_n}{h} + \frac{y_n}{h}\left(1 - \sum_{j=0}^{p} a_j\right)$$
$$+ y'_n\left(\sum_{j=0}^{p} j a_j - \sum_{j=-1}^{p} b_j\right) - \mathcal{O}(h)\left(\sum_{j=0}^{p} a_j - \sum_{j=-1}^{p} b_j\right).$$

Since, for any n, $(y_{n+1} - y_n)/h \to y'_n$, as $h \to 0$, it follows that $\tau_{n+1}(h)$ tends to 0 as h goes to 0 iff the algebraic conditions (11.52) are satisfied. The rest of the proof can be carried out in a similar manner, accounting for terms of progressively higher order in the expansions (11.53). ◇

11.6.2 The Root Conditions

Let us employ the multistep method (11.45) to approximately solve the model problem (11.24). The numerical solution satisfies the linear difference equation

$$u_{n+1} = \sum_{j=0}^{p} a_j u_{n-j} + h\lambda \sum_{j=-1}^{p} b_j u_{n-j}, \tag{11.54}$$

which fits the form (11.29). We can therefore apply the theory developed in Section 11.4 and look for fundamental solutions of the form $u_k = [r_i(h\lambda)]^k$, $k = 0, 1, \ldots$, where $r_i(h\lambda)$, for $i = 0, \ldots, p$, are the roots of the polynomial $\Pi \in \mathbb{P}_{p+1}$

$$\Pi(r) = \rho(r) - h\lambda\sigma(r). \tag{11.55}$$

We have denoted by

$$\rho(r) = r^{p+1} - \sum_{j=0}^{p} a_j r^{p-j}, \quad \sigma(r) = b_{-1} r^{p+1} + \sum_{j=0}^{p} b_j r^{p-j}$$

the *first* and *second characteristic polynomials* of the multistep method (11.45), respectively. The polynomial $\Pi(r)$ is the *characteristic polynomial* associated with the difference equation (11.54), and $r_j(h\lambda)$ are its *characteristic roots*.

The roots of ρ are $r_i(0)$, $i = 0, \ldots, p$, and will be abbreviated henceforth by r_i. From the first condition in (11.52) it follows that if a multistep method is consistent then 1 is a root of ρ. We shall assume that such a root (the consistency root) is labelled as $r_0(0) = r_0$ and call the corresponding root $r_0(h\lambda)$ the *principal root*.

Definition 11.10 (Root condition) The multistep method (11.45) is said to satisfy the root condition if all roots r_i are contained within the unit

circle centered at the origin of the complex plane, otherwise, if they fall on its boundary, they must be simple roots of ρ. Equivalently,

$$\begin{cases} |r_j| \leq 1, \qquad j = 0, \ldots, p; \\ \text{furthermore, for those } j \text{ such that } |r_j| = 1, \text{ then } \rho'(r_j) \neq 0. \end{cases} \tag{11.56}$$

■

Definition 11.11 (Strong root condition) The multistep method (11.45) is said to satisfy the strong root condition if it satisfies the root condition and $r_0 = 1$ is the only root lying on the boundary of the unit circle. Equivalently,

$$|r_j| < 1 \qquad j = 1, \ldots, p. \tag{11.57}$$

■

Definition 11.12 (Absolute root condition) The multistep method (11.45) satisfies the absolute root condition if there exists $h_0 > 0$ such that

$$|r_j(h\lambda)| < 1 \qquad j = 0, \ldots, p, \quad \forall h \leq h_0.$$

■

11.6.3 Stability and Convergence Analysis for Multistep Methods

Let us now examine the relation between root conditions and the stability of multistep methods. Generalizing the Definition 11.4, we can get the following.

Definition 11.13 (Zero-stability of multistep methods) The multistep method (11.45) is zero-stable if

$$\exists h_0 > 0, \ \exists C > 0 : \quad \forall h \in (0, h_0], \ |z_n^{(h)} - u_n^{(h)}| \leq C\varepsilon, \ 0 \leq n \leq N_h, \tag{11.58}$$

where $N_h = \max\{n : \ t_n \leq t_0 + T\}$ and $z_n^{(h)}$ and $u_n^{(h)}$ are, respectively, the solutions of problems

$$\begin{cases} z_{n+1}^{(h)} = \displaystyle\sum_{j=0}^{p} a_j z_{n-j}^{(h)} + h \sum_{j=-1}^{p} b_j f(t_{n-j}, z_{n-j}^{(h)}) + h\delta_{n+1}, \\ z_k^{(h)} = w_k^{(h)} + \delta_k, \qquad k = 0, \ldots, p \end{cases} \tag{11.59}$$

$$\begin{cases} u_{n+1}^{(h)} = \displaystyle\sum_{j=0}^{p} a_j u_{n-j}^{(h)} + h \sum_{j=-1}^{p} b_j f(t_{n-j}, u_{n-j}^{(h)}), \\ u_k^{(h)} = w_k^{(h)}, \qquad k = 0, \dots, p \end{cases} \tag{11.60}$$

for $p \le n \le N_h - 1$, where $|\delta_k| \le \varepsilon$, $0 \le k \le N_h$, $w_0^{(h)} = y_0$ and $w_k^{(h)}$, $k = 1, \dots, p$, are p initial values generated by using another numerical scheme. ■

Theorem 11.4 (Equivalence of zero-stability and root condition) *For a consistent multistep method, the root condition is equivalent to zero-stability.*

Proof. Let us begin by proving that the root condition is *necessary* for the zero-stability to hold. We proceed by contradiction and assume that the method is zero-stable and there exists a root r_i which violates the root condition.

Since the method is zero-stable, condition (11.58) must be satisfied for *any* Cauchy problem, in particular for the problem $y'(t) = 0$ with $y(0) = 0$, whose solution is, clearly, the null function. Similarly, the solution $u_n^{(h)}$ of (11.60) with $f = 0$ and $w_k^{(h)} = 0$ for $k = 0, \dots, p$ is identically zero.

Consider first the case $|r_i| > 1$. Then, define

$$\delta_n = \begin{cases} \varepsilon r_i^n & \text{if } r_i \in \mathbb{R}, \\ \varepsilon (r_i + \bar{r}_i)^n & \text{if } r_i \in \mathbb{C}, \end{cases}$$

for $\varepsilon > 0$. It is simple to check that the sequence $z_n^{(h)} = \delta_n$ for $n = 0, 1, \dots$ is a solution of (11.59) with initial conditions $z_k^{(h)} = \delta_k$ and that $|\delta_k| \le \varepsilon$ for $k = 0, 1, \dots, p$. Let us now choose $\bar{t}$ in $(t_0, t_0 + T)$ and let x_n be the nearest grid node to $\bar{t}$. Clearly, n is the integral part of $\bar{t}/h$ and $\lim_{h\to 0} |z_n^{(h)}| = \lim_{h\to 0} |u_n^{(h)} - z_n^{(h)}| \to \infty$ as $h \to 0$. This proves that $|u_n^{(h)} - z_n^{(h)}|$ cannot be uniformly bounded with respect to h as $h \to 0$, which contradicts the assumption that the method is zero-stable.

A similar proof can be carried out if $|r_i| = 1$ but has multiplicity greater than 1, provided that one takes into account the form of the solution (11.33).

Let us now prove that the root condition is sufficient for method (11.45) to be zero-stable. Recalling (11.46) and denoting by $z_{n+j}^{(h)}$ and $u_{n+j}^{(h)}$ the solutions to (11.59) and (11.60), respectively, for $j \ge 1$, it turns out that the function $w_{n+j}^{(h)} = z_{n+j}^{(h)} - u_{n+j}^{(h)}$ satisfies the following difference equation

$$\sum_{j=0}^{p+1} \alpha_j w_{n+j}^{(h)} = \varphi_{n+p+1}, \quad n = 0, \dots, N_h - (p+1), \tag{11.61}$$

having set

$$\varphi_{n+p+1} = h \sum_{j=0}^{p+1} \beta_j \left[f(t_{n+j}, z_{n+j}^{(h)}) - f(t_{n+j}, u_{n+j}^{(h)}) \right] + h\delta_{n+p+1}. \tag{11.62}$$

Denote by $\left\{\psi_j^{(n)}\right\}$ a sequence of fundamental solutions to the homogeneous equation associated with (11.61). Recalling (11.42), the general solution of (11.61) is given by

$$w_n^{(h)} = \sum_{j=0}^{p} w_j^{(h)} \psi_j^{(n)} + \sum_{l=p+1}^{n} \psi_p^{(n-l+p)} \varphi_l, \quad n = p+1, \dots$$

The following result expresses the connection between the root condition and the boundedness of the solution of a difference equation (for the proof, see [Gau97], Theorem 6.3.2).

Lemma 11.3 *There exists a constant $M > 0$ for any solution $\{u_n\}$ of the difference equation* (11.28) *such that*

$$|u_n| \le M \left\{ \max_{j=0,\dots,k-1} |u_j| + \sum_{l=k}^{n} |\varphi_l| \right\}, \quad n = 0, 1, \dots \tag{11.63}$$

iff the root condition is satisfied for the polynomial (11.30)*, i.e.,* (11.56) *holds for the zeros of the polynomial* (11.30)*.*

Let us now recall that, for any j, $\{\psi_j^{(n)}\}$ is solution of a homogeneous difference equation whose initial data are $\psi_j^{(i)} = \delta_{ij}$, $i, j = 0, \dots, p$. On the other hand, for any l, $\psi_p^{(n-l+p)}$ is solution of a difference equation with zero initial conditions and right-hand sides equal to zero except for the one corresponding to $n = l$ which is $\psi_p^{(p)} = 1$.

Therefore, Lemma 11.3 can be applied in both cases so we can conclude that there exists a constant $M > 0$ such that $|\psi_j^{(n)}| \le M$ and $|\psi_p^{(n-l+p)}| \le M$, uniformly with respect to n and l. The following estimate thus holds

$$|w_n^{(h)}| \le M \left\{ (p+1) \max_{j=0,\dots,p} |w_j^{(h)}| + \sum_{l=p+1}^{n} |\varphi_l| \right\}, \quad n = 0, 1, \dots, N_h. \tag{11.64}$$

If L denotes the Lipschitz constant of f, from (11.62) we get

$$|\varphi_{n+p+1}| \le h \max_{j=0,\dots,p+1} |\beta_j| L \sum_{j=0}^{p+1} |w_{n+j}^{(h)}| + h|\delta_{n+p+1}|.$$

Let $\beta = \max_{j=0,\dots,p+1} |\beta_j|$ and $\Delta_{[q,r]} = \max_{j=q,\dots,r} |\delta_{j+q}|$, q and r being some integers with $q \le r$. From (11.64), the following estimate is therefore obtained

$$\begin{aligned} |w_n^{(h)}| &\le M \left\{ (p+1)\Delta_{[0,p]} + h\beta L \sum_{l=p+1}^{n} \sum_{j=0}^{p+1} |w_{l-p-1+j}^{(h)}| + N_h h \Delta_{[p+1,n]} \right\} \\ &\le M \left\{ (p+1)\Delta_{[0,p]} + h\beta L (p+2) \sum_{m=0}^{n} |w_m^{(h)}| + T\Delta_{[p+1,n]} \right\}. \end{aligned}$$

Let $Q = 2(p+2)\beta LM$ and $h_0 = 1/Q$, so that $1 - h\frac{Q}{2} \geq \frac{1}{2}$ if $h \leq h_0$. Then

$$\begin{aligned}\frac{1}{2}|w_n^{(h)}| &\leq |w_n^{(h)}|(1 - h\tfrac{Q}{2}) \\ &\leq M\left\{(p+1)\Delta_{[0,p]} + h\beta L(p+2)\sum_{m=0}^{n-1}|w_m^{(h)}| + T\Delta_{[p+1,n]}\right\}.\end{aligned}$$

Letting $R = 2M\left\{(p+1)\Delta_{[0,p]} + T\Delta_{[p+1,n]}\right\}$, we finally obtain

$$|w_n^{(h)}| \leq hQ\sum_{m=0}^{n-1}|w_m^{(h)}| + R.$$

Applying Lemma 11.2 with the following identifications: $\varphi_n = |w_n^{(h)}|$, $g_0 = R$, $p_s = 0$ and $k_s = hQ$ for any $s = 0, \ldots, n-1$, yields

$$|w_n^{(h)}| \leq 2Me^{TQ}\left\{(p+1)\Delta_{[0,p]} + T\Delta_{[p+1,n]}\right\}. \tag{11.65}$$

Method (11.45) is thus zero-stable for any $h \leq h_0$. ◇

Theorem 11.4 allows for characterizing the stability behavior of several families of discretization methods.

In the special case of consistent one-step methods, the polynomial ρ admits only the root $r_0 = 1$. They thus *automatically satisfy the root condition* and are zero-stable.

For the Adams methods (11.49), the polynomial ρ is always of the form $\rho(r) = r^{p+1} - r^p$. Thus, its roots are $r_0 = 1$ and $r_1 = 0$ (with multiplicity p) so that all Adams methods are zero-stable.

Also the midpoint method (11.43) and Simpson method (11.44) are zero-stable: for both of them, the first characteristic polynomial is $\rho(r) = r^2 - 1$, so that $r_0 = 1$ and $r_1 = -1$.

Finally, the BDF methods of Section 11.5.2 are zero-stable provided that $p \leq 5$, since in such a case the root condition is satisfied (see [Cry73]).

We are in position to give the following convergence result.

Theorem 11.5 (Convergence) *A consistent multistep method is convergent iff it satisfies the root condition and the error on the initial data tends to zero as $h \to 0$. Moreover, the method converges with order q if it has order q and the error on the initial data tends to zero as $\mathcal{O}(h^q)$.*

Proof. Suppose that the MS method is consistent and convergent. To prove that the root condition is satisfied, we refer to the problem $y'(t) = 0$ with $y(0) = 0$ and on the interval $I = (0, T)$. Convergence means that the numerical solution $\{u_n\}$ must tend to the exact solution $y(t) = 0$ for any converging set of initial data u_k, $k = 0, \ldots, p$, i.e. $\max_{k=0,\ldots,p}|u_k| \to 0$ as $h \to 0$. From this observation, the proof follows by contradiction along the same lines as the proof of Theorem 11.4, where the parameter ε is now replaced by h.

Let us now prove that consistency, together with the root condition, implies convergence under the assumption that the error on the initial data tends to zero as $h \to 0$. We can apply Theorem 11.4, setting $u_n^{(h)} = u_n$ (approximate solution of the Cauchy problem) and $z_n^{(h)} = y_n$ (exact solution), and from (11.47) it turns out that $\delta_m = \tau_m(h)$. Then, due to (11.65), for any $n \geq p+1$ we obtain

$$|u_n - y_n| \leq 2Me^{TQ} \left\{ (p+1) \max_{j=0,\dots,p} |u_j - y_j| + T \max_{j=p+1,\dots,n} |\tau_j(h)| \right\}.$$

Convergence holds by noticing that the right-hand side of this inequality tends to zero as $h \to 0$. ◇

A remarkable consequence of the above theorem is the following equivalence Lax-Richtmyer theorem.

Corollary 11.1 (Equivalence theorem) *A consistent multistep method is convergent iff it is zero-stable and if the error on the initial data tends to zero as h tends to zero.*

We conclude this section with the following result, which establishes an upper limit for the order of multistep methods (see [Dah63]).

Property 11.1 (First Dahlquist barrier) *There isn't any zero-stable, p-step linear multistep method with order greater than $p+1$ if p is odd, $p+2$ if p is even.*

11.6.4 Absolute Stability of Multistep Methods

Consider again the difference equation (11.54), which was obtained by applying the MS method (11.45) to the model problem (11.24). According to (11.33), its solution takes the form

$$u_n = \sum_{j=1}^{k'} \left(\sum_{s=0}^{m_j - 1} \gamma_{sj} n^s \right) [r_j(h\lambda)]^n, \qquad n = 0, 1, \dots$$

where $r_j(h\lambda)$, $j = 1, \dots, k'$, are the distinct roots of the characteristic polynomial (11.55), and having denoted by m_j the multiplicity of $r_j(h\lambda)$. In view of (11.25), it is clear that the *absolute root condition* introduced by Definition 11.12 is necessary and sufficient to ensure that the multistep method (11.45) is absolutely stable as $h \leq h_0$.

Among the methods enjoying the absolute stability property, the preference should go to those for which the region of absolute stability $\mathcal{A}$, introduced in (11.26), is as wide as possible or even unbounded. Among these are the *A-stable* methods introduced at the end of Section 11.3.3 and

the ϑ-*stable* methods, for which $\mathcal{A}$ contains the angular region defined by $z \in \mathbb{C}$ such that $-\vartheta < \pi - \arg(z) < \vartheta$, with $\vartheta \in (0, \pi/2)$. A-stable methods are of remarkable importance when solving *stiff* problems (see Section 11.10).

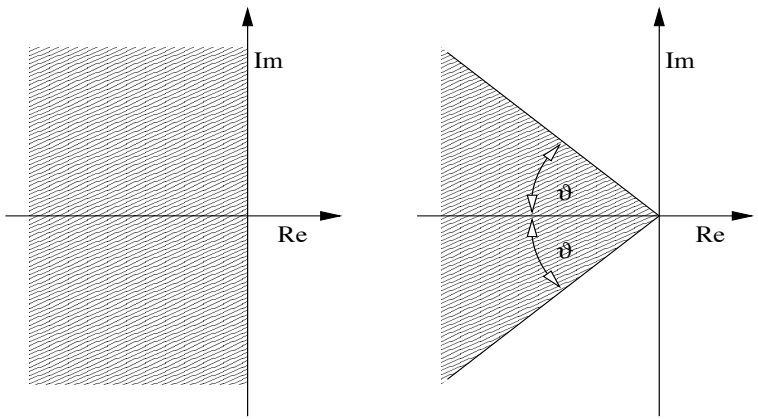

FIGURE 11.4. Regions of absolute stability for A-stable (left) and (right) ϑ-stable methods

The following result, whose proof is given in [Wid67], establishes a relation between the order of a multistep method, the number of its steps and its stability properties.

Property 11.2 (Second Dahlquist barrier) *A linear explicit multistep method can be neither A-stable, nor ϑ-stable. Moreover, there is no A-stable linear multistep method with order greater than 2. Finally, for any $\vartheta \in (0, \pi/2)$, there only exist ϑ-stable p-step linear multistep methods of order p for $p = 3$ and $p = 4$.*

Let us now examine the region of absolute stability of several MS methods.

The regions of absolute stability of both explicit and implicit Adams schemes reduce progressively as the order of the method increases. In Figure 11.5 (left) we show the regions of absolute stability for the AB methods examined in Section 11.5.1, with exception of the Forward Euler method whose region is shown in Figura 11.3.

The regions of absolute stability of the Adams-Moulton schemes, except for the Crank-Nicolson method which is A-stable, are represented in Figure 11.5 (right).

In Figure 11.6 the regions of absolute stability of some of the BDF methods introduced in Section 11.5.2 are drawn. They are unbounded and al-

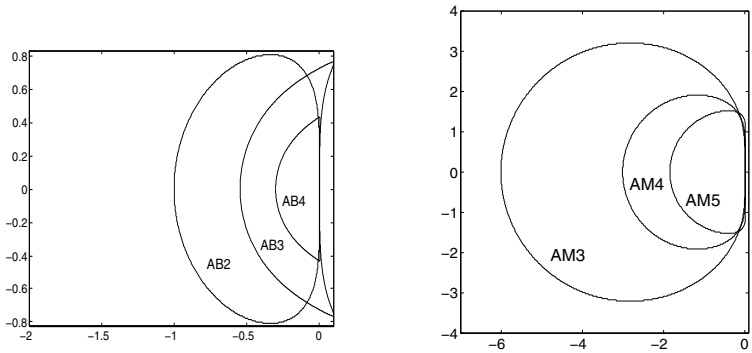

FIGURE 11.5. Outer contours of the regions of absolute stability for Adams-Bashforth methods (left) ranging from second to fourth-order (AB2, AB3 and AB4) and for Adams-Moulton methods (right), from third to fifth-order (AM3, AM4 and AM5). Notice that the region of the AB3 method extends into the half-plane with positive real part. The region for the explicit Euler (AB1) method was drawn in Figure 11.3

ways contain the negative real numbers. These stability features make BDF methods quite attractive for solving *stiff* problems (see Section 11.10).

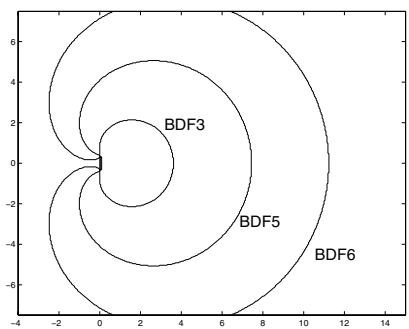

FIGURE 11.6. Inner contours of regions of absolute stability for three-step (BDF3), five-step (BDF5) and six-step (BDF6) BDF methods. Unlike Adams methods, these regions are unbounded and extend outside the limited portion that is shown in the figure

Remark 11.3 Some authors (see, e.g., [BD74]) adopt an alternative definition of absolute stability by replacing (11.25) with the milder property

$$\exists C > 0 \, : \, |u_n| \leq C, \text{ as } t_n \to +\infty.$$

According to this new definition, the absolute stability of a numerical method should be regarded as the counterpart of the *asymptotic* stabil-

ity (11.6) of the Cauchy problem. The new region of absolute stability $\mathcal{A}^*$ would be

$$\mathcal{A}^* = \{z \in \mathbb{C} : \ \exists C > 0, \ |u_n| \leq C, \ \forall n \geq 0\}$$

and it would not necessarily coincide with $\mathcal{A}$. For example, in the case of the midpoint method $\mathcal{A}$ is empty (thus, it is unconditionally absolutely *unstable*), while $\mathcal{A}^* = \{z = \alpha i, \ \alpha \in [-1, 1]\}$.

In general, if $\mathcal{A}$ is nonempty, then $\mathcal{A}^*$ is its closure. We notice that *zero-stable methods are those for which the region $\mathcal{A}^*$ contains the origin $z = 0$ of the complex plane.* ■

To conclude, let us notice that the strong root condition (11.57) implies, for a linear problem, that

$$\forall h \leq h_0, \ \exists C > 0 \ : \ |u_n| \leq C(|u_0| + \ldots + |u_p|), \quad \forall n \geq p + 1. \quad (11.66)$$

We say that a method is *relatively stable* if it satisfies (11.66). Clearly, (11.66) implies zero-stability, but the converse does not hold.
Figure 11.7 summarizes the main conclusions that have been drawn in this section about stability, convergence and root-conditions, in the particular case of a consistent method applied to the model problem (11.24).

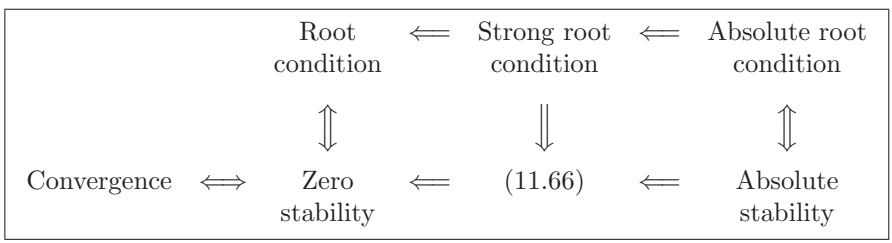

FIGURE 11.7. Relations between root conditions, stability and convergence for a consistent method applied to the model problem (11.24)

11.7 Predictor-Corrector Methods

When solving a nonlinear Cauchy problem of the form (11.1), at each time step implicit schemes require dealing with a nonlinear equation. For instance, if the Crank-Nicolson method is used, we get the nonlinear equation

$$u_{n+1} = u_n + \frac{h}{2}\left[f_n + f_{n+1}\right] = \Psi(u_{n+1}),$$

that can be cast in the form $\Phi(u_{n+1}) = 0$, where $\Phi(u_{n+1}) = u_{n+1} - \Psi(u_{n+1})$. To solve this equation the Newton method would give

$$u_{n+1}^{(k+1)} = u_{n+1}^{(k)} - \Phi(u_{n+1}^{(k)})/\Phi'(u_{n+1}^{(k)}),$$

for $k = 0, 1, \ldots$, until convergence and require an initial datum $u_{n+1}^{(0)}$ sufficiently close to u_{n+1}. Alternatively, one can resort to fixed-point iterations

$$u_{n+1}^{(k+1)} = \Psi(u_{n+1}^{(k)}) \tag{11.67}$$

for $k = 0, 1, \ldots$, until convergence. In such a case, the global convergence condition for the fixed-point method (see Theorem 6.1) sets a constraint on the discretization stepsize of the form

$$h < \frac{1}{|b_{-1}|L} \tag{11.68}$$

where L is the Lipschitz constant of f with respect to y. In practice, except for the case of stiff problems (see Section 11.10), this restriction on h is not significant since considerations of accuracy put a much more restrictive constraint on h. However, each iteration of (11.67) requires one evaluation of the function f and the computational cost can be reduced by providing a good initial guess $u_{n+1}^{(0)}$. This can be done by taking one step of an explicit MS method and then iterating on (11.67) for a fixed number m of iterations. By doing so, the implicit MS method that is employed in the fixed-point scheme "corrects" the value of u_{n+1} "predicted" by the explicit MS method. A procedure of this sort is called a *predictor-corrector method*, or PC method. There are many ways in which a predictor-corrector method can be implemented.

In its basic version, the value $u_{n+1}^{(0)}$ is computed by an explicit $\tilde{p}+1$-step method, called the *predictor* (here identified by the coefficients $\{\tilde{a}_j, \tilde{b}_j\}$)

$$[P] \quad u_{n+1}^{(0)} = \sum_{j=0}^{\tilde{p}} \tilde{a}_j u_{n-j}^{(1)} + h \sum_{j=0}^{\tilde{p}} \tilde{b}_j f_{n-j}^{(0)},$$

where $f_k^{(0)} = f(t_k, u_k^{(0)})$ and $u_k^{(1)}$ are the solutions computed by the PC method at the previous steps or are the initial conditions. Then, we evaluate the function f at the new point $(t_{n+1}, u_{n+1}^{(0)})$ (*evaluation step*)

$$[E] \quad f_{n+1}^{(0)} = f(t_{n+1}, u_{n+1}^{(0)}),$$

and finally, one single fixed-point iteration is carried out using an implicit MS scheme of the form (11.45)

$$[C] \quad u_{n+1}^{(1)} = \sum_{j=0}^{p} a_j u_{n-j}^{(1)} + h b_{-1} f_{n+1}^{(0)} + h \sum_{j=0}^{p} b_j f_{n-j}^{(0)}.$$

This second step of the procedure, which is actually explicit, is called the *corrector*. The overall procedure is shortly denoted by PEC or $P(EC)^1$ method, in which P and C denote one application at time t_{n+1} of the predictor and the corrector methods, respectively, while E indicates one evaluation of the function f.

This strategy above can be generalized supposing to perform $m > 1$ iterations at each step t_{n+1}. The corresponding methods are called *predictor-multicorrector* schemes and compute $u_{n+1}^{(0)}$ at time step t_{n+1} using the predictor in the following form

$$[P] \quad u_{n+1}^{(0)} = \sum_{j=0}^{\tilde{p}} \tilde{a}_j u_{n-j}^{(m)} + h \sum_{j=0}^{\tilde{p}} \tilde{b}_j f_{n-j}^{(m-1)}. \tag{11.69}$$

Here $m \geq 1$ denotes the (fixed) number of corrector iterations that are carried out in the following steps $[E]$, $[C]$: for $k = 0, 1, \ldots, m-1$

$$[E] \quad f_{n+1}^{(k)} = f(t_{n+1}, u_{n+1}^{(k)}),$$

$$[C] \quad u_{n+1}^{(k+1)} = \sum_{j=0}^{p} a_j u_{n-j}^{(m)} + h b_{-1} f_{n+1}^{(k)} + h \sum_{j=0}^{p} b_j f_{n-j}^{(m-1)}.$$

These implementations of the predictor-corrector technique are referred to as $P(EC)^m$. Another implementation, denoted by $P(EC)^m E$, consists of updating at the end of the process also the function f and is given by

$$[P] \quad u_{n+1}^{(0)} = \sum_{j=0}^{\tilde{p}} \tilde{a}_j u_{n-j}^{(m)} + h \sum_{j=0}^{\tilde{p}} \tilde{b}_j f_{n-j}^{(m)},$$

and for $k = 0, 1, \ldots, m-1$,

$$[E] \quad f_{n+1}^{(k)} = f(t_{n+1}, u_{n+1}^{(k)}),$$

$$[C] \quad u_{n+1}^{(k+1)} = \sum_{j=0}^{p} a_j u_{n-j}^{(m)} + h b_{-1} f_{n+1}^{(k)} + h \sum_{j=0}^{p} b_j f_{n-j}^{(m)},$$

followed by

$$[E] \quad f_{n+1}^{(m)} = f(t_{n+1}, u_{n+1}^{(m)}).$$

Example 11.8 Heun's method (11.10) can be regarded as a predictor-corrector method whose predictor is the forward Euler method, while the corrector is the Crank-Nicolson method.

Another example is provided by the Adams-Bashforth method of order 2 (11.50) and the Adams-Moulton method of order 3 (11.51). Its corresponding

PEC implementation is: given $u_0^{(0)} = u_0^{(1)} = u_0$, $u_1^{(0)} = u_1^{(1)} = u_1$ and $f_0^{(0)} = f(t_0, u_0^{(0)})$, $f_1^{(0)} = f(t_1, u_1^{(0)})$, compute for $n = 1, 2, \dots$,

$$
\begin{aligned}
&[P] \quad u_{n+1}^{(0)} = u_n^{(1)} + \frac{h}{2}\left[3f_n^{(0)} - f_{n-1}^{(0)}\right], \\
&[E] \quad f_{n+1}^{(0)} = f(t_{n+1}, u_{n+1}^{(0)}), \\
&[C] \quad u_{n+1}^{(1)} = u_n^{(1)} + \frac{h}{12}\left[5f_{n+1}^{(0)} + 8f_n^{(0)} - f_{n-1}^{(0)}\right],
\end{aligned}
$$

while the $PECE$ implementation is: given $u_0^{(0)} = u_0^{(1)} = u_0$, $u_1^{(0)} = u_1^{(1)} = u_1$ and $f_0^{(1)} = f(t_0, u_0^{(1)})$, $f_1^{(1)} = f(t_1, u_1^{(1)})$, compute for $n = 1, 2, \dots$,

$$
\begin{aligned}
&[P] \quad u_{n+1}^{(0)} = u_n^{(1)} + \frac{h}{2}\left[3f_n^{(1)} - f_{n-1}^{(1)}\right], \\
&[E] \quad f_{n+1}^{(0)} = f(t_{n+1}, u_{n+1}^{(0)}), \\
&[C] \quad u_{n+1}^{(1)} = u_n^{(1)} + \frac{h}{12}\left[5f_{n+1}^{(0)} + 8f_n^{(1)} - f_{n-1}^{(1)}\right], \\
&[E] \quad f_{n+1}^{(1)} = f(t_{n+1}, u_{n+1}^{(1)}).
\end{aligned}
$$

•

Before studying the convergence of predictor-corrector methods, we introduce a simplification in the notation. Usually the number of steps of the predictor is greater than those of the corrector, so that we define the number of steps of the predictor-corrector pair as being equal to the number of steps of the predictor. This number will be denoted henceforth by p. Owing to this definition we no longer demand that the coefficients of the corrector satisfy $|a_p| + |b_p| \neq 0$. Consider for example the predictor-corrector pair

$$
\begin{aligned}
&[P] \quad u_{n+1}^{(0)} = u_n^{(1)} + hf(t_{n-1}, u_{n-1}^{(0)}), \\
&[C] \quad u_{n+1}^{(1)} = u_n^{(1)} + \tfrac{h}{2}\left[f(t_n, u_n^{(0)}) + f(t_{n+1}, u_{n+1}^{(0)})\right],
\end{aligned}
$$

for which $p = 2$ (even though the corrector is a one-step method). Consequently, the first and the second characteristic polynomials of the corrector method will be $\rho(r) = r^2 - r$ and $\sigma(r) = (r^2 + r)/2$ instead of $\rho(r) = r - 1$ and $\sigma(r) = (r + 1)/2$.

In any predictor-corrector method, the truncation error of the *predictor* combines with the one of the *corrector*, generating a new truncation error which we are going to examine. Let $\tilde{q}$ and q be, respectively, the orders of the predictor and the corrector and assume that $y \in C^{\widehat{q}+1}$, where $\widehat{q} = \max(\tilde{q}, q)$.

Then

$$\begin{aligned} y(t_{n+1}) &- \sum_{j=0}^{p} \tilde{a}_j y(t_{n-j}) - h \sum_{j=0}^{p} \tilde{b}_j f(t_{n-j}, y_{n-j}) \\ &= \tilde{C}_{\tilde{q}+1} h^{\tilde{q}+1} y^{(\tilde{q}+1)}(t_n) + \mathcal{O}(h^{\tilde{q}+2}), \end{aligned}$$

$$\begin{aligned} y(t_{n+1}) &- \sum_{j=0}^{p} a_j y(t_{n-j}) - h \sum_{j=-1}^{p} b_j f(t_{n-j}, y_{n-j}) \\ &= C_{q+1} h^{q+1} y^{(q+1)}(t_n) + \mathcal{O}(h^{q+2}), \end{aligned}$$

where $\tilde{C}_{\tilde{q}+1}, C_{q+1}$ are the error constants of the predictor and the corrector method respectively. The following result holds.

Property 11.3 *Let the predictor method have order $\tilde{q}$ and the corrector method have order q. Then:*

If $\tilde{q} \geq q$ (or $\tilde{q} < q$ with $m > q - \tilde{q}$), then the predictor-corrector method has the same order and the same PLTE as the corrector.

If $\tilde{q} < q$ and $m = q - \tilde{q}$, then the predictor-corrector method has the same order as the corrector, but different PLTE.

If $\tilde{q} < q$ and $m \leq q - \tilde{q} - 1$, then the predictor-corrector method has order equal to $\tilde{q} + m$ (thus less than q).

In particular, notice that if the predictor has order $q-1$ and the corrector has order q, the PEC suffices to get a method of order q. Moreover, the $P(EC)^m E$ and $P(EC)^m$ schemes have always the same order and the same PLTE.

Combining the Adams-Bashforth method of order q with the corresponding Adams-Moulton method of the same order we obtain the so-called ABM method of order q. It is possible to estimate its PLTE as

$$\frac{C_{q+1}}{C^*_{q+1} - C_{q+1}} \left(u^{(m)}_{n+1} - u^{(0)}_{n+1} \right),$$

where C_{q+1} and C^*_{q+1} are the error constants given in Table 11.1. Accordingly, the steplength h can be decreased if the estimate of the PLTE exceeds a given tolerance and increased otherwise (for the adaptivity of the step length in a predictor-corrector method, see [Lam91], pp.128–147).

Program 93 provides an implementation of the $P(EC)^m E$ methods. The input parameters `at`, `bt`, `a`, `b` contain the coefficients $\tilde{a}_j, \tilde{b}_j$ $(j = 0, \ldots, \tilde{p})$ of the predictor and the coefficients a_j $(j = 0, \ldots, p)$, b_j $(j = -1, \ldots, p)$ of the corrector. Moreover, `f` is a string containing the expression of $f(t, y)$,

`h` is the stepsize, `t0` and `tf` are the end points of the time integration interval, `u0` is the vector of the initial data, `m` is the number of the corrector inner iterations. The input variable `pece` must be set equal to `'y'` if the $P(EC)^m E$ is selected, conversely the $P(EC)^m$ scheme is chosen.

Program 93 - predcor : Predictor-corrector scheme

```
function [u,t]=predcor(a,b,at,bt,h,f,t0,u0,tf,pece,m)
p  = max(length(a),length(b)-1); pt = max(length(at),length(bt));
q  = max(p,pt); if length(u0) < q, break, end;
t  = [t0:h:t0+(q-1)*h]; u = u0; y = u0; fe = eval(f);
k  = q;
for t = t0+q*h:h:tf
   ut = sum(at.*u(k:-1:k-pt+1))+h*sum(bt.*fe(k:-1:k-pt+1));
   y  = ut; foy = eval(f);
   uv = sum(a.*u(k:-1:k-p+1))+h*sum(b(2:p+1).*fe(k:-1:k-p+1));
   k = k+1;
   for j = 1:m
      fy = foy; up = uv + h*b(1)*fy; y  = up; foy = eval(f);
   end
   if (pece=='y'|pece=='Y')
      fe = [fe, foy];
   else
      fe = [fe, fy];
   end
   u = [u, up];
end
t = [t0:h:tf];
```

Example 11.9 Let us check the performance of the $P(EC)^m E$ method on the Cauchy problem $y'(t) = e^{-y(t)}$ for $t \in [0,1]$ with $y(0) = 1$. The exact solution is $y(t) = \log(1+t)$. In all the numerical experiments, the corrector method is the Adams-Moulton third-order scheme (AM3), while the explicit Euler (AB1) and the Adams-Bashforth second-order (AB2) methods are used as predictors. Figure 11.8 shows that the pair AB2-AM3 ($m = 1$) yields third-order convergence rate, while AB1-AM3 ($m = 1$) has a first-order accuracy. Taking $m = 2$ allows to recover the third-order convergence rate of the corrector for the AB1-AM3 pair. •

As for the absolute stability, the characteristic polynomial of $P(EC)^m$ methods reads

$$\Pi_{P(EC)^m}(r) = b_{-1} r^p \left(\widehat{\rho}(r) - h\lambda\widehat{\sigma}(r)\right) + \frac{H^m(1-H)}{1-H^m}\left(\tilde{\rho}(r)\widehat{\sigma}(r) - \widehat{\rho}(r)\tilde{\sigma}(r)\right)$$

while for $P(EC)^m E$ we have

$$\Pi_{P(EC)^m E}(r) = \widehat{\rho}(r) - h\lambda\widehat{\sigma}(r) + \frac{H^m(1-H)}{1-H^m}\left(\tilde{\rho}(r) - h\lambda\tilde{\sigma}(r)\right).$$

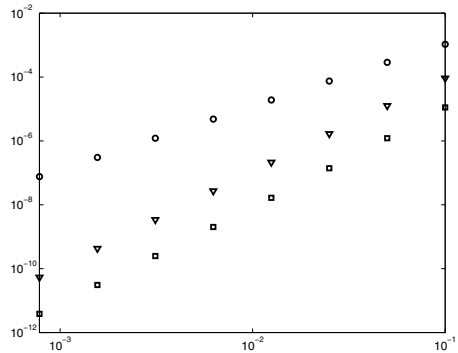

FIGURE 11.8. Convergence rate for $P(EC)^m E$ methods as a function of $\log(h)$. The symbol ∇ refers to the AB2-AM3 method ($m = 1$), $\circ$ to AB1-AM3 ($m = 1$) and $\square$ to AB1-AM3 with $m = 2$

We have set $H = h\lambda b_{-1}$ and denoted by $\tilde{\rho}$ and $\tilde{\sigma}$ the first and second characteristic polynomial of the *predictor* method, respectively. The polynomials $\widehat{\rho}$ and $\widehat{\sigma}$ are related to the first and second characteristic polynomials of the corrector, as previously explained after Example 11.8. Notice that in both cases the characteristic polynomial tends to the corresponding polynomial of the *corrector* method, since the function $H^m(1-H)/(1-H^m)$ tends to zero as m tends to infinity.

Example 11.10 If we consider the ABM methods with a number of steps p, the characteristic polynomials are $\widehat{\rho}(r) = \tilde{\rho}(r) = r(r^{p-1} - r^{p-2})$, while $\widehat{\sigma}(r) = r\sigma(r)$, where $\sigma(r)$ is the second characteristic polynomial of the corrector. In Figure 11.9 (right) the stability regions for the ABM methods of order 2 are plotted. In the case of the ABM methods of order 2, 3 and 4, the corresponding stability regions can be ordered by size, namely, from the largest to the smallest one the regions of $PECE$, $P(EC)^2E$, the predictor and PEC methods are plotted in Figure 11.9, left. The one-step ABM method is an exception to the rule and the largest region is the one corresponding to the predictor method (see Figure 11.9, left). •

11.8 Runge-Kutta (RK) Methods

When evolving from the forward Euler method (11.7) toward higher-order methods, linear multistep methods (MS) and Runge-Kutta methods (RK) adopt two opposite strategies.

Like the Euler method, MS schemes are linear with respect to both u_n and $f_n = f(t_n, u_n)$, require only one functional evaluation at each time step and their accuracy can be increased at the expense of increasing the number of steps. On the other hand, RK methods maintain the structure of one-step methods, and increase their accuracy at the price of an increase of functional evaluations at each time level, thus sacrifying linearity.

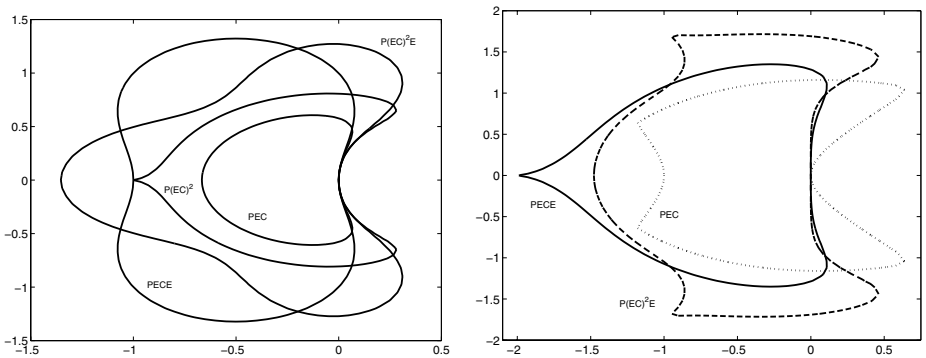

FIGURE 11.9. Stability regions for the ABM methods of order 1 (left) and 2 (right)

A consequence is that RK methods are more suitable than MS methods at adapting the stepsize, whereas estimating the local error for RK methods is more difficult than it is in the case of MS methods.
In its most general form, an RK method can be written as

$$u_{n+1} = u_n + hF(t_n, u_n, h; f), \qquad n \geq 0 \tag{11.70}$$

where F is the increment function defined as follows

$$\begin{aligned} F(t_n, u_n, h; f) &= \sum_{i=1}^{s} b_i K_i, \\ K_i &= f(t_n + c_i h, u_n + h\sum_{j=1}^{s} a_{ij} K_j), \quad i = 1, 2, \ldots, s \end{aligned} \tag{11.71}$$

and s denotes the number of *stages* of the method. The coefficients $\{a_{ij}\}$, $\{c_i\}$ and $\{b_i\}$ fully characterize an RK method and are usually collected in the so-called *Butcher array*

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \ldots & a_{1s} \\ c_2 & a_{21} & a_{22} & & a_{2s} \\ \vdots & \vdots & & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \ldots & a_{ss} \\ \hline & b_1 & b_2 & \ldots & b_s \end{array} \qquad \text{or} \qquad \begin{array}{c|c} \mathbf{c} & \mathrm{A} \\ \hline & \mathbf{b}^T \end{array}$$

where $\mathrm{A} = (a_{ij}) \in \mathbb{R}^{s\times s}$, $\mathbf{b} = (b_1, \ldots, b_s)^T \in \mathbb{R}^s$ and $\mathbf{c} = (c_1, \ldots, c_s)^T \in \mathbb{R}^s$. We shall henceforth assume that the following condition holds

$$c_i = \sum_{j=1}^{s} a_{ij} \quad i = 1, \ldots, s. \tag{11.72}$$

If the coefficients a_{ij} in A are equal to zero for $j \geq i$, with $i = 1, 2, \ldots, s$, then each K_i can be explicitly computed in terms of the $i-1$ coefficients $K_1, \ldots, K_{i-1}$ that have already been determined. In such a case the RK method is *explicit*. Otherwise, it is *implicit* and solving a nonlinear system of size s is necessary for computing the coefficients K_i.

The increase in the computational effort for implicit schemes makes their use quite expensive; an acceptable compromise is provided by RK *semi-implicit* methods, in which case $a_{ij} = 0$ for $j > i$ so that each K_i is the solution of the nonlinear equation

$$K_i = f\left(t_n + c_i h, u_n + h a_{ii} \boxed{K_i} + h \sum_{j=1}^{i-1} a_{ij} K_j\right).$$

A semi-implicit scheme thus requires s nonlinear independent equations to be solved.

The local truncation error $\tau_{n+1}(h)$ at node t_{n+1} of the RK method (11.70) is defined through the residual equation

$$h\tau_{n+1}(h) = y_{n+1} - y_n - hF(t_n, y_n, h; f),$$

where $y(t)$ is the exact solution to the Cauchy problem (11.1). Method (11.70) is *consistent* if $\tau(h) = \max_n |\tau_n(h)| \to 0$ as $h \to 0$. It can be shown (see [Lam91]) that this happens iff

$$\sum_{i=1}^{s} b_i = 1.$$

As usual, we say that (11.70) is a method of order p (≥ 1) with respect to h if $\tau(h) = \mathcal{O}(h^p)$ as $h \to 0$.

As for *convergence*, since RK methods are one-step methods, consistency implies stability and, in turn, convergence. As happens for MS methods, estimates of $\tau(h)$ can be derived; however, these estimates are often too complicated to be profitably used. We only mention that, as for MS methods, if an RK scheme has a local truncation error $\tau_n(h) = \mathcal{O}(h^p)$, for any n, then also the convergence order will be equal to p.

The following result establishes a relation between order and number of stages of explicit RK methods.

Property 11.4 *The order of an s-stage explicit RK method cannot be greater than s. Also, there do not exist s-stage explicit RK methods with order $s \geq 5$.*

We refer the reader to [But87] for the proofs of this result and the results we give below. In particular, for orders ranging between 1 and 10, the minimum

number of stages s_{min} required to get a method of corresponding order is shown below

order	1	2	3	4	**5**	6	7	8
s_{min}	1	2	3	4	**6**	7	9	11

Notice that 4 is the maximum number of stages for which the order of the method is not less than the number of stages itself. An example of a fourth-order RK method is provided by the following explicit 4-stage method

$$\begin{aligned} u_{n+1} &= u_n + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\ K_1 &= f_n, \\ K_2 &= f(t_n + \tfrac{h}{2}, u_n + \tfrac{h}{2}K_1), \\ K_3 &= f(t_n + \tfrac{h}{2}, u_n + \tfrac{h}{2}K_2), \\ K_4 &= f(t_{n+1}, u_n + hK_3). \end{aligned} \tag{11.73}$$

As far as implicit schemes are concerned, the maximum achievable order using s stages is equal to $2s$.

Remark 11.4 (The case of systems of ODEs) An RK method can be readily extended to systems of ODEs. However, the order of an RK method in the scalar case does not necessarily coincide with that in the vector case. In particular, for $p \geq 4$, a method having order p in the case of the autonomous system $\mathbf{y}' = \mathbf{f}(\mathbf{y})$, with $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^n$ maintains order p even when applied to an autonomous scalar equation $y' = f(y)$, but the converse is not true. Regarding this concern, see [Lam91], Section 5.8. ■

11.8.1 Derivation of an Explicit RK Method

The standard technique for deriving an explicit RK method consists of enforcing that the highest number of terms in Taylor's expansion of the exact solution y_{n+1} about t_n coincide with those of the approximate solution u_{n+1}, assuming that we take one step of the RK method starting from the exact solution y_n. We provide an example of this technique in the case of an explicit 2-stage RK method.

Let us consider a 2-stage explicit RK method and assume to dispose at the n-th step of the exact solution y_n. Then

$$u_{n+1} = y_n + hF(t_n, y_n, h; f) = y_n + h(b_1K_1 + b_2K_2),$$

$$K_1 = f_n, \qquad K_2 = f(t_n + hc_2, y_n + hc_2K_1),$$

having assumed that (11.72) is satisfied. Expanding K_2 in a Taylor series in a neighborhood of t_n and truncating the expansion at the second order,

we get

$$K_2 = f_n + hc_2(f_{n,t} + K_1 f_{n,y}) + \mathcal{O}(h^2).$$

We have denoted by $f_{n,z}$ (for $z = t$ or $z = y$) the partial derivative of f with respect to z evaluated at (t_n, y_n). Then

$$u_{n+1} = y_n + hf_n(b_1 + b_2) + h^2 c_2 b_2 (f_{n,t} + f_n f_{n,y}) + \mathcal{O}(h^3).$$

If we perform the same expansion on the exact solution, we find

$$y_{n+1} = y_n + hy'_n + \frac{h^2}{2} y''_n + \mathcal{O}(h^3) = y_n + hf_n + \frac{h^2}{2}(f_{n,t} + f_n f_{n,y}) + \mathcal{O}(h^3).$$

Forcing the coefficients in the two expansions above to agree, up to higher-order terms, we obtain that the coefficients of the RK method must satisfy $b_1 + b_2 = 1$, $c_2 b_2 = \frac{1}{2}$.

Thus, there are infinitely many 2-stage explicit RK methods with second-order accuracy. Two examples are the Heun method (11.10) and the modified Euler method (11.91). Of course, with similar (and cumbersome) computations in the case of higher-stage methods, and accounting for a higher number of terms in Taylor's expansion, one can generate higher-order RK methods. For instance, retaining all the terms up to the fifth one, we get scheme (11.73).

11.8.2 Stepsize Adaptivity for RK Methods

Since RK schemes are one-step methods, they are well-suited to adapting the stepsize h, provided that an efficient estimator of the local error is available. Usually, a tool of this kind is an *a posteriori* error estimator, since the *a priori* local error estimates are too complicated to be used in practice. The error estimator can be constructed in two ways:

- using the same RK method, but with two different stepsizes (typically $2h$ and h);
- using two RK methods of different order, but with the same number s of stages.

In the first case, if an RK method of order p is being used, one pretends that, starting from an exact datum $u_n = y_n$ (which would not be available if $n \geq 1$), the local error at t_{n+1} is less than a fixed tolerance. The following relation holds

$$y_{n+1} - u_{n+1} = \Phi(y_n) h^{p+1} + \mathcal{O}(h^{p+2}), \tag{11.74}$$

where Φ is an unknown function evaluated at y_n. (Notice that, in this special case, $y_{n+1} - u_{n+1} = h\tau_{n+1}(h)$).

Carrying out the same computation with a stepsize of $2h$, starting from t_{n-1}, and denoting by $\widehat{u}_{n+1}$ the computed solution, yields

$$y_{n+1} - \widehat{u}_{n+1} = \Phi(y_{n-1})(2h)^{p+1} + \mathcal{O}(h^{p+2}) = \Phi(y_n)(2h)^{p+1} + \mathcal{O}(h^{p+2}) \quad (11.75)$$

having expanded also y_{n-1} with respect to t_n. Subtracting (11.74) from (11.75), we get

$$(2^{p+1} - 1)h^{p+1}\Phi(y_n) = u_{n+1} - \widehat{u}_{n+1} + \mathcal{O}(h^{p+2}),$$

from which

$$y_{n+1} - u_{n+1} \simeq \frac{u_{n+1} - \widehat{u}_{n+1}}{(2^{p+1} - 1)} = \mathcal{E}.$$

If $|\mathcal{E}|$ is less than the fixed tolerance ε, the scheme moves to the next time step, otherwise the estimate is repeated with a halved stepsize. In general, the stepsize is doubled whenever $|\mathcal{E}|$ is less than $\varepsilon/2^{p+1}$.

This approach yields a considerable increase in the computational effort, due to the $s-1$ extra functional evaluations needed to generate the value $\widehat{u}_{n+1}$. Moreover, if one needs to half the stepsize, the value u_n must also be computed again.

An alternative that does not require extra functional evaluations consists of using simultaneously two different RK methods with s stages, of order p and $p+1$, respectively, which share the same set of values K_i. These methods are synthetically represented by the modified Butcher array

$$\begin{array}{c|l} \mathbf{c} & \mathrm{A} \\ \hline & \mathbf{b}^T \\ & \widehat{\mathbf{b}}^T \\ \hline & \mathbf{E}^T \end{array} \quad (11.76)$$

where the method of order p is identified by the coefficients $\mathbf{c}$, A and $\mathbf{b}$, while that of order $p+1$ is identified by $\mathbf{c}$, A and $\widehat{\mathbf{b}}$, and where $\mathbf{E} = \mathbf{b} - \widehat{\mathbf{b}}$.

Taking the difference between the approximate solutions at t_{n+1} produced by the two methods provides an estimate of the local truncation error for the scheme of lower order. On the other hand, since the coefficients K_i coincide, this difference is given by $h\sum_{i=1}^{s} E_i K_i$ and thus it does not require extra functional evaluations.

Notice that, if the solution u_{n+1} computed by the scheme of order p is used to initialize the scheme at time step $n+2$, the method will have order p, as a whole. If, conversely, the solution computed by the scheme of order $p+1$ is employed, the resulting scheme would still have order $p+1$ (exactly as happens with predictor-corrector methods).

The Runge-Kutta Fehlberg method of fourth-order is one of the most popular schemes of the form (11.76) and consists of a fourth-order RK

scheme coupled with a fifth-order RK method (for this reason, it is known as the RK45 method). The modified Butcher array for this method is shown below

$$
\begin{array}{c|cccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\
\frac{3}{8} & \frac{3}{32} & \frac{9}{32} & 0 & 0 & 0 & 0 \\
\frac{12}{13} & \frac{1932}{2197} & -\frac{7200}{2197} & \frac{7296}{2197} & 0 & 0 & 0 \\
1 & \frac{439}{216} & -8 & \frac{3680}{513} & -\frac{845}{4104} & 0 & 0 \\
\frac{1}{2} & -\frac{8}{27} & 2 & -\frac{3544}{2565} & \frac{1859}{4104} & -\frac{11}{40} & 0 \\
\hline
 & \frac{25}{216} & 0 & \frac{1408}{2565} & \frac{2197}{4104} & -\frac{1}{5} & 0 \\
 & \frac{16}{135} & 0 & \frac{6656}{12825} & \frac{28561}{56430} & -\frac{9}{50} & \frac{2}{55} \\
\hline
 & \frac{1}{360} & 0 & -\frac{128}{4275} & -\frac{2197}{75240} & \frac{1}{50} & \frac{2}{55}
\end{array}
$$

This method tends to underestimate the error. As such, its use is not completely reliable when the stepsize h is large.

Remark 11.5 MATLAB provides a package tool `funfun`, which, besides the two classical Runge-Kutta Fehlberg methods, RK23 (second-order and third-order pair) and RK45 (fourth-order and fifth-order pair), also implements other methods suitable for solving *stiff* problems. These methods are derived from BDF methods (see [SR97]) and are included in the MATLAB program `ode15s`. ■

11.8.3 Implicit RK Methods

Implicit RK methods can be derived from the integral formulation of the Cauchy problem (11.2). In fact, if a quadrature formula with s nodes in (t_n, t_{n+1}) is employed to approximate the integral of f (which we assume, for simplicity, to depend only on t), we get

$$\int_{t_n}^{t_{n+1}} f(\tau)\, d\tau \simeq h \sum_{j=1}^{s} b_j f(t_n + c_j h)$$

having denoted by b_j the weights and by $t_n + c_j h$ the quadrature nodes. It can be proved (see [But64]) that for any RK formula (11.70)-(11.71), there exists a correspondence between the coefficients b_j, c_j of the formula and the weights and nodes of a Gauss quadrature rule.

In particular, the coefficients $c_1, \ldots, c_s$ are the roots of the Legendre polynomial L_s in the variable $x = 2c - 1$, so that $x \in [-1, 1]$. Once the

s coefficients c_j have been found, we can construct RK methods of order $2s$, by determining the coefficients a_{ij} and b_j as being the solutions of the linear systems

$$\sum_{j=1}^{s} c_j^{k-1} a_{ij} = (1/k) c_i^k, \quad k = 1, 2, \ldots, s,$$
$$\sum_{j=1}^{s} c_j^{k-1} b_j = 1/k, \qquad k = 1, 2, \ldots, s.$$

The following families can be derived:

1. *Gauss-Legendre RK methods*, if Gauss-Legendre quadrature nodes are used. These methods, for a fixed number of stages s, attain the maximum possible order $2s$. Remarkable examples are the one-stage method (*implicit midpoint method*) of order 2

$$u_{n+1} = u_n + hf\left(t_n + \tfrac{1}{2}h, \tfrac{1}{2}(u_n + u_{n+1})\right), \quad \begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array}$$

and the 2-stage method of order 4, described by the following Butcher array

$$\begin{array}{c|cc} \frac{3-\sqrt{3}}{6} & \frac{1}{4} & \frac{3-2\sqrt{3}}{12} \\ \frac{3+\sqrt{3}}{6} & \frac{3+2\sqrt{3}}{12} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

2. *Gauss-Radau methods*, which are characterized by the fact that the quadrature nodes include one of the two endpoints of the interval (t_n, t_{n+1}). The maximum order that can be achieved by these methods is $2s-1$, when s stages are used. Elementary examples correspond to the following Butcher arrays

$$\begin{array}{c|c} 0 & 1 \\ \hline & 1 \end{array}, \qquad \begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}, \qquad \begin{array}{c|cc} \frac{1}{3} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{3}{4} & \frac{1}{4} \\ \hline & \frac{3}{4} & \frac{1}{4} \end{array}$$

and have order 1, 1 and 3, respectively. The Butcher array in the middle represents the backward Euler method.

3. *Gauss-Lobatto methods*, where both the endpoints t_n and t_{n+1} are quadrature nodes. The maximum order that can be achieved using s stages is $2s-2$. We recall the methods of the family corresponding to the following

Butcher arrays

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}, \quad \begin{array}{c|cc} 0 & \frac{1}{2} & 0 \\ 1 & \frac{1}{2} & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}, \quad \begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{6} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

which have order 2, 2 and 3, respectively. The first array represents the Crank-Nicolson method.

As for semi-implicit RK methods, we limit ourselves to mentioning the case of DIRK methods (*diagonally implicit RK*), which, for $s = 3$, are represented by the following Butcher array

$$\begin{array}{c|ccc} \frac{1+\mu}{2} & \frac{1+\mu}{2} & 0 & 0 \\ \frac{1}{2} & -\frac{\mu}{2} & \frac{1+\mu}{2} & 0 \\ \frac{1-\mu}{2} & 1+\mu & -1-2\mu & \frac{1+\mu}{2} \\ \hline & \frac{1}{6\mu^2} & 1-\frac{1}{3\mu^2} & \frac{1}{6\mu^2} \end{array}$$

The parameter μ represents one of the three roots of $3\mu^3 - 3\mu - 1 = 0$ (i.e., $(2/\sqrt{3})\cos(10^o)$, $-(2/\sqrt{3})\cos(50^o)$, $-(2/\sqrt{3})\cos(70^o)$). The maximum order that has been determined in the literature for these methods is 4.

11.8.4 Regions of Absolute Stability for RK Methods

Applying an s-stage RK method to the model problem (11.24) yields

$$K_i = u_n + h\lambda \sum_{i=1}^{s} a_{ij} K_j, \quad u_{n+1} = u_n + h\lambda \sum_{i=1}^{s} b_i K_i, \tag{11.77}$$

that is, a first-order difference equation. If $\mathbf{K}$ and $\mathbf{1}$ are the vectors of components $(K_1, \ldots, K_s)^T$ and $(1, \ldots, 1)^T$, respectively, then (11.77) becomes

$$\mathbf{K} = u_n \mathbf{1} + h\lambda \mathrm{A}\mathbf{K}, \quad u_{n+1} = u_n + h\lambda \mathbf{b}^T \mathbf{K},$$

from which, $\mathbf{K} = (\mathrm{I} - h\lambda \mathrm{A})^{-1} \mathbf{1} u_n$ and thus

$$u_{n+1} = \left[1 + h\lambda \mathbf{b}^T (\mathrm{I} - h\lambda \mathrm{A})^{-1} \mathbf{1}\right] u_n = R(h\lambda) u_n$$

where $R(h\lambda)$ is the so-called *stability function*.

The RK method is absolutely stable, i.e., the sequence $\{u_n\}$ satisfies (11.25), iff $|R(h\lambda)| < 1$. Its region of absolute stability is given by

$$\mathcal{A} = \{z = h\lambda \in \mathbb{C} \text{ such that } \ |R(h\lambda)| < 1\}.$$

If the method is explicit, A is strictly lower triangular and the function R can be written in the following form (see [DV84])

$$R(h\lambda) = \frac{\det(\mathrm{I} - h\lambda\mathrm{A} + h\lambda\mathbf{1}\mathbf{b}^T)}{\det(\mathrm{I} - h\lambda\mathrm{A})}.$$

Thus since $\det(\mathrm{I}-h\lambda\mathrm{A}) = 1$, $R(h\lambda)$ is a polynomial function in the variable $h\lambda$, $|R(h\lambda)|$ can never be less than 1 for all values of $h\lambda$. Consequently, $\mathcal{A}$ can never be unbounded for an explicit RK method.

In the special case of an explicit RK of order $s = 1, \ldots, 4$, one gets (see [Lam91])

$$R(h\lambda) = \sum_{k=0}^{s} \frac{1}{k!}(h\lambda)^k.$$

The corresponding regions of absolute stability are drawn in Figure 11.10. Notice that, unlike MS methods, the regions of absolute stability of RK methods increase in size as the order grows.

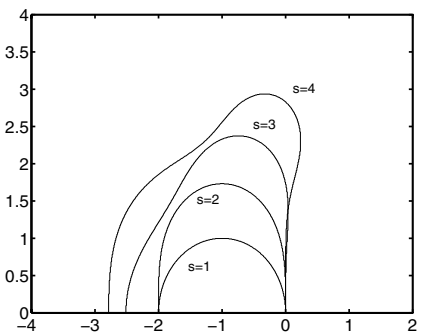

FIGURE 11.10. Regions of absolute stability for s-stage explicit RK methods, with $s = 1, \ldots, 4$. The plot only shows the portion $\mathrm{Im}(h\lambda) \geq 0$ since the regions are symmetric about the real axis

We finally notice that the regions of absolute stability for explicit RK methods can fail to be connected; an example is given in Exercise 14.

11.9 Systems of ODEs

Let us consider the system of first-order ODEs

$$\mathbf{y}' = \mathbf{F}(t, \mathbf{y}), \tag{11.78}$$

where $\mathbf{F} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ is a given vector function and $\mathbf{y} \in \mathbb{R}^n$ is the solution vector which depends on n arbitrary constants set by the n initial

conditions

$$\mathbf{y}(t_0) = \mathbf{y}_0. \tag{11.79}$$

Let us recall the following property (see [PS91], p. 209).

Property 11.5 *Let* $\mathbf{F} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ *be a continuous function on* $D = [t_0, T] \times \mathbb{R}^n$*, with* t_0 *and* T *finite. Then, if there exists a positive constant* L *such that*

$$\|\mathbf{F}(t, \mathbf{y}) - \mathbf{F}(t, \bar{\mathbf{y}})\| \leq L\|\mathbf{y} - \bar{\mathbf{y}}\| \tag{11.80}$$

holds for any $(t, \mathbf{y})$ *and* $(t, \bar{\mathbf{y}}) \in D$*, then, for any* $\mathbf{y}_0 \in \mathbb{R}^n$ *there exists a unique* $\mathbf{y}$*, continuous and differentiable with respect to* t *for any* $(t, \mathbf{y}) \in D$*, which is a solution of the Cauchy problem* (11.78)-(11.79)*.*

Condition (11.80) expresses the fact that $\mathbf{F}$ is *Lipschitz continuous* with respect to the second argument.

It is seldom possible to write out in closed form the solution to system (11.78). A special case is where the system takes the form

$$\mathbf{y}'(t) = \mathrm{A}\mathbf{y}(t), \tag{11.81}$$

with $\mathrm{A} \in \mathbb{R}^{n \times n}$. Assume that A has n distinct eigenvalues λ_j, $j = 1, \ldots, n$; therefore, the solution $\mathbf{y}$ can be written as

$$\mathbf{y}(t) = \sum_{j=1}^{n} C_j e^{\lambda_j t} \mathbf{v}_j, \tag{11.82}$$

where $C_1, \ldots, C_n$ are some constants and $\{\mathbf{v}_j\}$ is a basis formed by the eigenvectors of A, associated with the eigenvalues λ_j for $j = 1, \ldots, n$. The solution is determined by setting n initial conditions.

From the numerical standpoint, the methods introduced in the scalar case can be extended to systems. A delicate matter is how to generalize the theory developed about absolute stability.

With this aim, let us consider system (11.81). As previously seen, the property of absolute stability is concerned with the behavior of the numerical solution as t grows to infinity, in the case where the solution of problem (11.78) satisfies

$$\|\mathbf{y}(t)\| \to 0 \quad \text{as } t \to \infty. \tag{11.83}$$

Condition (11.83) is satisfied if all the real parts of the eigenvalues of A are negative since this ensures that

$$e^{\lambda_j t} = e^{\mathrm{Re}\lambda_j t}(\cos(\mathrm{Im}\lambda_j) + i \sin(\mathrm{Im}\lambda_i)) \to 0, \quad \text{as } t \to \infty, \tag{11.84}$$

from which (11.83) follows recalling (11.82). Since A has n distinct eigenvalues, there exists a nonsingular matrix Q such that $\Lambda = \mathrm{Q}^{-1}\mathrm{AQ}$, Λ being the diagonal matrix whose entries are the eigenvalues of A (see Section 1.8).

Introducing the auxiliary variable $\mathbf{z} = \mathrm{Q}^{-1}\mathbf{y}$, the initial system can therefore be transformed into

$$\mathbf{z}' = \Lambda\mathbf{z}. \tag{11.85}$$

Since Λ is a diagonal matrix, the results holding in the scalar case immediately apply to the vector case as well, provided that the analysis is repeated on all the (scalar) equations of system (11.85).

11.10 Stiff Problems

Consider a nonhomogeneous linear system of ODEs with constant coefficients

$$\mathbf{y}'(t) = \mathrm{A}\mathbf{y}(t) + \boldsymbol{\varphi}(t), \qquad \text{with } \mathrm{A} \in \mathbb{R}^{n\times n}, \quad \boldsymbol{\varphi}(t) \in \mathbb{R}^n,$$

and assume that A has n distinct eigenvalues λ_j, $j = 1, \ldots, n$. Then

$$\mathbf{y}(t) = \sum_{j=1}^{n} C_j e^{\lambda_j t}\mathbf{v}_j + \boldsymbol{\psi}(t)$$

where $C_1, \ldots, C_n$, are n constants, $\{\mathbf{v}_j\}$ is a basis formed by the eigenvectors of A and $\boldsymbol{\psi}(t)$ is a particular solution of the ODE at hand. Throughout the section, we assume that $\mathrm{Re}\lambda_j < 0$ for all j.

As $t \to \infty$, the solution $\mathbf{y}$ tends to the particular solution $\boldsymbol{\psi}$. We can therefore interpret $\boldsymbol{\psi}$ as the *steady-state solution* (that is, after an infinite time) and $\displaystyle\sum_{j=1}^{n} C_j e^{\lambda_j t}$ as being the *transient solution* (that is, for t finite). Assume that we are interested only in the steady-state. If we employ a numerical scheme with a bounded region of absolute stability, the stepsize h is subject to a constraint that depends on the maximum module eigenvalue of A. On the other hand, the greater this module, the shorter the time interval where the corresponding component in the solution is meaningful. We are thus faced with a sort of paradox: the scheme is forced to employ a small integration stepsize to track a component of the solution that is virtually flat for large values of t.

Precisely, if we assume that

$$\sigma \le \mathrm{Re}\lambda_j \le \tau < 0, \qquad \forall j = 1, \ldots, n \tag{11.86}$$

and introduce the *stiffness quotient* $r_s = \sigma/\tau$, we say that a linear system of ODEs with constant coefficients is *stiff* if the eigenvalues of matrix A all have negative real parts and $r_s \gg 1$.

However, referring only to the spectrum of A to characterize the *stiffness* of a problem might have some drawbacks. For instance, when $\tau \simeq 0$, the *stiffness* quotient can be very large while the problem appears to be "genuinely" *stiff* only if $|\sigma|$ is very large. Moreover, enforcing suitable initial conditions can affect the *stiffness* of the problem (for example, selecting the data in such a way that the constants multiplying the "*stiff*" components of the solution vanish).

For this reason, several authors find the previous definition of a *stiff* problem unsatisfactory, and, on the other hand, they agree on the fact that it is not possible to exactly state what it is meant by a *stiff* problem. We limit ourselves to quoting only one alternative definition, which is of some interest since it focuses on what is observed in practice to be a *stiff* problem.

Definition 11.14 (from [Lam91], p. 220) A system of ODEs is *stiff* if, when approximated by a numerical scheme characterized by a region of absolute stability with finite size, it forces the method, for any initial condition for which the problem admits a solution, to employ a discretization stepsize excessively small with respect to the smoothness of the exact solution. ■

From this definition, it is clear that no conditionally absolute stable method is suitable for approximating a *stiff* problem. This prompts resorting to implicit methods, such as MS or RK, which are more expensive than explicit schemes, but have regions of absolute stability of infinite size. However, it is worth recalling that, for nonlinear problems, implicit methods lead to nonlinear equations, for which it is thus crucial to select iterative numerical methods free of limitations on h for convergence.

For instance, in the case of MS methods, we have seen that using fixed-point iterations sets the constraint (11.68) on h in terms of the Lipschitz constant L of f. In the case of a linear system this constraint is

$$L \geq \max_{i=1,\dots,n} |\lambda_i|,$$

so that (11.68) would imply a strong limitation on h (which could even be more stringent than those required for an explicit scheme to be stable). One way of circumventing this drawback consists of resorting to Newton's method or some variants. The presence of Dahlquist barriers imposes a strong limitation on the use of MS methods, the only exception being BDF methods, which, as already seen, are θ-stable for $p \leq 5$ (for a larger number of steps they are even not zero-stable). The situation becomes definitely more favorable if implicit RK methods are considered, as observed at the end of Section 11.8.4.

The theory developed so far holds rigorously if the system is linear. In the nonlinear case, let us consider the Cauchy problem (11.78), where the

function $\mathbf{F} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ is assumed to be differentiable. To study its stability a possible strategy consists of linearizing the system as

$$\mathbf{y}'(t) = \mathbf{F}(\tau, \mathbf{y}(\tau)) + \mathrm{J}_{\mathbf{F}}(\tau, \mathbf{y}(\tau))\,[\mathbf{y}(t) - \mathbf{y}(\tau)]\,,$$

in a neighborhood $(\tau, \mathbf{y}(\tau))$, where τ is an arbitrarily chosen value of t within the time integration interval.

The above technique might be dangerous since the eigenvalues of $\mathrm{J}_{\mathbf{F}}$ do not suffice in general to describe the behavior of the exact solution of the original problem. Actually, some counterexamples can be found where:

1. $\mathrm{J}_{\mathbf{F}}$ has complex conjugate eigenvalues, while the solution of (11.78) does not exhibit oscillatory behavior;

2. $\mathrm{J}_{\mathbf{F}}$ has real nonnegative eigenvalues, while the solution of (11.78) does not grow monotonically with t;

3. $\mathrm{J}_{\mathbf{F}}$ has eigenvalues with negative real parts, but the solution of (11.78) does not decay monotonically with t.

   As an example of the case at item 3. let us consider the system of ODEs

$$\mathbf{y}' = \begin{bmatrix} -\dfrac{1}{2t} & \dfrac{2}{t^3} \\ -\dfrac{t}{2} & -\dfrac{1}{2t} \end{bmatrix} \mathbf{y} \quad = \quad \mathrm{A}(t)\mathbf{y}.$$

   For $t \geq 1$ its solution is

$$\mathbf{y}(t) = C_1 \begin{bmatrix} t^{-3/2} \\ -\frac{1}{2}t^{1/2} \end{bmatrix} + C_2 \begin{bmatrix} 2t^{-3/2}\log t \\ t^{1/2}(1 - \log t) \end{bmatrix}$$

   whose Euclidean norm diverges monotonically for $t > (12)^{1/4} \simeq 1.86$ when $C_1 = 1$, $C_2 = 0$, whilst the eigenvalues of $\mathrm{A}(t)$, equal to $(-1 \pm 2i)/(2t)$, have negative real parts.

Therefore, the nonlinear case must be tackled using *ad hoc* techniques, by suitably reformulating the concept of stability itself (see [Lam91], Chapter 7).

11.11 Applications

We consider two examples of dynamical systems that are well-suited to checking the performances of several numerical methods introduced in the previous sections.

11.11.1 Analysis of the Motion of a Frictionless Pendulum

Let us consider the frictionless pendulum in Figure 11.11 (left), whose motion is governed by the following system of ODEs

$$\begin{cases} y_1' = & y_2, \\ y_2' = & -K\sin(y_1), \end{cases} \tag{11.87}$$

for $t > 0$, where $y_1(t)$ and $y_2(t)$ represent the position and angular velocity of the pendulum at time t, respectively, while K is a positive constant depending on the geometrical-mechanical parameters of the pendulum. We consider the initial conditions: $y_1(0) = \theta_0$, $y_2(0) = 0$.

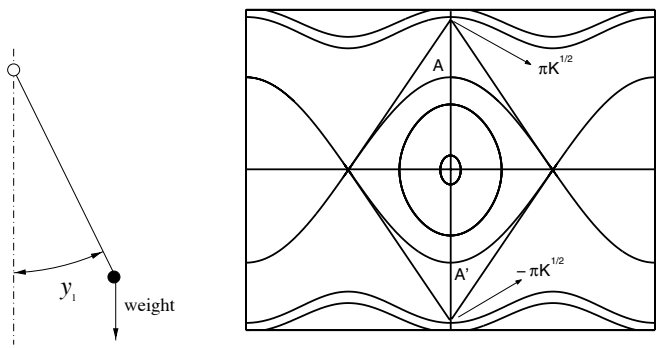

FIGURE 11.11. Left: frictionless pendulum; right: orbits of system (11.87) in the phase space

Denoting by $\mathbf{y} = (y_1, y_2)^T$ the solution to system (11.87), this admits infinitely many equilibrium conditions of the form $\mathbf{y} = (n\pi, 0)^T$ for $n \in \mathbb{Z}$, corresponding to the situations where the pendulum is vertical with zero velocity. For n even, the equilibrium is stable, while for n odd it is unstable. These conclusions can be drawn by analyzing the linearized system

$$\mathbf{y}' = \mathrm{A}_e\mathbf{y} = \begin{bmatrix} 0 & 1 \\ -K & 0 \end{bmatrix}\mathbf{y}, \qquad \mathbf{y}' = \mathrm{A}_o\mathbf{y} = \begin{bmatrix} 0 & 1 \\ K & 0 \end{bmatrix}\mathbf{y}.$$

If n is even, matrix A_e has complex conjugate eigenvalues $\lambda_{1,2} = \pm i\sqrt{K}$ and associated eigenvectors $\mathbf{y}_{1,2} = (\mp i/\sqrt{K}, 1)^T$, while for n odd, A_o has real and opposite eigenvalues $\lambda_{1,2} = \pm\sqrt{K}$ and eigenvectors $\mathbf{y}_{1,2} = (1/\sqrt{K}, \mp 1)^T$.

Let us consider two different sets of initial data: $\mathbf{y}^{(0)} = (\theta_0, 0)^T$ and $\mathbf{y}^{(0)} = (\pi + \theta_0, 0)^T$, where $|\theta_0| \ll 1$. The solutions of the corresponding linearized system are, respectively,

$$\begin{cases} y_1(t) = & \theta_0\cos(\sqrt{K}t) \\ y_2(t) = & -\sqrt{K}\theta_0\sin(\sqrt{K}t) \end{cases}, \quad \begin{cases} y_1(t) = & (\pi+\theta_0)\cosh(\sqrt{K}t) \\ y_2(t) = & \sqrt{K}(\pi+\theta_0)\sinh(\sqrt{K}t), \end{cases}$$

and will be henceforth denoted as "stable" and "unstable", respectively, for reasons that will be clear later on. To these solutions we associate in the plane (y_1, y_2), called the *phase space*, the following orbits (i.e., the graphs obtained plotting the curve $(y_1(t), y_2(t))$ in the phase space).

$$\left(\frac{y_1}{\theta_0}\right)^2 + \left(\frac{y_2}{\sqrt{K}\theta_0}\right)^2 = 1, \qquad \text{(stable case)}$$

$$\left(\frac{y_1}{\pi + \theta_0}\right)^2 - \left(\frac{y_2}{\sqrt{K}(\pi + \theta_0)}\right)^2 = 1, \quad \text{(unstable case)}.$$

In the stable case, the orbits are ellipses with period $2\pi/\sqrt{K}$ and are centered at $(0,0)^T$, while in the unstable case they are hyperbolae centered at $(0,0)^T$ and asymptotic to the straight lines $y_2 = \pm\sqrt{K}y_1$.
The complete picture of the motion of the pendulum in the phase space is shown in Figure 11.11 (right). Notice that, letting $v = |y_2|$ and fixing the initial position $y_1(0) = 0$, there exists a limit value $v_L = 2\sqrt{K}$ which corresponds in the figure to the points A and A'. For $v(0) < v_L$, the orbits are closed, while for $v(0) > v_L$ they are open, corresponding to a continuous revolution of the pendulum, with infinite passages (with periodic and non null velocity) through the two equilibrium positions $y_1 = 0$ and $y_1 = \pi$. The limit case $v(0) = v_L$ yields a solution such that, thanks to the total energy conservation principle, $y_2 = 0$ when $y_1 = \pi$. Actually, these two values are attained asymptotically only as $t \to \infty$.
The first-order nonlinear differential system (11.87) has been numerically solved using the forward Euler method (FE), the midpoint method (MP) and the Adams-Bashforth second-order scheme (AB). In Figure 11.12 we show the orbits in the phase space that have been computed by the two methods on the time interval $(0, 30)$ and taking $K = 1$ and $h = 0.1$. The crosses denote initial conditions.

As can be noticed, the orbits generated by FE do not close. This kind of instability is due to the fact that the region of absolute stability of the FE method completely excludes the imaginary axis. On the contrary, the MP method describes accurately the closed system orbits due to the fact that its region of asymptotic stability (see Section 11.6.4) includes pure imaginary eigenvalues in the neighborhood of the origin of the complex plane. It must also be noticed that the MP scheme gives rise to oscillating solutions as v_0 gets larger. The second-order AB method, instead, describes correctly all kinds of orbits.

11.11.2 Compliance of Arterial Walls

An arterial wall subject to blood flow can be modelled by a compliant circular cylinder of length L and radius R_0 with walls made by an incompressible, homogeneous, isotropic, elastic tissue of thickness H. A simple

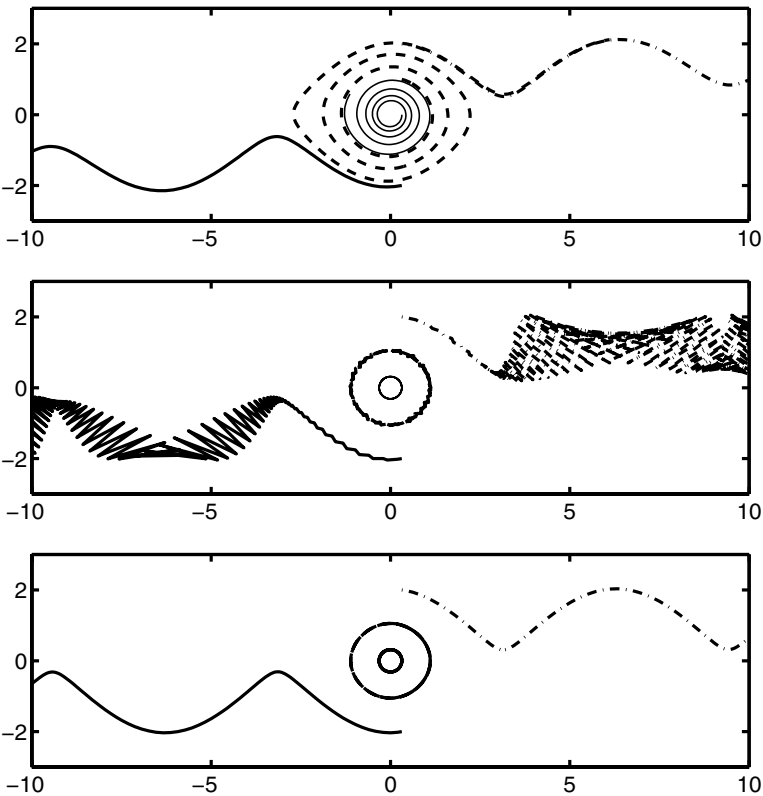

FIGURE 11.12. Orbits for system (11.87) in the case $K = 1$ and $h = 0.1$, computed using the FE method (upper plot), the MP method (central plot) and the AB method (lower plot), respectively. The initial conditions are $\theta_0 = \pi/10$ and $v_0 = 0$ (thin solid line), $v_0 = 1$ (dashed line), $v_0 = 2$ (dash-dotted line) and $v_0 = -2$ (thick solid line)

model describing the mechanical behavior of the walls interacting with the blood flow is the so called "independent-rings" model according to which the vessel wall is regarded as an assembly of rings which are not influenced one by the others.

This amounts to neglecting the longitudinal (or axial) inner actions along the vessel, and to assuming that the walls can deform only in the radial direction. Thus, the vessel radius R is given by $R(t) = R_0 + y(t)$, where y is the radial deformation of the ring with respect to a reference radius R_0 and t is the time variable. The application of Newton's law to the independent-ring system yields the following equation modeling the time mechanical

behavior of the wall

$$y''(t) + \beta y'(t) + \alpha y(t) = \gamma(p(t) - p_0) \tag{11.88}$$

where $\alpha = E/(\rho_w R_0^2)$, $\gamma = 1/(\rho_w H)$ and β is a positive constant. The physical parameters ρ_w and E denote the vascular wall density and the Young modulus of the vascular tissue, respectively. The function $p - p_0$ is the forcing term acting on the wall due to the pressure drop between the inner part of the vessel (where the blood flows) and its outer part (surrounding organs). At rest, if $p = p_0$, the vessel configuration coincides with the undeformed circular cylinder having radius equal exactly to R_0 ($y = 0$).

Equation (11.88) can be formulated as $\mathbf{y}'(t) = \mathrm{A}\mathbf{y}(t) + \mathbf{b}(t)$ where $\mathbf{y} = (y, y')^T$, $\mathbf{b} = (0, -\gamma(p - p_0))^T$ and

$$\mathrm{A} = \begin{pmatrix} 0 & 1 \\ -\alpha & -\beta \end{pmatrix}. \tag{11.89}$$

The eigenvalues of A are $\lambda_\pm = (-\beta \pm \sqrt{\beta^2 - 4\alpha})/2$; therefore, if $\beta \geq 2\sqrt{\alpha}$ both the eigenvalues are real and negative and the system is asymptotically stable with $\mathbf{y}(t)$ decaying exponentially to zero as $t \to \infty$. Conversely, if $0 < \beta < 2\sqrt{\alpha}$ the eigenvalues are complex conjugate and damped oscillations arise in the solution which again decays exponentially to zero as $t \to \infty$.
Numerical approximations have been carried out using both the backward Euler (BE) and Crank-Nicolson (CN) methods. We have set $\mathbf{y}(t) = \mathbf{0}$ and used the following (physiological) values of the physical parameters: $L = 5 \cdot 10^{-2}[m]$, $R_0 = 5 \cdot 10^{-3}[m]$, $\rho_w = 10^3[Kgm^{-3}]$, $H = 3 \cdot 10^{-4}[m]$ and $E = 9 \cdot 10^5[Nm^{-2}]$, from which $\gamma \simeq 3.3[Kg^{-1}m^{-2}]$ and $\alpha = 36 \cdot 10^6[s^{-2}]$. A sinusoidal function $p - p_0 = x\Delta p(a + b\cos(\omega_0 t))$ has been used to model the pressure variation along the vessel direction x and time, where $\Delta p = 0.25 \cdot 133.32\ [Nm^{-2}]$, $a = 10 \cdot 133.32\ [Nm^{-2}]$, $b = 133.32\ [Nm^{-2}]$ and the pulsation $\omega_0 = 2\pi/0.8\ [rad\, s^{-1}]$ corresponds to a heart beat.

The results reported below refer to the ring coordinate $x = L/2$. The two (very different) cases (1) $\beta = \sqrt{\alpha}\,[s^{-1}]$ and (2) $\beta = \alpha\,[s^{-1}]$ have been analyzed; it is easily seen that in case (2) the *stiffness quotient* (see Section 11.10) is almost equal to α, thus the problem is highly stiff. We notice also that in both cases the real parts of the eigenvalues of A are very large, so that an appropriately small time step should be taken to accurately describe the fast transient of the problem.
In case (1) the differential system has been studied on the time interval $[0, 2.5 \cdot 10^{-3}]$ with a time step $h = 10^{-4}$. We notice that the two eigenvalues of A have modules equal to 6000, thus our choice of h is compatible with the use of an explicit method as well.

Figure 11.13 (left) shows the numerical solutions as functions of time. The solid (thin) line is the exact solution while the thick dashed and solid

lines are the solutions given by the CN and BE methods, respectively. A far better accuracy of the CN method over the BE is clearly demonstrated; this is confirmed by the plot in Figure 11.13 (right) which shows the trajectories of the computed solutions in the phase space. In this case the differential system has been integrated on the time interval $[0, 0.25]$ with a time step $h = 2.5 \cdot 10^{-4}$. The dashed line is the trajectory of the CN method while the solid line is the corresponding one obtained using the BE scheme. A strong dissipation is clearly introduced by the BE method with respect to the CN scheme; the plot also shows that both methods converge to a limit cycle which corresponds to the cosine component of the forcing term.

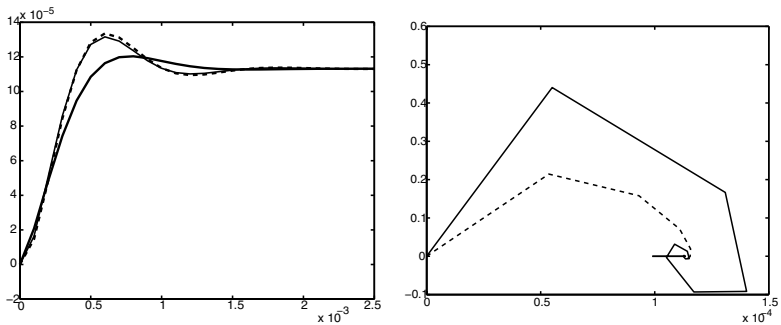

FIGURE 11.13. Transient simulation (left) and phase space trajectories (right)

In the second case (2) the differential system has been integrated on the time interval $[0, 10]$ with a time step $h = 0.1$. The stiffness of the problem is demonstrated by the plot of the deformation velocities z shown in Figure 11.14 (left). The solid line is the solution computed by the BE method while the dashed line is the corresponding one given by the CN scheme; for the sake of graphical clarity, only one third of the nodal values have been plotted for the CN method. Strong oscillations arise since the eigenvalues of matrix A are $\lambda_1 = -1$, $\lambda_2 = -36 \cdot 10^6$ so that the CN method approximates the first component y of the solution $\mathbf{y}$ as

$$y_k^{CN} = \left(\frac{1 + (h\lambda_1)/2}{1 - (h\lambda_1)/2} \right)^k \simeq (0.9048)^k, \qquad k \geq 0,$$

which is clearly stable, while the approximate second component $z(= y')$ is

$$z_k^{CN} = \left(\frac{1 + (h\lambda_2)/2}{1 - (h\lambda_2)/2} \right)^k \simeq (-0.9999)^k, \qquad k \geq 0$$

which is obviously oscillating. On the contrary, the BE method yields

$$y_k^{BE} = \left(\frac{1}{1 - h\lambda_1} \right)^k \simeq (0.9090)^k, \qquad k \geq 0,$$

and

$$z_k^{CN} = \left(\frac{1}{1-h\lambda_2}\right)^k \simeq (0.2777)^k, \qquad k \geq 0$$

which are both stable for every $h > 0$. According to these conclusions the first component y of the vector solution $\mathbf{y}$ is correctly approximated by both the methods as can be seen in Figure 11.14 (right) where the solid line refers to the BE scheme while the dashed line is the solution computed by the CN method.

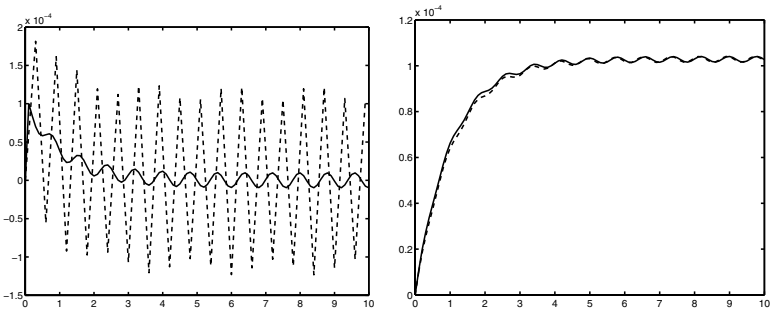

FIGURE 11.14. Long-time behavior of the solution: velocities (left) and displacements (right)

11.12 Exercises

1. Prove that Heun's method has order 2 with respect to h.
   [*Suggerimento* : notice that $h\tau_{n+1} = y_{n+1} - y_n - h\Phi(t_n, y_n; h) = E_1 + E_2$, where $E_1 = \left\{ \int_{t_n}^{t_{n+1}} f(s, y(s))ds - \frac{h}{2}[f(t_n, y_n) + f(t_{n+1}, y_{n+1})] \right\}$ and $E_2 = \frac{h}{2}\{[f(t_{n+1}, y_{n+1}) - f(t_{n+1}, y_n + hf(t_n, y_n))]\}$, where E_1 is the error due to numerical integration with the trapezoidal method and E_2 can be bounded by the error due to using the forward Euler method.]

2. Prove that the Crank-Nicoloson method has order 2 with respect to h.
   [*Solution* : using (9.12) we get, for a suitable ξ_n in (t_n, t_{n+1})

$$y_{n+1} = y_n + \frac{h}{2}\left[f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right] - \frac{h^3}{12} f''(\xi_n, y(\xi_n))$$

   or, equivalently,

$$\frac{y_{n+1} - y_n}{h} = \frac{1}{2}\left[f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right] - \frac{h^2}{12} f''(\xi_n, y(\xi_n)). \qquad (11.90)$$

   Therefore, relation (11.9) coincides with (11.90) up to an infinitesimal of order 2 with respect to h, provided that $f \in C^2(I)$.]

3. Solve the difference equation $u_{n+4} - 6u_{n+3} + 14u_{n+2} - 16u_{n+1} + 8u_n = n$ subject to the initial conditions $u_0 = 1$, $u_1 = 2$, $u_2 = 3$ and $u_3 = 4$.
   [*Solution* : $u_n = 2^n(n/4 - 1) + 2^{(n-2)/2}\sin(\pi/4) + n + 2$.]

4. Prove that if the characteristic polynomial Π defined in (11.30) has simple roots, then any solution of the associated difference equation can be written in the form (11.32).
   [*Hint* : notice that a generic solution u_{n+k} is completely determined by the initial values $u_0, \ldots, u_{k-1}$. Moreover, if the roots r_i of Π are distinct, there exist unique k coefficients α_i such that $\alpha_1 r_1^j + \ldots + \alpha_k r_k^j = u_j$ with $j = 0, \ldots, k-1 \ldots$]

5. Prove that if the characteristic polynomial Π has simple roots, the matrix R defined in (11.37) is not singular.
   [*Hint:* it coincides with the transpose of the Vandermonde matrix where x_i^j is replaced by r_j^i (see Exercise 2, Chapter 8).]

6. The Legendre polynomials L_i satisfy the difference equation
$$(n+1)L_{n+1}(x) - (2n+1)xL_n(x) + nL_{n-1}(x) = 0$$
   with $L_0(x) = 1$ and $L_1(x) = x$ (see Section 10.1.2). Defining the generating function $F(z,x) = \sum_{n=0}^{\infty} P_n(x)z^n$, prove that $F(z,x) = (1 - 2zx + z^2)^{-1/2}$.

7. Prove that the *gamma function*
$$\Gamma(z) = \int_0^\infty e^{-t}t^{z-1}dt, \qquad z \in \mathbb{C}, \quad \mathrm{Re}z > 0$$
   is the solution of the difference equation $\Gamma(z+1) = z\Gamma(z)$
   [*Hint* : integrate by parts.]

8. Study, as functions of $\alpha \in \mathbb{R}$, stability and order of the family of linear multistep methods
$$u_{n+1} = \alpha u_n + (1-\alpha)u_{n-1} + 2hf_n + \frac{h\alpha}{2}\left[f_{n-1} - 3f_n\right].$$

9. Consider the following family of linear multistep methods depending on the real parameter α
$$u_{n+1} = u_n + h[(1 - \frac{\alpha}{2})f(x_n, u_n) + \frac{\alpha}{2}f(x_{n+1}, u_{n+1})].$$
   Study their consistency as a function of α; then, take $\alpha = 1$ and use the corresponding method to solve the Cauchy problem
$$\begin{aligned} &y'(x) = -10y(x), \quad x > 0, \\ &y(0) = 1. \end{aligned}$$
   Determine the values of h in correspondance of which the method is absolutely stable.
   [*Solution* : the only consistent method of the family is the Crank-Nicolson method ($\alpha = 1$).]

10. Consider the family of linear multistep methods

$$u_{n+1} = \alpha u_n + \frac{h}{2}\left(2(1-\alpha) f_{n+1} + 3\alpha f_n - \alpha f_{n-1}\right)$$

where α is a real parameter.

(a) Analyze consistency and order of the methods as functions of α, determining the value α^* for which the resulting method has maximal order.

(b) Study the zero-stability of the method with $\alpha = \alpha^*$, write its characteristic polynomial $\Pi(r; h\lambda)$ and, using MATLAB, draw its region of absolute stability in the complex plane.

11. Adams methods can be easily generalized, integrating between t_{n-r} and t_{n+1} with $r \geq 1$. Show that, by doing so, we get methods of the form

$$u_{n+1} = u_{n-r} + h \sum_{j=-1}^{p} b_j f_{n-j}$$

and prove that for $r = 1$ the midpoint method introduced in (11.43) is recovered (the methods of this family are called *Nystron methods*.)

12. Check that Heun's method (11.10) is an explicit two-stage RK method and write the Butcher arrays of the method. Then, do the same for the *modified Euler method*, given by

$$u_{n+1} = u_n + hf(t_n + \frac{h}{2}, u_n + \frac{h}{2} f_n), \quad n \geq 0. \tag{11.91}$$

[*Solution* : the methods have the following Butcher arrays

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \qquad \begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array} .]$$

13. Check that the Butcher array for method (11.73) is given by

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

14. Write a MATLAB program to draw the regions of absolute stability for a RK method, for which the function $R(h\lambda)$ is available. Check the code in the special case of

$$R(h\lambda) = 1 + h\lambda + (h\lambda)^2/2 + (h\lambda)^3/6 + (h\lambda)^4/24 + (h\lambda)^5/120 + (h\lambda)^6/600$$

and verify that such a region is not connected.

[Axe94] Axelsson O. (1994) *Iterative Solution Methods.* Cambridge University Press, New York.

[Bar89] Barnett S. (1989) Leverrier's Algorithm: A New Proof and Extensions. *Numer. Math.* 7: 338–352.

[Bat90] Batterson S. (1990) Convergence of the Shifted QR Algorithm on 3 by 3 Normal Matrices. *Numer. Math.* 58: 341–352.

[BBC+94] Barrett R., Berry M., Chan T., Demmel J., Donato J., Dongarra J., Eijkhout V., Pozo V., Romine C., and van der Vorst H. (1994) *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, Philadelphia.

[BD74] Björck A. and Dahlquist G. (1974) *Numerical Methods.* Prentice-Hall, Englewood Cliffs, N.J.

[BDMS79] Bunch J., Dongarra J., Moler C., and Stewart G. (1979) *LINPACK User's Guide.* SIAM, Philadelphia.

[Ber82] Bertsekas D. P. (1982) *Constrained Optimization and Lagrange Multiplier Methods.* Academic Press. Inc., San Diego, California.

[Bjö88] Björck A. (1988) *Least Squares Methods: Handbook of Numerical Analysis Vol. 1 Solution of Equations in* $\mathbb{R}^N$. Elsevier North Holland.

[BM92] Bernardi C. and Maday Y. (1992) *Approximations Spectrales des Problémes aux Limites Elliptiques.* Springer-Verlag, Paris.

[BMW67] Barth W., Martin R. S., and Wilkinson J. H. (1967) Calculation of the Eigenvalues of a Symmetric Tridiagonal Matrix by the Method of Bisection. *Numer. Math.* 9: 386–393.

[BO78] Bender C. M. and Orszag S. A. (1978) *Advanced Mathematical Methods for Scientists and Engineers.* McGraw-Hill, New York.

[Boe80] Boehm W. (1980) Inserting New Knots into B-spline Curves. *Computer Aided Design* 12: 199–201.

[Bos93] Bossavit A. (1993) *Electromagnetisme, en vue de la modelisation.* Springer-Verlag, Paris.

[BR81] Bank R. E. and Rose D. J. (1981) Global Approximate Newton Methods. *Numer. Math.* 37: 279–295.

[Bra75] Bradley G. (1975) *A Primer of Linear Algebra.* Prentice-Hall, Englewood Cliffs, New York.

[Bre73] Brent R. (1973) *Algorithms for Minimization Without Derivatives*. Prentice-Hall, Englewood Cliffs, New York.

[Bri74] Brigham E. O. (1974) *The Fast Fourier Transform*. Prentice-Hall, Englewood Cliffs, New York.

[BS90] Brown P. and Saad Y. (1990) Hybrid Krylov Methods for Nonlinear Systems of equations. *SIAM J. Sci. and Stat. Comput.* 11(3): 450–481.

[BSG96] B. Smith P. B. and Gropp P. (1996) *Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Univ. Cambridge Press, Cambridge.

[But64] Butcher J. C. (1964) Implicit Runge-Kutta Processes. *Math. Comp.* 18: 233–244.

[But66] Butcher J. C. (1966) On the Convergence of Numerical Solutions to Ordinary Differential Equations. *Math. Comp.* 20: 1–10.

[But87] Butcher J. (1987) *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*. Wiley, Chichester.

[CCP70] Cannon M., Cullum C., and Polak E. (1970) *Theory and Optimal Control and Mathematical Programming*. McGraw-Hill, New York.

[CFL28] Courant R., Friedrichs K., and Lewy H. (1928) Über die partiellen differenzengleichungen der mathematischen physik. *Math. Ann.* 100: 32–74.

[CHQZ88] Canuto C., Hussaini M. Y., Quarteroni A., and Zang T. A. (1988) *Spectral Methods in Fluid Dynamics*. Springer, New York.

[CI95] Chandrasekaren S. and Ipsen I. (1995) On the Sensitivity of Solution Components in Linear Systems of equations. *SIAM J. Matrix Anal. Appl.* 16: 93–112.

[CL91] Ciarlet P. G. and Lions J. L. (1991) *Handbook of Numerical Analysis: Finite Element Methods (Part 1)*. North-Holland, Amsterdam.

[CM94] Chan T. and Mathew T. (1994) Domain Decomposition Algorithms. *Acta Numerica* pages 61–143.

[CMSW79] Cline A., Moler C., Stewart G., and Wilkinson J. (1979) An Estimate for the Condition Number of a Matrix. *SIAM J. Sci. and Stat. Comput.* 16: 368–375.

[Col66] Collin R. E. (1966) *Foundations for Microwave Engineering.* McGraw-Hill Book Co., Singapore.

[Com95] Comincioli V. (1995) *Analisi Numerica Metodi Modelli Applicazioni.* McGraw-Hill Libri Italia, Milano.

[Cox72] Cox M. (1972) The Numerical Evaluation of B-splines. *Journal of the Inst. of Mathematics and its Applications* 10: 134–149.

[Cry73] Cryer C. W. (1973) On the Instability of High Order Backward-Difference Multistep Methods. *BIT* 13: 153–159.

[CT65] Cooley J. and Tukey J. (1965) An Algorithm for the Machine Calculation of Complex Fourier Series. *Math. Comp.* 19: 297–301.

[Dah56] Dahlquist G. (1956) Convergence and Stability in the Numerical Integration of Ordinary Differential Equations. *Math. Scand.* 4: 33–53.

[Dah63] Dahlquist G. (1963) A Special Stability Problem for Linear Multistep Methods. *BIT* 3: 27–43.

[Dat95] Datta B. (1995) *Numerical Linear Algebra and Applications.* Brooks/Cole Publishing, Pacific Grove, CA.

[Dau88] Daubechies I. (1988) Orthonormal bases of compactly supported wavelets. *Commun. on Pure and Appl. Math.* XLI.

[Dav63] Davis P. (1963) *Interpolation and Approximation.* Blaisdell Pub., New York.

[Day96] Day D. (1996) How the QR algorithm Fails to Converge and How to Fix It. Technical Report 96-0913J, Sandia National Laboratory, Albuquerque.

[dB72] de Boor C. (1972) On Calculating with B-splines. *Journal of Approximation Theory* 6: 50–62.

[dB83] de Boor C. (1983) A Practical Guide to Splines. In *Applied Mathematical Sciences.* (27), Springer-Verlag, New York.

[dB90] de Boor C. (1990) *SPLINE TOOLBOX for use with MATLAB.* The Math Works, Inc., South Natick.

[DD95] Davis T. and Duff I. (1995) A combined unifrontal/multifrontal method for unsymmetric sparse matrices. Technical Report TR-95-020, Computer and Information Sciences Department, University of Florida.

[Dek69] Dekker T. (1969) Finding a Zero by means of Successive Linear Interpolation. In Dejon B. and Henrici P. (eds) *Constructive Aspects of the Fundamental Theorem of Algebra*, pages 37–51. Wiley, New York.

[Dek71] Dekker T. (1971) A Floating-Point Technique for Extending the Available Precision. *Numer. Math.* 18: 224–242.

[Dem97] Demmel J. (1997) *Applied Numerical Linear Algebra*. SIAM, Philadelphia.

[DGK84] Dongarra J., Gustavson F., and Karp A. (1984) Implementing Linear Algebra Algorithms for Dense Matrices on a Vector Pipeline Machine. *SIAM Review* 26(1): 91–112.

[Die87a] Dierckx P. (1987) *FITPACK User Guide part 1: Curve Fitting Routines.* TW Report, Dept. of Computer Science, Katholieke Universiteit, Leuven, Belgium.

[Die87b] Dierckx P. (1987) *FITPACK User Guide part 2: Surface Fitting Routines.* TW Report, Dept. of Computer Science, Katholieke Universiteit, Leuven, Belgium.

[Die93] Dierckx P. (1993) *Curve and Surface Fitting with Splines*. Claredon Press, New York.

[DL92] DeVore R. and Lucier J. (1992) Wavelets. *Acta Numerica* pages 1–56.

[DR75] Davis P. and Rabinowitz P. (1975) *Methods of Numerical Integration*. Academic Press, New York.

[DS83] Dennis J. and Schnabel R. (1983) *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, New York.

[Dun85] Dunavant D. (1985) High Degree Efficient Symmetrical Gaussian Quadrature Rules for the Triangle. *Internat. J. Numer. Meth. Engrg.* 21: 1129–1148.

[Dun86] Dunavant D. (1986) Efficient Symmetrical Cubature Rules for Complete Polynomials of High Degree over the Unit Cube. *Internat. J. Numer. Meth. Engrg.* 23: 397–407.

[DV84] Dekker K. and Verwer J. (1984) *Stability of Runge-Kutta Methods for Stiff Nonlinear Differential Equations.* North-Holland, Amsterdam.

[dV89] der Vorst H. V. (1989) High Performance Preconditioning. *SIAM J. Sci. Stat. Comput.* 10: 1174–1185.

[EEHJ96] Eriksson K., Estep D., Hansbo P., and Johnson C. (1996) *Computational Differential Equations.* Cambridge Univ. Press, Cambridge.

[Elm86] Elman H. (1986) A Stability Analisys of Incomplete LU Factorization. *Math. Comp.* 47: 191–218.

[Erd61] Erdös P. (1961) Problems and Results on the Theory of Interpolation. *Acta Math. Acad. Sci. Hungar.* 44: 235–244.

[Erh97] Erhel J. (1997) About Newton-Krylov Methods. In Periaux J. and al. (eds) *Computational Science for* 21^{st} *Century*, pages 53–61. Wiley, New York.

[Fab14] Faber G. (1914) Über die interpolatorische Darstellung stetiger Funktionem. *Jber. Deutsch. Math. Verein.* 23: 192–210.

[FF63] Faddeev D. K. and Faddeeva V. N. (1963) *Computational Methods of Linear Algebra.* Freeman, San Francisco and London.

[Fle75] Fletcher R. (1975) Conjugate gradient methods for indefinite systems. In Springer-Verlag (ed) *Numerical Analysis*, pages 73–89. New York.

[FM67] Forsythe G. E. and Moler C. B. (1967) *Computer Solution of Linear Algebraic Systems.* Prentice-Hall, Englewood Cliffs, New York.

[Fra61] Francis J. G. F. (1961) The QR Transformation: A Unitary Analogue to the LR Transformation. Parts I and II. *Comp. J.* pages 265–272,332–334.

[FRL55] F. Richtmyer E. K. and Lauritsen T. (1955) *Introduction to Modern Physics.* McGraw-Hill, New York.

[Gas83] Gastinel N. (1983) *Linear Numerical Analysis.* Kershaw Publishing, London.

[Gau94] Gautschi W. (1994) Algorithm 726: ORTHPOL - A Package of Routines for Generating Orthogonal Polynomials and Gauss-type Quadrature Rules. *ACM Trans. Math. Software* 20: 21–62.

[Gau96] Gautschi W. (1996) Orthogonal Polynomials: Applications and Computation. *Acta Numerica* pages 45–119.

[Gau97] Gautschi W. (1997) *Numerical Analysis. An Introduction.* Birkhäuser, Berlin.

[Geo73] George A. (1973) Nested Dissection of a Regular Finite Element Mesh. *SIAM J. Num. Anal.* 10: 345–363.

[Giv54] Givens W. (1954) Numerical Computation of the Characteristic Values of a Real Symmetric Matrix. *Oak Ridge National Laboratory* ORNL-1574.

[GL81] George A. and Liu J. (1981) *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, Englewood Cliffs, New York.

[GL89] Golub G. and Loan C. V. (1989) *Matrix Computations.* The John Hopkins Univ. Press, Baltimore and London.

[GM83] Golub G. and Meurant G. (1983) *Resolution Numerique des Grands Systemes Lineaires.* Eyrolles, Paris.

[GMW81] Gill P., Murray W., and Wright M. (1981) *Practical Optimization.* Academic Press, London.

[God66] Godeman R. (1966) *Algebra.* Kershaw, London.

[Gol91] Goldberg D. (1991) What Every Computer Scientist Should Know about Floating-point Arithmetic. *ACM Computing Surveys* 23(1): 5–48.

[GP67] Goldstein A. A. and Price J. B. (1967) An Effective Algorithm for Minimization. *Numer. Math* 10: 184–189.

[GR96] Godlewski E. and Raviart P. (1996) *Numerical Approximation of Hyperbolic System of Conservation Laws*, volume 118 of *Applied Mathematical Sciences.* Springer-Verlag, New York.

[Hac94] Hackbush W. (1994) *Iterative Solution of Large Sparse Systems of Equations.* Springer-Verlag, New York.

[Hah67] Hahn W. (1967) *Stability of Motion.* Springer-Verlag, Berlin.

[Hal58] Halmos P. (1958) *Finite-Dimensional Vector Spaces.* Van Nostrand, Princeton, New York.

[Hen62] Henrici P. (1962) *Discrete Variable Methods in Ordinary Differential Equations.* Wiley, New York.

[Hen74] Henrici P. (1974) *Applied and Computational Complex Analysis*, volume 1. Wiley, New York.

[HGR96] H-G. Roos M. Stynes L. T. (1996) *Numerical Methods for Singularly Perturbed Differential Equations.* Springer-Verlag, Berlin Heidelberg.

[Hig88] Higham N. (1988) The Accuracy of Solutions to Triangular Systems. *University of Manchester, Dep. of Mathematics* 158: 91–112.

[Hig89] Higham N. (1989) The Accuracy of Solutions to Triangular Systems. *SIAM J. Numer. Anal.* 26(5): 1252–1265.

[Hig96] Higham N. (1996) *Accuracy and Stability of Numerical Algorithms.* SIAM Publications, Philadelphia, PA.

[Hil87] Hildebrand F. (1987) *Introduction to Numerical Analysis.* McGraw-Hill, New York.

[Hou75] Householder A. (1975) *The Theory of Matrices in Numerical Analysis.* Dover Publications, New York.

[HP94] Hennessy J. and Patterson D. (1994) *Computer Organization and Design - The Hardware/Software Interface.* Morgan Kaufmann, San Mateo.

[HW76] Hammarling S. and Wilkinson J. (1976) The Practical Behaviour of Linear Iterative Methods with Particular Reference to S.O.R. Technical Report Report NAC 69, National Physical Laboratory, Teddington, UK.

[IK66] Isaacson E. and Keller H. (1966) *Analysis of Numerical Methods.* Wiley, New York.

[Inm94] Inman D. (1994) *Engineering Vibration.* Prentice-Hall, Englewood Cliffs, NJ.

[Iro70] Irons B. (1970) A Frontal Solution Program for Finite Element Analysis. *Int. J. for Numer. Meth. in Engng.* 2: 5–32.

[Jac26] Jacobi C. (1826) Uber Gauβ neue Methode, die Werthe der Integrale näherungsweise zu finden. *J. Reine Angew. Math.* 30: 127–156.

[Jer96] Jerome J. J. (1996) *Analysis of Charge Transport. A Mathematical Study of Semiconductor Devices.* Springer, Berlin Heidelberg.

[Jia95] Jia Z. (1995) The Convergence of Generalized Lanczos Methods for Large Unsymmetric Eigenproblems. *SIAM J. Matrix Anal. Applic.* 16: 543–562.

[JM92] Jennings A. and McKeown J. (1992) *Matrix Computation.* Wiley, Chichester.

[Joh90] Johnson C. (1990) *Numerical Solution of Partial Differential Equations by the Finite Element Method.* Cambridge Univ. Press.

[JW77] Jankowski M. and Wozniakowski M. (1977) Iterative Refinement Implies Numerical Stability. *BIT* 17: 303–311.

[Kah66] Kahan W. (1966) Numerical Linear Algebra. *Canadian Math. Bull.* 9: 757–801.

[Kan66] Kaniel S. (1966) Estimates for Some Computational Techniques in Linear Algebra. *Math. Comp.* 20: 369–378.

[Kea86] Keast P. (1986) Moderate-Degree Tetrahedral Quadrature Formulas. *Comp. Meth. Appl. Mech. Engrg.* 55: 339–348.

[Kel99] Kelley C. (1999) *Iterative Methods for Optimization*, volume 18 of *Frontiers in Applied Mathematics.* SIAM, Philadelphia.

[KT51] Kuhn H. and Tucker A. (1951) Nonlinear Programming. In *Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492. Univ. of California Press, Berkeley and Los Angeles.

[Lam91] Lambert J. (1991) *Numerical Methods for Ordinary Differential Systems.* John Wiley and Sons, Chichester.

[Lan50] Lanczos C. (1950) An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operator. *J. Res. Nat. Bur. Stand.* 45: 255–282.

[Lax65] Lax P. (1965) Numerical Solution of Partial Differential Equations. *Amer. Math. Monthly* 72(2): 74–84.

[Lel92] Lele S. (1992) Compact Finite Difference Schemes with Spectral-like Resolution. *Journ. of Comp. Physics* 103(1): 16–42.

[Lem89] Lemarechal C. (1989) Nondifferentiable Optimization. In Nemhauser G., Kan A. R., and Todd M. (eds) *Handbooks Oper. Res. Management Sci.*, volume 1. Optimization, pages 529–572. North-Holland, Amsterdam.

[LH74] Lawson C. and Hanson R. (1974) *Solving Least Squares Problems*. Prentice-Hall, Englewood Cliffs, New York.

[LM68] Lions J. L. and Magenes E. (1968) *Problemes aux limitès nonhomogènes et applications*. Dunod, Paris.

[LS96] Lehoucq R. and Sorensen D. (1996) Deflation Techniques for an Implicitly Restarted Iteration. *SIAM J. Matrix Anal. Applic.* 17(4): 789–821.

[Lue73] Luenberger D. (1973) *Introduction to Linear and Non Linear Programming*. Addison-Wesley, Reading, Massachusetts.

[Man69] Mangasarian O. (1969) *Non Linear Programming*. Prentice-Hall, Englewood Cliffs, New Jersey.

[Man80] Manteuffel T. (1980) An Incomplete Factorization Technique for Positive Definite Linear Systems. *Math. Comp.* 150(34): 473–497.

[Mar86] Markowich P. (1986) *The Stationary Semiconductor Device Equations*. Springer-Verlag, Wien and New York.

[McK62] McKeeman W. (1962) Crout with Equilibration and Iteration. *Comm. ACM* 5: 553–555.

[MdV77] Meijerink J. and der Vorst H. V. (1977) An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric M-matrix. *Math. Comp.* 137(31): 148–162.

[MM71] Maxfield J. and Maxfield M. (1971) *Abstract Algebra and Solution by Radicals*. Saunders, Philadelphia.

[MMG87] Martinet R., Morlet J., and Grossmann A. (1987) Analysis of sound patterns through wavelet transforms. *Int. J. of Pattern Recogn. and Artificial Intellig.* 1(2): 273–302.

[MNS74] Mäkela M., Nevanlinna O., and Sipilä A. (1974) On the Concept of Convergence, Consistency and Stability in Connection with Some Numerical Methods. *Numer. Math.* 22: 261–274.

[Mor84] Morozov V. (1984) *Methods for Solving Incorrectly Posed Problems*. Springer-Verlag, New York.

[Mul56] Muller D. (1956) A Method for Solving Algebraic Equations using an Automatic Computer. *Math. Tables Aids Comput.* 10: 208–215.

[NAG95] NAG (1995) *NAG Fortran Library Manual - Mark 17.* NAG Ltd., Oxford.

[Nat65] Natanson I. (1965) *Constructive Function Theory*, volume III. Ungar, New York.

[NM65] Nelder J. and Mead R. (1965) A simplex method for function minimization. *The Computer Journal* 7: 308–313.

[Nob69] Noble B. (1969) *Applied Linear Algebra.* Prentice-Hall, Englewood Cliffs, New York.

[OR70] Ortega J. and Rheinboldt W. (1970) *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, New York and London.

[Pap62] Papoulis A. (1962) *The Fourier Integral and its Application.* McGraw-Hill, New York.

[Pap87] Papoulis A. (1987) *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, New York.

[Par80] Parlett B. (1980) *The Symmetric Eigenvalue Problem.* Prentice-Hall, Englewood Cliffs, NJ.

[PdKÜK83] Piessens R., deDoncker Kapenga E., Überhuber C. W., and Kahaner D. K. (1983) *QUADPACK: A Subroutine Package for Automatic Integration.* Springer-Verlag, Berlin and Heidelberg.

[PJ55] Peaceman D. and Jr. H. R. (1955) The numerical solution of parabolic and elliptic differential equations. *J. Soc. Ind. Appl. Math.* 3: 28–41.

[Pou96] Poularikas A. (1996) *The Transforms and Applications Handbook.* CRC Press, Inc., Boca Raton, Florida.

[PR70] Parlett B. and Reid J. (1970) On the Solution of a System of Linear Equations Whose Matrix is Symmetric but not Definite. *BIT* 10: 386–397.

[PS91] Pagani C. and Salsa S. (1991) *Analisi Matematica*, volume II. Masson, Milano.

[PW79] Peters G. and Wilkinson J. (1979) Inverse iteration, ill-conditioned equations, and newton's method. *SIAM Review* 21: 339–360.

[QV94] Quarteroni A. and Valli A. (1994) *Numerical Approximation of Partial Differential Equations*. Springer, Berlin and Heidelberg.

[QV99] Quarteroni A. and Valli A. (1999) *Domain Decomposition Methods for Partial Differential Equations*. Oxford Science Publications, New York.

[Ral65] Ralston A. (1965) *A First Course in Numerical Analysis*. McGraw-Hill, New York.

[Red86] Reddy B. D. (1986) *Applied Functional Analysis and Variational Methods in Engineering*. McGraw-Hill, New York.

[Ric81] Rice J. (1981) *Matrix Computations and Mathematical Software*. McGraw-Hill, New York.

[Riv74] Rivlin T. (1974) *The Chebyshev Polynomials*. John Wiley and Sons, New York.

[RM67] Richtmyer R. and Morton K. (1967) *Difference Methods for Initial Value Problems*. Wiley, New York.

[RR78] Ralston A. and Rabinowitz P. (1978) *A First Course in Numerical Analysis*. McGraw-Hill, New York.

[Rud83] Rudin W. (1983) *Real and Complex Analysis*. Tata McGraw-Hill, New Delhi.

[Rut58] Rutishauser H. (1958) Solution of Eigenvalue Problems with the LR Transformation. *Nat. Bur. Stand. Appl. Math. Ser.* 49: 47–81.

[Saa90] Saad Y. (1990) Sparskit: A basic tool kit for sparse matrix computations. Technical Report 90-20, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffet Field, CA.

[Saa92] Saad Y. (1992) *Numerical Methods for Large Eigenvalue Problems*. Halstead Press, New York.

[Saa96] Saad Y. (1996) *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, Boston.

[Sch67] Schoenberg I. (1967) On Spline functions. In Shisha O. (ed) *Inequalities*, pages 255–291. Academic Press, New York.

[Sch81] Schumaker L. (1981) *Splines Functions: Basic Theory*. Wiley, New York.

[Sel84] Selberherr S. (1984) *Analysis and Simulation of Semiconductor Devices*. Springer-Verlag, Wien and New York.

[SG69] Scharfetter D. and Gummel H. (1969) Large-signal analysis of a silicon Read diode oscillator. *IEEE Trans. on Electr. Dev.* 16: 64–77.

[Ske79] Skeel R. (1979) Scaling for Numerical Stability in Gaussian Elimination. *J. Assoc. Comput. Mach.* 26: 494–526.

[Ske80] Skeel R. (1980) Iterative Refinement Implies Numerical Stability for Gaussian Elimination. *Math. Comp.* 35: 817–832.

[SL89] Su B. and Liu D. (1989) *Computational Geometry: Curve and Surface Modeling*. Academic Press, New York.

[Sla63] Slater J. (1963) *Introduction to Chemical Physics*. McGraw-Hill Book Co.

[Smi85] Smith G. (1985) *Numerical Solution of Partial Differential Equations: Finite Difference Methods*. Oxford University Press, Oxford.

[Son89] Sonneveld P. (1989) Cgs, a fast lanczos-type solver for nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* 10(1): 36–52.

[SR97] Shampine L. F. and Reichelt M. W. (1997) The MATLAB ODE Suite. *SIAM J. Sci. Comput.* 18: 1–22.

[SS90] Stewart G. and Sun J. (1990) *Matrix Perturbation Theory*. Academic Press, New York.

[SS98] Schwab C. and Schötzau D. (1998) Mixed hp-FEM on Anisotropic Meshes. *Mat. Models Methods Appl. Sci.* 8(5): 787–820.

[Ste71] Stetter H. (1971) Stability of discretization on infinite intervals. In Morris J. (ed) *Conf. on Applications of Numerical Analysis*, pages 207–222. Springer-Verlag, Berlin.

[Ste73] Stewart G. (1973) *Introduction to Matrix Computations*. Academic Press, New York.

[Str69] Strassen V. (1969) Gaussian Elimination is Not Optimal. *Numer. Math.* 13: 727–764.

[Str80] Strang G. (1980) *Linear Algebra and Its Applications.* Academic Press, New York.

[Str89] Strikwerda J. (1989) *Finite Difference Schemes and Partial Differential Equations.* Wadsworth and Brooks/Cole, Pacific Grove.

[Sze67] Szegö G. (1967) *Orthogonal Polynomials.* AMS, Providence, R.I.

[Tit37] Titchmarsh E. (1937) *Introduction to the Theory of Fourier Integrals.* Oxford.

[Var62] Varga R. (1962) *Matrix Iterative Analysis.* Prentice-Hall, Englewood Cliffs, New York.

[vdV92] van der Vorst H. (1992) Bi-cgstab: a fast and smoothly converging variant of bi-cg for the solution of non-symmetric linear systems. *SIAM Jour. on Sci. and Stat. Comp.* 12: 631–644.

[Ver96] Verfürth R. (1996) *A Review of a Posteriori Error Estimation and Adaptive Mesh Refinement Techniques.* Wiley, Teubner, Germany.

[Wac66] Wachspress E. (1966) *Iterative Solutions of Elliptic Systems.* Prentice-Hall, Englewood Cliffs, New York.

[Wal75] Walsh G. (1975) *Methods of Optimization.* Wiley.

[Wal91] Walker J. (1991) *Fast Fourier Transforms.* CRC Press, Boca Raton.

[Wen66] Wendroff B. (1966) *Theoretical Numerical Analysis.* Academic Press, New York.

[Wid67] Widlund O. (1967) A Note on Unconditionally Stable Linear Multistep Methods. *BIT* 7: 65–70.

[Wil62] Wilkinson J. (1962) Note on the Quadratic Convergence of the Cyclic Jacobi Process. *Numer. Math.* 6: 296–300.

[Wil63] Wilkinson J. (1963) *Rounding Errors in Algebraic Processes.* Prentice-Hall, Englewood Cliffs, New York.

[Wil65] Wilkinson J. (1965) *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford.

[Wil68] Wilkinson J. (1968) A priori Error Analysis of Algebraic Processes. In *Intern. Congress Math.*, volume 19, pages 629–639. Izdat. Mir, Moscow.

[Wol69] Wolfe P. (1969) Convergence Conditions for Ascent Methods. *SIAM Review* 11: 226–235.

[Wol71] Wolfe P. (1971) Convergence Conditions for Ascent Methods. II: Some Corrections. *SIAM Review* 13: 185–188.

[Wol78] Wolfe M. (1978) *Numerical Methods for Unconstrained Optimization.* Van Nostrand Reinhold Company, New York.

[You71] Young D. (1971) *Iterative Solution of Large Linear Systems.* Academic Press, New York.

[Zie77] Zienkiewicz O. C. (1977) *The Finite Element Method (Third Edition).* McGraw Hill, London.