Chapter 2
Polynomials and Series

Abstract This chapter introduces the reader to the numerical aspects of polynomials. In particular, we examine different polynomial bases such as the monomial, the Chebyshev, and the Lagrange basis; we provide *algorithms to evaluate polynomials* in many of those bases and examine the *different condition numbers in different bases*. We give a first look at the important problem of numerically finding *zeros* and *pseudozeros* of polynomials. We give an algorithmic overview of the *numerical computation of truncated power series* including Taylor series. Finally, we give a brief discussion of *asymptotics*. ◁

Computation with polynomials is one of the pillars on which numerical analysis stands. This book makes extensive use of polynomials, as do all numerical analysis texts, but it takes advantage of several recent theoretical and practical advances in this foundational discipline. It is perhaps somewhat surprising that there were advances to be made in so venerable and well-studied an area, but there were, and almost certainly there still are. This chapter introduces our notations, reviews the basic ideas of the theory and practice of univariate polynomial computation, and gives several facts and algorithms. Some of these algorithms and theorems may be surprising even to people who have some numerical analysis background, and so we recommend that everyone at least skim this chapter, for notation if nothing else.

The related topic of series algebra is also one of the pillars of numerical analysis; indeed, numerical analysis has often been dubbed nothing but "a huge collection of applications of Taylor's theorem." We believe that it isn't quite true (even when the assertion is modified to include "—and, of course, linear algebra"). More properly, the theory of Taylor series provides an interesting and common way of generating polynomial approximations to functions. While Taylor series are of more than just marginal value in this book, they aren't central; but they are useful, and so a section on how to compute them (which will most likely differ from the way the reader was taught to compute them, in their first-year calculus class!) is included.

2.1 Polynomials, Their Bases, and Their Roots

Let us begin with a definition of the main object of this chapter.

Definition 2.1 (Polynomial). A polynomial is a function $f:\mathbb{C}\to\mathbb{C}$ such that, for some nonnegative integer n and for some $a_k\in\mathbb{C}$, $0\le k\le n$, with $a_n\neq 0$,

$$f(z)=\sum_{k=0}^{n}a_k z^k \tag{2.1}$$

for all $z\in\mathbb{C}$. The functions $1,z,z^2,\ldots,z^n$ are called *monomials*, and the a_k are called the coefficients of the monomials for $f(z)$. By convention, the identically zero function $f(z)\equiv 0$ is also called a polynomial, and in this case alone there is no n with $a_n\neq 0$. The *degree* of $f(z)$, written $\deg f$ or $\deg_z f$, is the number n of Eq. (2.1). Moreover, by convention, the degree of the identically zero polynomial is $-\infty$. $\lhd$

The set of all polynomials of degree at most n forms a finite-dimensional vector space. As we can see from their definitions, polynomials are linear combinations of $1,z,z^2,z^3,\ldots,z^m$, for $m\le n$. Moreover, the following fact is obtained:

Theorem 2.1. *If a polynomial $p(z)$ is identically zero, that is, if*

$$a_0+a_1z+a_2z^2+\cdots+a_nz^n\equiv 0\,,$$

then $a_k=0$ for all k such that $0\le k\le n$.

The proof is left as Exercise 2.1. As a result, the functions $1,z,z^2,z^3,\ldots,z^n$ are linearly independent in $\mathbb{C}$. Also, the functions $1,z,z^2,z^3,\ldots,z^n$ span the vector space of polynomials of degrees at most n. Consequently, the monomials form an $(n+1)$-dimensional basis. This basis is known as the *monomial basis*.

There are many other possible bases that can be used to represent spaces of polynomials and, as we will see, what basis we use has important consequences in numerical contexts. The most common bases will be discussed in Sect. 2.2. We can define bases generally as follows.

Definition 2.2 (Basis). A *basis* for the space of polynomials of degree at most n is a set of polynomials $\{\phi_k(z)\}_{k=0}^{n}$ that may be written as

$$\begin{bmatrix}\phi_0(z)\\ \phi_1(z)\\ \vdots\\ \phi_n(z)\end{bmatrix}=\mathbf{B}\begin{bmatrix}1\\ z\\ \vdots\\ z^n\end{bmatrix} \tag{2.2}$$

for some *nonsingular* $(n+1)\times(n+1)$ matrix $\mathbf{B}$. In this case, $\boldsymbol{\phi}(z)$ will denote the vector $[\phi_0(z),\ldots,\phi_n(z)]^T$ and $\mathbf{z^k}$ will denote the vector $[1,z,\ldots,z^n]^T$, and we will simply write

$$\boldsymbol{\phi}(z) = \mathbf{B}\mathbf{z}^{\mathbf{k}}\,. \tag{2.3}$$

When the degree of each polynomial in the basis is such that $\deg \phi_k(z) = k$, we say that the basis is *degree-graded*. ◁

Moreover, polynomial bases have the properties we expect from bases, most notably uniqueness of representation.

Theorem 2.2. *The coefficients of $f(z)$ in the basis $\{\phi_k(z)\}_{k=0}^{n}$ are unique. That is, if*

$$f(z) = \sum_{k=0}^{n} c_k \phi_k(z) \qquad \text{and} \qquad f(z) = \sum_{k=0}^{n} b_k \phi_k(z) \tag{2.4}$$

for all $z \in \mathbb{C}$, then $c_k = b_k$ for $0 \le k \le n$.

The proof is left as Exercise 2.2.

The role of polynomials in scientific computation is such that we often want to find their roots. Because of that, we now turn to some important facts about roots of polynomials that will be used in what follows.

Definition 2.3 (Root, or Zero). A complex number r is called a *root* (or *zero*) of $f(z)$ if $f(r) = 0$. The *multiplicity* of r is the least number m such that $f^{(m)}(r) \neq 0$. It is guaranteed that $m \le n$ unless $f(z) \equiv 0$. A root is called *simple* if $m = 1$. ◁

One of the most important properties of polynomials is revealed by this theorem, first proved by Gauss in 1797:

Theorem 2.3 (Fundamental theorem of algebra). *If $f(z)$ is a polynomial not equal to a nonzero constant, that is, if $\deg f \neq 0$ (remember that $\deg f = -\infty$ if $f = 0$ identically), then f has a root.*

As Wilkinson (1984) notices,

> [t]he Fundamental Theorem of Algebra asserts that every polynomial equation over the complex field has a root. It is almost beneath such a majestic theorem to mention that in fact it has precisely n roots.

Remark 2.1. The problem of *finding* all roots of a polynomial, and in particular finding multiple roots when the data are ambiguous, is quite difficult[1]; we shall discuss this material later. A good place for the impatient to start some extra reading is Zeng (2004). ◁

We end this subsection with two important theorems that will be used later:

[1] With some definitions of "finding," it is impossible for generic polynomials $p(z)$ of degree 5 or more. Degree-5 polynomials can be solved using elliptic functions, though, and there are other tricks. Here, by "finding," we mean finding a good approximation.

Theorem 2.4 (Factor theorem). *If $f(z)$ has ℓ distinct roots r_k, $1 \le k \le \ell$, each with multiplicity m_k (so $n = \sum_{k=1}^{\ell} m_k$), then*

$$f(z) = a_n \prod_{k=1}^{\ell} (z - r_k)^{m_k} \,. \tag{2.5}$$

Theorem 2.5 (Continuity). *(Ostrowski 1940, 1973) The roots of a polynomial are continuous functions of the coefficients a_k (in any fixed basis). Simple roots are continuously differentiable functions of the coefficients.*

2.1.1 Change of Polynomial Bases

One sometimes wants to change a representation of a polynomial p from one basis to another. In other words, given two bases $\{\phi_k(z)\}_{k=0}^{n}$ and $\{\psi_k(z)\}_{k=0}^{n}$, what is the relation between the coefficients a_k and b_k in the expression

$$p(z) = \sum_{k=0}^{n} a_k \phi_k(z) = \sum_{k=0}^{n} b_k \psi_k(z)\,?$$

In theory, the answer straightforwardly follows from the definition of a basis: If we are given a basis $\{\phi_k(z)\}_{k=0}^{n}$, then it can be expressed as the product of a nonsingular matrix $\mathbf{B}$ and the vector of monomials $\mathbf{z^k}$. The same is true of another basis $\{\psi_k(z)\}_{k=0}^{n}$. Thus, if we let $\boldsymbol{\phi}(z) = \mathbf{B}_1\mathbf{z^k}$ and $\boldsymbol{\psi}(z) = \mathbf{B}_2\mathbf{z^k}$, the relation between the bases is given by

$$\boldsymbol{\phi}(z) = \mathbf{B}_1\mathbf{B}_2^{-1}\boldsymbol{\psi}(z)\,. \tag{2.6}$$

If we let $\boldsymbol{\Phi} = \mathbf{B}_1\mathbf{B}_2^{-1}$ denote the *change-of-basis matrix*,[2] we see that change of basis is the following simple linear transformation:

$$\begin{bmatrix} \phi_0(z) \\ \phi_1(z) \\ \vdots \\ \phi_n(z) \end{bmatrix} = \begin{bmatrix} \phi_{00} & \phi_{01} & \cdots & \phi_{0n} \\ \phi_{10} & \phi_{11} & \cdots & \phi_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{n0} & \phi_{n1} & \cdots & \phi_{nn} \end{bmatrix} \begin{bmatrix} \psi_0(z) \\ \psi_1(z) \\ \vdots \\ \psi_n(z) \end{bmatrix} . \tag{2.7}$$

When the basis is degree-graded, the change-of-basis matrix is triangular.

Finally, observe that the relation between the coefficients of the polynomial bases is as follows. Since

[2] Note that, depending on the author and conventions being used, $\boldsymbol{\Phi}$ or its transpose may refer to the change-of-basis matrix.

$$p(z) = \begin{bmatrix} b_0 \, b_1 \, \cdots \, b_n \end{bmatrix} \begin{bmatrix} \psi_0(z) \\ \psi_1(z) \\ \vdots \\ \psi_n(z) \end{bmatrix} = \begin{bmatrix} a_0 \, a_1 \, \cdots \, a_n \end{bmatrix} \begin{bmatrix} \phi_0(z) \\ \phi_1(z) \\ \vdots \\ \phi_n(z) \end{bmatrix}$$

$$= \begin{bmatrix} a_0 \, a_1 \, \cdots \, a_n \end{bmatrix} \begin{bmatrix} \phi_{00} & \phi_{01} & \cdots & \phi_{0n} \\ \phi_{10} & \phi_{11} & \cdots & \phi_{1n} \\ \vdots & \vdots & \ddots & \\ \phi_{n0} & \phi_{n1} & \cdots & \phi_{nn} \end{bmatrix} \begin{bmatrix} \psi_0(z) \\ \psi_1(z) \\ \vdots \\ \psi_n(z) \end{bmatrix}, \tag{2.8}$$

the relation between the coefficients of $p(z)$ in the bases $\{\phi_k(z)\}_{k=0}^{n}$ and $\{\psi_k(z)\}_{k=0}^{n}$ is given by

$$\begin{bmatrix} b_0 \, b_1 \, \cdots \, b_n \end{bmatrix} = \begin{bmatrix} a_0 \, a_1 \, \cdots \, a_n \end{bmatrix} \boldsymbol{\Phi}\,. \tag{2.9}$$

Thus, the same matrix $\boldsymbol{\Phi}$ relates the bases vectors $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ and their coefficients.

Remark 2.2. Changing the expression of a polynomial from one basis to another is a *mathematically* valid operation, but we remark right now that it is not always (or even often) a good thing to do *numerically*. This is why Wilkinson (1959a) claims that if

> the explicit polynomial [in monomial basis] has been derived by expanding some other expression, then we may well question the wisdom of this step.

As we will see in Sect. 8.6, changing polynomial bases can amplify numerical errors dramatically: even in the normwise sense, error bounds can grow exponentially with the degree of the polynomial, and componentwise the relative errors can be infinitely larger in one basis than in another. Changing the basis must be done with caution, if at all. $\triangleleft$

2.1.2 Operations on Polynomials

The following operations can be performed in any polynomial basis. To begin with, the sum of two polynomials (say f and g, of degrees n and m) is a polynomial (just add the coefficients), the negation of a polynomial is a polynomial (just negate the coefficients), and the product of two polynomials is again a polynomial (in this case, the coefficients of the product are bilinear functions of the coefficients of the multiplicands, and the particular function depends on the basis, as we will see).

Polynomial division is a bit more complicated, but not that much. If $f(z) = Q(z)g(z) + R(z)$ and $\deg R < \deg g$, we say that $R(z)$ is the *remainder* on division of $f(z)$ by $g(z)$; if $R(z)$ is identically zero, then we say that $g(z)$ *divides* (or divides evenly into) $f(z)$. This cannot happen if $g(z)$ is identically zero. If g does divide f, then we write $g \mid f$ (which is read as "g divides f"). The polynomial $Q(z)$ in $f = Qg + R$ is called the *quotient*. It is easy to prove that, given $f(z)$ and $g(z)$, the quotient and remainder are unique.[3] Polynomial division is merely mentioned in this book, but is occasionally needed in applications. Again the details of the division process depend on the basis being used, but note that it amounts to solving a linear system of equations for the unknown coefficients of $Q(z)$ and $R(z)$, once the bilinear functions of multiplication in that basis are known.

We also occasionally need the notion of *relatively prime polynomials*, and for that we need the notion of greatest common divisor, or GCD. A polynomial $d(z)$ is a common divisor of f and g if both $d \mid f$ and $d \mid g$. If d has the maximum possible degree of all common divisors of f and g, we say that it is a GCD of f and g. Every constant multiple of a common divisor is a common divisor, and so GCDs are unique only up to multiplication by a constant.

The *composition* $f(g(z))$ is also a polynomial, of degree nm. It is sometimes worthwhile to seek to rewrite a large polynomial $F(z)$ as a composition $F(z) = f(g(z))$; finding such f and g is called polynomial decomposition. We will not pursue this further in this book, but it also finds use in some applications.

2.2 Examples of Polynomial Bases

Several polynomial bases are commonly encountered in applications. We have already encountered the monomial basis, and we will soon see why it should sometimes be avoided in applications. Before that, we examine some of the most common bases that arise, and indicate some of their advantages and disadvantages.

2.2.1 Shifted Monomials

Shifted monomials (shifted by a constant $a \in \mathbb{C}$) are polynomials having the form

$$\phi_k(z) = (z-a)^k\,, \tag{2.10}$$

and the set $\{(z-a)^k\}_{k=0}^n$ forms a basis. The expansion of a polynomial $f(z)$ in this basis is just its Taylor series:

$$f(z) = f(a) + f'(a)(z-a) + \cdots + \frac{f^{(n)}(a)}{n!}(z-a)^n\,. \tag{2.11}$$

If $a = 0$, this is just the standard monomial basis, also called the power basis. The change-of-basis matrix from the monomials to the shifted monomials is simple. For $n = 3$, this is

[3] This is more generally true than we need here: the coefficients of our polynomials are complex numbers or real numbers and this statement is true for more general objects as well.

$$\begin{bmatrix}\phi_0(z)\\ \phi_1(z)\\ \phi_2(z)\\ \phi_3(z)\end{bmatrix} = \begin{bmatrix}1\\ z-a\\ (z-a)^2\\ (z-a)^3\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0\\ -a & 1 & 0 & 0\\ a^2 & -2a & 1 & 0\\ -a^3 & 3a^2 & -3a & 1\end{bmatrix}\begin{bmatrix}1\\ z\\ z^2\\ z^3\end{bmatrix}. \tag{2.12}$$

The change-of-basis matrix that goes from the shifted monomials to the monomials is just the inverse of this matrix (and that it exists and is nonsingular for any a means that the shifted monomials are indeed a basis).

Remark 2.3. Multiplication of polynomials expressed in the monomial basis is a familiar operation. Multiplication of two polynomials expressed in an arbitrary (but common to the two polynomials) shifted monomial basis may be done by embedding them in Taylor series and using the methods of Sect. 2.6. This can also be done rapidly by use of the fast Fourier transform (FFT) (see Chap. 9). ◁

As we have seen in Chap. 1, it is important to compute sums in a stable and predictable way when we use computer arithmetic. For polynomials expressed in the shifted monomial basis, we can use *Horner's method*, which can be written as

$$f(z) = f(a) + (z-a)\left(f'(a) + (z-a)\left(\frac{f''(a)}{2} + (z-a)\left(\cdots + (z-a)\left(\frac{f^{(n)}(a)}{n!}\right)\cdots\right)\right)\right). \tag{2.13}$$

The key difference with the use of Eq. (2.11) is that we associate terms in a way that does not require us to compute higher powers of $z-a$. In addition, as in this formula, it is generally preferable to include the factorials in the Taylor coefficients. For a polynomial of degree n, this formula requires $O(n)$ flops, where explicitly forming each terms $(z-a)^k$ requires more.

Assuming that the coefficients of f in this basis are stored in a vector c indexed from 1 to $n+1$, so that c(1) $= f(a)$, c(2) $= f'(a)$, c(3) $= f^{(2)}(a)/2!$, and so on, one can use a simple MATLAB program to carry out the computation of $f(z)$ based on Horner's method:

```
p = c(n+1)*ones(size(z));
za = z - a;
for i=n:-1:1,
    p = za.*p + c(i);
end;
```

Note that, in this code, the coefficient of the power-n term is the last component of the vector of coefficients, as opposed to other commands such as MATLAB's built-in command polyval, where the order is reversed. Because polyval can be simply adapted to use a shifted monomial basis, we show how to use it in an example.

Example 2.1. A monomial basis polynomial is entered as a vector of coefficients (in decreasing order of exponent, and zero coefficients must be explicitly included).

Consider the polynomial $p(z) = z^4 - 4z^3 + 3z^2 - 2z + 5$ on, say, $0 \le z \le 2$.[4] Thus, we simply need to execute

```
p = [ 1, -4, 3, -2, 5 ];
z = linspace( 0, 2, 101 );
pz = polyval( p, z );
plot( z, pz, 'k' )
```

This code generates a graph in which we see, by eye, a zero of $p(z)$ near $z = 1.5$. ◁

Instead of just evaluating a polynomial, one can change a polynomial from the monomial basis to a shifted monomial basis (that is, a Taylor series) by using an extension of Horner's method called *synthetic division*. This method, which is widely discussed in the literature, is described by Algorithm 2.1. We will use this algorithm occasionally, and so we will discuss its accuracy later, in Sect. 2.2.1.2.

Algorithm 2.1 Synthetic division of a polynomial $f(z) = \sum_{j=0}^{n} c_j (z-a)^j$ expressed in a shifted monomial basis, evaluating $f(z)$ and its first k derivatives at $z = b$, returning $f_k = f^{(k)}(b)/k!$

```
Require: The expansion point $a \in \mathbb{C}$, a vector of monomial coefficients $\mathbf{c} \in \mathbb{C}^{n+1}$ (indexed from
  0 to $n$) such that $f(z) = \sum_{j=0}^{n} c_j (z-a)^j$, a new expansion point $b$ and a desired number $k \ge 0$ of
  Taylor coefficients of $f(z)$ at $z = b$.
  $f_0 := c_n$
  $f_{(1:k)} := 0$
  for $j=n-1:-1:0$ do
    for $i=\min(k, n-j):-1:1$ do
      $f_i = (b-a) f_i + f_{i-1}$
    end for
    $f_0 = (b-a) f_0 + f_j$
  end for
  return The $(k+1)$-vector $\mathbf{f}$ such that $f(z) = \sum_{j=0}^{k} f_j (z-b)^j + O(z-b)^{k+1}$. That is, $f_j = f^{(j)}(b)/j!$.
```

Example 2.2. Consider Example 2.1 again, and let us expand this polynomial about $z = 1.5$, where we saw our approximate zero. We used MAPLE and its `series` command to effect Algorithm 2.1, and thus found that

$$p(z) = 0.3125 - 6.5(z-1.5) - 1.5(z-1.5)^2 + 2(z-1.5)^3 + (z-1.5)^4 . \quad (2.14)$$

This is a new expression for the polynomial, this time expanded about $z = 1.5$. ◁

2.2.1.1 Newton's Method for Polynomials

As an aside, we briefly introduce Newton's method for finding zeros of polynomials. This will be taken up in greater detail and generality in the next chapter. Newton

[4] This example is drawn from Henrici (1964). In Exercise 2.5, you will be asked to consider instead Newton's example, $p(z) = z^3 - 2z - 5$.

suggested that we could use the first two coefficients of the shifted polynomial as a linear approximation to the polynomial and could be used in an attempt to find a root: In this example, setting the linear approximation $0.3125 - 6.5\,(z-1.5) = 0$ yields $z - 1.5 = {}^{0.3125}\!/_{6.5} \approx 0.048076923$, suggesting that we should shift our expansion point again, this time to $z = 1.5 + 0.048076923 \approx 1.548$. When we do so, we find

$$p(z) = -0.002730 - 6.630\,(z-1.548) - 1.198\,(z-1.548)^2 \\ + 2.192\,(z-1.548)^3 + 1.0\,(z-1.548)^4$$

and since now $p(1.548)$ is smaller than before, we begin to see that the process might work in an iterative fashion.

In general, suppose that we have expanded the polynomial about $z = r_k$, where r_k is our current approximation of the root:

$$p(z) = p(r_k) + p'(r_k)(z - r_k) + O(z - r_k)^2\,. \tag{2.15}$$

Using Newton's idea, we solve this linear approximation (which is possible if $p'(r_k) \neq 0$) to find

$$z \doteq r_k - \frac{p(r_k)}{p'(r_k)}, \tag{2.16}$$

and it makes sense to name this approximation r_{k+1}:

$$r_{k+1} = r_k - \frac{p(r_k)}{p'(r_k)}\,. \tag{2.17}$$

This is, of course, Newton's method, which we will take up further in Chap. 3. For now, note two things: First, each approximation r_k is the *exact* root of

$$p(z) - p(r_k) = 0\,, \tag{2.18}$$

and so if $p(r_k)$ (which we call the *residual*) is small, then we have found the exact solution of a nearby polynomial, and, second, we have found that this process is apt to fail near multiple roots because if both $p(z^*)$ and $p'(z^*) = 0$, then since $r_k \to z^*$, both $p(r_k) \to 0$ and $p'(r_k) \to 0$, making the solving step problematic.

Continuing our example[5] just one more iteration, with $r_k = 1.548$ in Eq. (2.15), we have $r_{k+1} = 1.548 - \frac{(-0.0027295)}{(-6.62973)}$, that is, $r_{k+1} \approx 1.5475883$. Shifting to the basis centered here using synthetic division, we have

$$p(z) = -0.00000024948774 - 6.6287459395492\,(z - 1.5475883) - \cdots, \tag{2.19}$$

[5] If you try to reproduce these computations, your results may differ because we were somewhat cavalier in rounding intermediate results. Keeping all figures—as one should—will make the results slightly different.

and we notice that 1.5475883 is an exact root of the nearby polynomial $p(z) + 2.494877\ldots \times 10^{-7}$.

2.2.1.2 Errors in Synthetic Division

We refer to Higham (2002) for a complete accuracy analysis of synthetic division, but we state a result here connecting rounding errors and the forward error via a condition number. Let $B^{(j)}(z)$ be defined as

$$B^{(j)}(z) := \sum_{k=j}^{n} k^{\underline{j}} \left| \frac{f^{(k)}(a)}{k!} \right| \left| (z-a)^{k-j} \right| , \tag{2.20}$$

where $k^{\underline{j}}$, read as "k to the j falling," is defined as

$$k^{\underline{j}} = \frac{k!}{(k-j)!} = k(k-1)(k-2)\cdots(k-j+1)$$

(see Graham et al. 1994). Then the difference between the reference value of the derivative $f^{(j)}(\alpha)$ and the value computed by synthetic division, say $\hat{r}_j$, is bounded by

$$\left| f^{(j)}(\alpha) - \hat{r}_j \right| \leq O(n\mu_M) B^{(j)}(\alpha) + O(\mu_M^2) \, . \tag{2.21}$$

We will see later in this chapter many more examples of such $B(z)$ functions, which are called *condition numbers for evaluation of polynomials*. In some sense, the above theorem, which (to first order) bounds the *forward error* $|f^{(j)} - \hat{r}_j|$ by the product of a condition number and a backward error (here $O(n\mu_M)$), is as important to numerical analysis as $F = ma$ is to physics.

Changing from bases other than the monomial basis to shifted monomials is sometimes useful (again, numerically this has to be done with caution, as we will see). We pursue this in the exercises.

2.2.2 The Newton Basis

We have seen that the shifted monomial basis is defined in reference to a given data point a. Similarly, the Newton basis is defined in reference to a set of points, which we call *nodes*. The Newton basis on the $n+1$ nodes τ_0, τ_1, τ_2, $\ldots$, τ_n is given by

$$\{\phi_k(z)\}_{k=0}^{n} = \{1, z-\tau_0, (z-\tau_0)(z-\tau_1), \ldots, (z-\tau_0)(z-\tau_{‘})\cdots(z-\tau_{n-1})\} \, ,$$

or, more compactly,

$$\{\phi_k(z)\}_{k=0}^{n} = \left\{ \prod_{i=0}^{k-1} (z - \tau_i) \right\}_{k=0}^{n} . \tag{2.22}$$

Note that, by convention, if $m > n$, a product $\prod_{i=m}^{n}$ is just 1. Note also and especially that one node, namely, τ_n, is omitted from any mention in this basis. We remark that this permits choice: One speaks of “a” Newton basis, not of “the” Newton basis. There is a further choice involved, namely, the ordering of the nodes; once one of the $n+1$ nodes has been omitted, there is a further $n!$ different orderings possible if the nodes are distinct. Some of them are numerically better than others, as we will see.

Newton bases are typically used with what are called *divided differences* (see Problem 8.13). In fact, de Boor (2005) defines divided differences as the coefficients of $f(z)$ expressed in a Newton basis. Though divided differences and Newton bases have a rich theory and practice, they will only rarely be used in this book because there are better choices available. Trefethen (2013 p. 33) takes a similar stance:

Many textbooks claim that it is important to use this approach for reasons of numerical stability, but this is not true, and we shall not discuss the Newton approach here.

They are the preferred basis in de Boor (1978), because they are convenient, inexpensive, and, for low degrees, accurate. However, as we will see, the barycentric Lagrange basis that we prefer is much better conditioned for larger degrees on good sets of nodes. After introducing the Lagrange basis, we will return to this point.

2.2.3 Chebyshev Polynomials

The Chebyshev polynomials can be defined by

$$\phi_k(z) = T_k(z) = \cos(k \cos^{-1} z) \tag{2.23}$$

for $k = 0, 1, 2, \ldots$. It is easy to see that, for $k = 0$ and $k = 1$, these are indeed polynomials:

$$T_0(z) = \cos 0 = 1 \tag{2.24}$$

$$T_1(z) = \cos\cos^{-1} z = z . \tag{2.25}$$

Moreover, by applying the angle sum and angle difference formulæ for cosines to $\cos((k+1)\cos^{-1} z)$ and $\cos((k-1)\cos^{-1} z)$, it follows that, for $k > 1$,

$$T_{k+1}(z) = 2zT_k(z) - T_{k-1}(z) . \tag{2.26}$$

Hence, all $\phi_k(z)$ are polynomials. Figure 2.1 displays the first nine Chebyshev polynomials.

A well-known algorithm to compute the values of polynomials expressed in this basis is provided in Rivlin (1990 156–158). It turns out that this algorithm is called

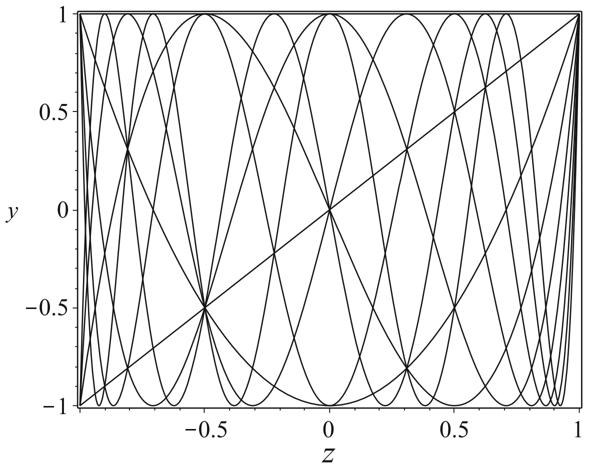

Fig. 2.1 The first nine Chebyshev polynomials $T_0(z) = 1$, $T_1(z) = z$, and $T_{n+1}(z) = 2zT_n(z) - T_{n-1}(z)$. See Exercise 2.33

the Clenshaw algorithm, which we take up after mentioning other polynomials belonging to an important class to which Chebyshev polynomials belong, namely, *orthogonal polynomials*. In the real case, Chebyshev polynomials can be shown to be orthogonal with respect to the inner product

$$\langle f, g\rangle := \int_{-1}^{1} \frac{f(x)g(x)}{\sqrt{1-x^2}}\,dx\,. \tag{2.27}$$

In the complex case, they are also orthogonal. The zeros of $T_n(z)$ are

$$\xi_k^{(n)} := \cos\left(\frac{\pi(k-1/2)}{n}\right) \tag{2.28}$$

for $k = 1, 2, \ldots, n$. The proof is left as Exercise 2.4. Chebyshev polynomials are *also* orthogonal with respect to the following *discrete* inner product:

$$\langle f, g\rangle := \sum_{j=1}^{n} f(\xi_j^{(n)})g(\xi_j^{(n)})\,. \tag{2.29}$$

See Rivlin (1990 Exercise 1.5.26, p. 53) for a complete enumeration of all cases of $\langle T_k, T_p\rangle$. You will be asked to prove in Exercise 2.6 that this discrete orthogonality relation allows easy computation of the coefficients of the expansion of a degree-$(n-1)$ polynomial $p(z)$ if you can evaluate it on the zeros of $T_n(z)$:

$$p(z) = \frac{A_0}{2}T_0(z) + A_1T_1(z) + \cdots + A_{n-1}T_{n-1}(z), \tag{2.30}$$

where[6]

$$A_m = \frac{2}{n} \sum_{j=1}^{n} p(\xi_j^{(n)}) T_m(\xi_j^{(n)}) \,, \tag{2.31}$$

for $m = 0, 1, \ldots, n-1$, can be computed with $O(n^2)$ floating-point operations.

Another and in some sense more interesting set of discrete points is called the *Chebyshev–Lobatto* points or *Chebyshev extreme* points, which are the places where $T_n(z)$ achieves its maximum and minimum values on $-1 \le z \le 1$. The endpoints are special, and are always included, because $T_n(1) = 1$ and $T_n(-1) = (-1)^n$, as you may easily prove by the definition $T_n(z) = \cos(n \cos^{-1} z)$. The interior (relative) extrema are the zeros of $T_n'(z)$ and there can be at most $n-1$ of them. Since $\cos(n\theta) = \pm 1$ at these extrema, we can verify that

$$\eta_k = \eta_k^{(n)} = \cos\frac{k\pi}{n} \tag{2.32}$$

for $k = 0, 1, \ldots, n$. This gives $n+1$ extrema on the interval, including the endpoints with $k = 0$ and $k = n$, and thus it must include all possible extrema. Note that both ξ_k and η_k run "backward" across the interval, which is sometimes inconvenient but only trivially so.

Chebyshev polynomials have a large collection of interesting and useful properties, some of which will be discussed when they come up naturally in the book. Chebyshev polynomials are the favorite of many numerical analysts. In particular, the Chebfun package is founded on the properties of Chebyshev polynomials (it uses the η_k, not the ξ_k). We will see several examples of its use in this book. Chebfun uses the syntax `chebpoly(n)` to pick out a Chebyshev polynomial. See Exercise 2.7.

2.2.4 *Other Orthogonal Polynomials*

There are a great many other examples of orthogonal polynomials. The orthogonal polynomials implemented in MAPLE include the Chebyshev polynomials, where the name `ChebyshevT` is used, with the syntax `ChebyshevT(n,z)`. Other orthogonal polynomials implemented include the Gegenbauer polynomials (`GegenbauerC`), the Hermite orthogonal polynomials[7] (`HermiteH`), and the Jacobi polynomials (`JacobiP(n,a,b,z)`). The latter include as a special case (`JacobiP(n,0,0,z)`), more usually called the *Legendre* polynomials; these will be used in Chap. 10 for Gaussian quadrature. Maple has another package for the ma-

[6] Note that we use $A_0/2$ in equation (2.30) so that formula (2.31) can be the same for $m = 0, 1, 2, \ldots$

[7] They will almost never be used in this book and are not to be confused with the Hermite interpolational basis polynomials, which will be used.

nipulation of orthogonal series, namely, the `OrthogonalSeries` package, which is quite extensive.

A common characteristic of orthogonal polynomials is that they generally satisfy a three-term recurrence relation for $n \geq 2$, which we write here as

$$\alpha_{n-1}\phi_n(z) = (z - \beta_{n-1})\phi_{n-1}(z) - \gamma_{n-1}\phi_{n-2}(z) \,. \tag{2.33}$$

As we saw above, the recurrence for the Chebyshev polynomials has $\alpha_{n-1} = \gamma_{n-1} = {}^1\!/_2$ and $\beta_{n-1} = 0$ for all n. However, for other classes of polynomials, there is a dependence on n. For instance, the recurrence relation for the Jacobi polynomials starts with $P_0(z) = 1$, $P_1(z) = {}^{(a-b)}\!/_2 + (1 + {}^{(a+b)}\!/_2)z$, and thereafter

$$\begin{aligned}
\alpha_{n-1} &= \frac{2n\,(n+a+b)}{(2n+a+b-1)(2n+a+b)} \\
\beta_{n-1} &= \frac{(b-a)(a+b)}{(2n+a+b-2)(2n+a+b)} \\
\gamma_{n-1} &= \frac{2\,(n+a-1)(n+b-1)}{(2n+a+b-1)(2n+a+b-2)} \,.
\end{aligned} \tag{2.34}$$

In the special case $a = b = 0$, for the Legendre polynomials, we have $P_0(z) = 1$, $P_1(z) = z$, and

$$\alpha_{n-1} = \frac{n}{2n-1} \qquad \beta_{n-1} = 0 \qquad \text{and} \qquad \gamma_{n-1} = \frac{n-1}{2n-1} \,. \tag{2.35}$$

The first 10 Legendre polynomials are plotted in Fig. 10.4 of Chap. 10.

Recent uses in mathematical handwriting recognition of generalizations of orthogonal polynomials—namely, the Legendre–Sobolev polynomials, which include derivatives in their inner product—can be seen in Golubitsky and Watt (2009) and in Golubitsky and Watt (2010). See Exercise 4.28.

2.2.5 *The Clenshaw Algorithm for Evaluating Polynomials Expressed in Orthogonal Bases*

The Clenshaw algorithm generalizes the idea used in Horner's method to certain orthogonal polynomial bases. If the elements of the polynomial basis $\phi_k(z)$ are related by a three-term recurrence relation

$$\phi_k(z) = \alpha_k(z)\phi_{k-1}(z) - \beta_k\phi_{k-2}(z) \tag{2.36}$$

(the notation has changed from the previous section, to match the paper Smoktunowicz (2002), which we reference ahead) and $\phi_0(z) = 1$ and $\phi_1(z) = \alpha_1(z)$, where, for all the examples we are concerned with, the $\alpha_k(z)$ are linear polynomials in z and the β_k are constants, then a polynomial $p(z)$ expressed in this orthogonal basis as

$$p(z) = \sum_{k=0}^{n} b_k \phi_k(z) \tag{2.37}$$

can be evaluated in $O(n)$ flops by the *Clenshaw algorithm*, as follows.

Algorithm 2.2 The Clenshaw algorithm

```
Require: A value $z$, a nonnegative integer $n$, and a sequence $b_0, b_1, \ldots, b_n$ of coefficients
Require: The functions $\alpha_k(z)$ and the constants $\beta_k$
  $y_{n+1} := 0$
  $y_n := b_n$
  for $k$ from $n-1$ by $-1$ to 1 do
    $y_k := b_k + \alpha_{k+1}(z)y_{k+1} - \beta_{k+2}y_{k+2}$
  end for
  $p := (y_0 - \beta_2 y_2)\phi_0(z) + y_1\phi_1(z)$
  return The value of $p(z) = \sum_{k=0}^{n} b_k\phi_k(z)$
```

To see that this algorithm is correct, note that a *loop invariant* for the algorithm is the sum

$$p(z) = -\beta_{k+1}y_{k+1}\phi_{k-1}(z) + y_k\phi_k(z) + \sum_{j=0}^{k-1} b_j\phi_j(z)\,. \tag{2.38}$$

That is, before the start of the loop i.e. when $k = n$, this statement is trivially true because $y_{k+1} = 0$, and the update step changes the value of what will be the next y_k and replaces $y_{k+1}\phi_{k+1}(z)$ with $y_{k+1}(\phi_{k+1}(z) - \alpha_{k+1}\phi_k(z))$ or $-\beta_{k+1}y_{k+1}\phi_{k-1}(z)$. The process finishes when there are only two terms left, which sum to $p(z) = (b_0 - \beta_2 y_2)\phi_0(z) + y_1\phi_1(z)$ by the loop invariant.

Now, let us address the numerical stability of this method for the evaluation of polynomials:

Theorem 2.6 (Backward Stability of the Clenshaw Algorithm). *Under natural assumptions, evaluation of this algorithm is backward stable: that is, for a given z, the algorithm gives the exact evaluation of $p + \Delta p$, where the coefficients of $p + \Delta p$ are only slightly perturbed: $b_k + \Delta b_k$, where, with a modestly growing function L of n,*

$$|\Delta b_k| \le \mu_M L |b_k| \tag{2.39}$$

in the best scenario (this holds only for some bases and polynomials with nonincreasing coefficients b_k), and

$$\|\Delta b\|_\infty \le \mu_M L \|b\|_\infty \tag{2.40}$$

in the usual case.

In particular, for the Chebyshev polynomials, we have $L = O(n^2)$ in Eq. (2.39), showing that the Chebyshev basis evaluated by the Clenshaw algorithm has excellent backward-stability properties. The proof of this theorem is given in Smoktunowicz (2002).

2.2.6 Lagrange Polynomials

We now look at a very important *non*orthogonal basis family, the Lagrange bases. These are different to the previously discussed examples in that they are not *degree-graded*: Each element of a particular Lagrange basis has full degree, here n. Given $n+1$ distinct nodes τ_k, $0 \le k \le n$, define the numbers β_k by the partial fraction expansion

$$\frac{1}{w(z)} = \frac{1}{\prod\limits_{k=0}^{n}(z-\tau_k)} = \sum_{k=0}^{n} \frac{\beta_k}{z-\tau_k}. \tag{2.41}$$

Then, solving for the numbers β_k, we obtain

$$\beta_k = \prod_{\substack{j=0 \\ j\neq k}}^{n} (\tau_k - \tau_j)^{-1}. \tag{2.42}$$

Definition 2.4 (Lagrange polynomials). Given a set of nodes $\{\tau_k\}_{k=0}^{n}$ and the resulting numbers β_k,

$$\phi_k(z) = L_k(z) = \beta_k \prod_{\substack{j=0 \\ j\neq k}}^{n} (z-\tau_j) \tag{2.43}$$

is the kth Lagrange polynomial. $\lhd$

Note that, using the Kronecker delta, we can write

$$L_k(\tau_j) = \delta_j^k = \begin{cases} 0 & j \neq k \\ 1 & j = k \end{cases}, \tag{2.44}$$

and so for any polynomial $f(z)$ of degree at most n,

$$f(z) = \sum_{j=0}^{n} f(\tau_j) L_j(z). \tag{2.45}$$

Theorem 2.7. *The set of polynomials $L_j(z)$, for $0 \le j \le n$, forms a basis if the nodes τ_k, $0 \le k \le n$, are distinct.*

Proof. The theorem is equivalent (by definition) to the statement that the change-of-basis matrix $\mathbf{A}$ in $[L_0(z), L_1(z), \ldots, L_n(z)] = [1, z, z^2, \ldots, z^n]\mathbf{A}$ is nonsingular. That in turn is equivalent to the statement that the change-of-basis matrix in the other direction $[1, z, z^2, \ldots, z^n] = \mathbf{LB}$ is nonsingular, and this is easier. By the above formula, the entries of $\mathbf{B}$ are $B_{k,j} = \tau_k^j$. It is an (interesting) exercise to show that $\det \mathbf{B} = \prod_{j>k}(\tau_j - \tau_k)$, which is nonzero when the nodes are distinct. See Exercise 4.14. $\natural$

Corollary 2.1. *The only polynomial of degree at most n that satisfies* $f(\tau_i) = 0$ *for* $n+1$ *distinct nodes* τ_i, $0 \le i \le n$, *is the identically zero polynomial.*

Proof. Since the $L_j(z)$ form a basis, we may express an arbitrary polynomial of degree at most n as $\sum_{j=0}^{n} a_j L_j(z)$. Evaluating this polynomial at each of τ_k in turn gives $a_k = 0$, uniquely resulting in the identically zero polynomial. ♮

Remark 2.4. This corollary is part of the normal proof that interpolants are unique; we here see, doing things in a different order, that it is a corollary of the theorem we proved directly above. That is, this proof is done in a different order than usual but is equivalent. ◁

We will see shortly another notation for Eq. (2.45): With $\rho_i := f(\tau_i)$,

$$f(z) = \sum_{i=0}^{n} \rho_i L_j(z) = \sum_{i=0}^{n} \rho_i \frac{w(z)}{z - \tau_i} \beta_i = w(z) \sum_{i=0}^{n} \frac{\rho_i \beta_i}{z - \tau_i}\,. \tag{2.46}$$

This is the *first barycentric form* of a polynomial expressed in the Lagrange basis. We will see the second barycentric form in the exercises in this chapter and again in Chap. 8. The coefficients in the expansion of $f(z)$ in the Lagrange basis on $\tau_0, \ldots, \tau_n$ are simply the values $f(\tau_j)$.

The Lagrange polynomials are wonderfully useful, and we will use them every chance we get. An algorithm to compute polynomials in this basis is provided by Berrut and Trefethen (2004) (see Algorithm 2.3).

Algorithm 2.3 First barycentric form for evaluation of a Lagrange interpolating polynomial

```
Require: A value $z$, an integer $n > 0$, a vector of coefficients $\rho_k$, a vector of nodes $\tau_k$, and a
  precomputed vector of barycentric weights $\beta_k$
  if $z$ is identical to any $\tau_k$ then
    return $\rho_k$
  end if
  $w = 1$
  for $j$=0:$n$ do
    $w = w \cdot (z - \tau_j)$
  end for
  $p = 0$
  for j=0:n do
    $p = p + \beta_j \rho_j / (z - \tau_j)$
  end for
  return $w \cdot p$
```

2.2.6.1 Numerical Stability of the Barycentric Form

The numerical stability for Algorithm 2.3 is interesting. The paper (Higham 2004) shows that evaluation of this (first) barycentric form is nearly perfectly backward stable: The computed $\hat{p}(z)$ satisfies

$$\hat{p}(z) = \prod_{j=0}^{n}(z-\tau_j)\sum_{j=0}^{n}\frac{\beta_j\hat{\rho}_j}{z-\tau_j} = w(z)\sum_{j=0}^{n}\frac{\beta_j\hat{\rho}_j}{z-\tau_j}\,, \tag{2.47}$$

where each perturbed $\hat{\rho}_j$ satisfies

$$\hat{\rho}_j = \rho_j(1+\delta_j), \tag{2.48}$$

such that $|\delta_j| < \gamma_{5(n+1)}$. That is, provided n isn't so large that it is $O(1/\mu_M)$, the computed sum is the *exact* value of a polynomial passing through only slightly different data values.

Remark 2.5. This result is one of the most important *backward-stability* results presented in this book. What the paper (Higham 2004) provides is a *guarantee* that evaluating the first barycentric form will *always* produce the exact value of a polynomial of the same form as the one we started with, with at most only slightly perturbed data. This result should be compared with the similar result quoted from Smoktunowicz (2002) for orthogonal polynomials, and contrasted with the results of the *forward* error analysis for Horner's method presented in Sect. 2.2.1. ◁

2.2.6.2 Change-of-Basis from a Lagrange Basis

The change-of-basis matrices are deceptively simple *from* a Lagrange basis. We say "deceptively" because the change-of-basis itself may be ill advised because of difficulties related to the conditioning of the matrix, as we will see. However, if it is desired (in spite of misgivings) to perform the change of basis, it is, in theory, simple to carry out. Because any polynomial can be written using Eq. (2.45), each element of a different basis, say $\phi_k(z)$, may be written as

$$\phi_k(z) = \sum_{j=0}^{n}\phi_k(\tau_j)L_j(z)\,. \tag{2.49}$$

That is, the change-of-basis matrix *from* a Lagrange basis is just (in the four-by-four case for simplicity)

$$\begin{bmatrix}\phi_0(z)\\ \phi_1(z)\\ \phi_2(z)\\ \phi_3(z)\end{bmatrix} = \begin{bmatrix}\phi_0(\tau_0) & \phi_0(\tau_1) & \phi_0(\tau_2) & \phi_0(\tau_3)\\ \phi_1(\tau_0) & \phi_1(\tau_1) & \phi_1(\tau_2) & \phi_1(\tau_3)\\ \phi_2(\tau_0) & \phi_2(\tau_1) & \phi_2(\tau_2) & \phi_2(\tau_3)\\ \phi_3(\tau_0) & \phi_3(\tau_1) & \phi_3(\tau_2) & \phi_3(\tau_3)\end{bmatrix}\begin{bmatrix}L_0(z)\\ L_1(z)\\ L_2(z)\\ L_3(z)\end{bmatrix}, \tag{2.50}$$

or, more compactly,

$$\boldsymbol{\varphi}(z) = \mathbf{V}\mathbf{L}(z)\,. \tag{2.51}$$

In the particular case when $\phi_k(z) = z^k$, $\mathbf{V}$ is called a *Vandermonde* matrix, and it is nonsingular precisely when the nodes τ_k are distinct. For other $\phi_k(z)$, $\mathbf{V}$ is called a *generalized Vandermonde matrix* (see, for example, Problems 8.36 and 8.37).

The Vandermonde matrices occurs often enough that, for emphasis, we will display one here:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ \tau_0 & \tau_1 & \tau_2 & \tau_3 \\ \tau_0^2 & \tau_1^2 & \tau_2^2 & \tau_3^2 \\ \tau_0^3 & \tau_1^3 & \tau_2^3 & \tau_3^3 \end{bmatrix}. \tag{2.52}$$

That matrix can be generated by the MAPLE command

```
V := Matrix( 4, 4,
      shape = Vandermonde[ [tau[0], tau[1], tau[2], tau[3]] ]  );
latex( LinearAlgebra:-Transpose(V) );
```

The convention of needing the transpose to get a "Vandermonde" matrix in this notation agrees with the use in Higham (2002); however, in Chap. 8, we often use the alternative.

In Chap. 8, we will extend the Lagrange basis to the *Hermite interpolational basis*, which allows some of the nodes τ_k to coalesce or "flow together," in which case we talk about *confluency*. This basis is quite distinct from the Hermite orthogonal basis mentioned briefly earlier, and is not to be confused with it. For change of bases to other bases, one talks about *confluent* Vandermonde matrices. This will be taken up later.

Multiplication and division of polynomials expressed in a Lagrange basis are not yet widely encountered in practice[8]; but multiplication is simple enough, provided there are enough data to represent the product (one needs $nm+1$ points if the degrees of the multiplicands are n and m). The entries are simply $f(\tau_i)g(\tau_i)$.[9]

2.2.6.3 The Degree of Difficulty

Given a polynomial expressed in a Lagrange basis, what is its degree? Clearly, if we have enough points to capture the polynomial (say $n+1$), then the degree is *at most* n. But it may very well be less than that, and knowledge of the actual degree can be quite important. We return to this problem in Sect. 11.8, but for now we note a lemma that you are asked to prove in Exercise 2.12.

Lemma 2.1. *If a polynomial $f(z)$ of degree at most n has the values ρ_k on the $n+1$ distinct nodes τ_k, for $0 \le k \le n$, then the degree of $f(z)$ is exactly n if*

$$\sum_{k=0}^{n} \beta_k \rho_k \neq 0\,, \tag{2.53}$$

where the β_k are the barycentric weights of the nodes.

[8] We believe this will change, as the realization that working directly in a Lagrange basis is a good idea gradually percolates through the communities.

[9] Division with remainder, on the other hand, requires solving an overspecified linear system, in order to enforce the degree constraints; see Amiraslani (2004).

Proof. Left as Exercise 2.12.

What happens if this is not zero, but small relative to $\|\boldsymbol{\rho}\|$, the norm of the vector of values of $f(z)$ on the nodes? Does this mean that $f(z)$ is "nearly" of lower degree? We do not give a complete answer to this, but rather note only that the generic case with values ρ_i on nodes τ_i is that the degree is exactly n; if the values of a low-degree polynomial are perturbed by arbitrarily small amounts, then almost certainly the perturbed values are the exact values of a degree-n polynomial.

But how close are the given values, then, to the values of a lower-degree polynomial? This question has been addressed (using the lemma above) in Rezvani and Corless (2005), and using the witness vector in Hölder's inequality, we can find an analytic solution in the case the nearby polynomial is of degree *one* less than n, in a manner similar to what we use later in Theorem 2.9. For still-lower degrees, a computational procedure is available. We take a different tack here and give an alternative characterization of the degree of a polynomial.

In exact arithmetic, a polynomial of degree n has exactly n complex roots, counting multiplicity. But numerically, the condition number $B(z)$ grows as $|z|$ grows, so the location of *large* roots is often very sensitive to perturbations. If a polynomial $p(z)$ can be perturbed by a small amount in such a way that some large roots go out to infinity, then the original polynomial is somehow close to a lower-degree polynomial. More precisely, we define the *ε-degree* of a polynomial $p(z) = \sum_{k=0}^{n} c_k \phi_k(z)$ expressed in a basis $\boldsymbol{\phi}$ with weights $w_k \geq 0$ not all zero as

$$\deg_\varepsilon(p) = \min\left\{ \deg(p + \Delta p) \,\middle|\, |\Delta c_k| \leq w_k \varepsilon \right\} . \tag{2.54}$$

In a degree-graded basis, the computation is easy; for the Lagrange and Hermite interpolational basis, it is not quite so easy. For the Bernstein–Bézier basis (to be introduced below), quite a bit of attention has been paid to this issue in the CAGD literature, and we refer you, for instance, to Farin (1996). The choice of norm for nearness they use is not a coefficientwise norm as we use here, but rather a function norm. Nonetheless, the ideas are similar.

2.2.7 Bernstein–Bézier Polynomials

The following family of polynomials is a basis for polynomials of degree n and is positive on the interval $a < z < b$. Like the Lagrange and Hermite interpolational bases, these are not degree-graded: Each element of the basis is degree n.

$$\phi_k(z) = (b-a)^{-n} \binom{n}{k} (z-a)^k (b-z)^{n-k} . \tag{2.55}$$

The Bernstein–Bézier polynomials of degree 8 or less are displayed in Fig. 2.2. These are extremely useful in computer-aided geometric design. To evaluate poly-

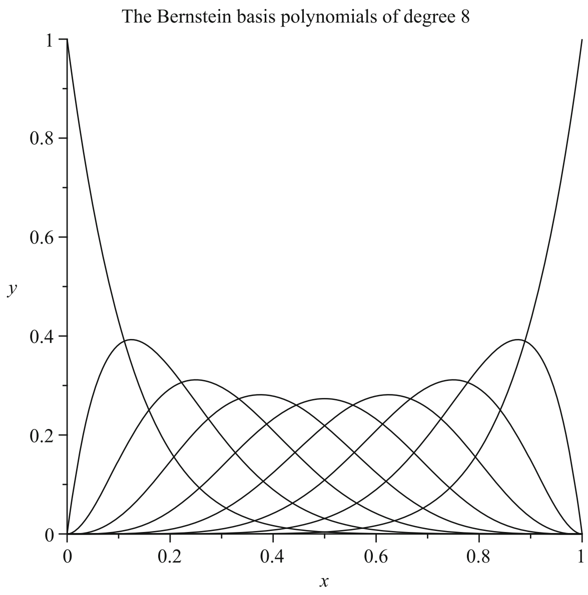

Fig. 2.2 Bernstein–Bézier basis polynomials of degree at most 8

nomials expressed in this basis, you may use de Casteljau's algorithm, as discussed, for example, in Tsai and Farouki (2001). We do not pursue that algorithm further here, except to note that it is partially implemented in MAPLE.[10] This basis has a number of interesting properties, including an *optimal conditioning* property, that we discuss in Chap. 8.

2.3 Condition Number for the Evaluation of Polynomials

Now, let us look at the *condition number* for evaluation of polynomials. This is studied in many works (for example, in de Boor (1978)), but we take the following formulation from Farouki and Rajan (1987).

Theorem 2.8. *If we consider a polynomial*

$$f(z) = \sum_{k=0}^{n} c_k \phi_k(z) \tag{2.56}$$

with coefficients c_k in the base $\{\phi_k(z)\}_{k=0}^{n} = 0^n$, and a perturbed polynomial

[10] The Bernstein–Bézier basis is not yet well supported in MAPLE: For serious use, we recommend instead the package described in Tsai and Farouki (2001).

$$(f+\Delta f)(z) = \sum_{k=0}^{n} c_k(1+\delta_k)\phi_k(z) \tag{2.57}$$

with perturbed coefficients $c_k(1+\delta_k)$*, then*

$$|\Delta f(z)| \le \left(\sum_{k=0}^{n} |c_k||\phi_k(z)|\right) \cdot \max_{0\le k\le n} |\delta_k| \, . \tag{2.58}$$

If we let $B(z) = \sum_{k=0}^{n} |c_k||\phi_k(z)|$*, we have the simple inequality*

$$|\Delta f(z)| \le B(z) \max_{0\le k\le n} |\delta_k| \, . \tag{2.59}$$

Here is a compact proof of this very important theorem, which we will use repeatedly in this book:

Proof. For the error term Δf, we have

$$\Delta f(z) = \sum_{k=0}^{n} c_k\delta_k\phi_k(z) = \left[c_0\phi_0(z), c_1\phi_1(z), \cdots, c_n\phi_n(z)\right] \begin{bmatrix} \delta_0 \\ \delta_1 \\ \vdots \\ \delta_n \end{bmatrix} \, . \tag{2.60}$$

In this form, we can use Hölder's inequality (Steele 2004): If $1/p + 1/q = 1$, then

$$|\mathbf{a}\cdot\mathbf{b}| \le \|\mathbf{a}\|_p\|\mathbf{b}\|_q \, . \tag{2.61}$$

The result follows directly if we take $\mathbf{a} = [c_0\phi_0(z), \ldots, c_n\phi_n(z)]$, $\mathbf{b} = [\delta_0, \delta_1, \ldots, \delta_n]$, $p = 1$, and $q = \infty$. $\natural$

The number $B(z)$ (and for fixed z, it is indeed just a number) thus serves as an absolute *condition number* for evaluation of the polynomial f at the point z: If we change the coefficients c_k by a relative amount $|\delta_k| \le \varepsilon$, this means that the value of f might change by as much as $\varepsilon B(z)$. Higham (2004) uses instead $B(z)/|f(z)|$, which is a *relative* condition number, and indeed this may be more informative in many situations.

Remark 2.6. This is the first derivation of an explicit general formula in this book for a *condition number*, which was defined for general computation in Sect. 1.4.2 and used earlier to express the error results for Horner's method. This notion is usually introduced in numerical analysis texts not with polynomial evaluation as we have done here, but rather with the solution of linear systems of equations (which we begin in Chap. 4). The notion is perhaps the most important in the book, and the reader will see it in every single chapter. The reader is urged to make a note of this usage here, and later in Chap. 3, and again in Chap. 4; after that, return and reread Sect. 1.4.2 before going on. $\triangleleft$

Example 2.3. As an example, we take a single polynomial, $f(t)$, and plot its condition number (2.59) in several different bases. Consider the polynomial $f(t)$ that takes on the values $\boldsymbol{\rho} = [1, -1, 1, -1]$ on $\boldsymbol{\tau} = [-1, -1/3, 1/3, 1]$. Its Lagrange form is

$$f(t) = -\frac{9}{16}\left(t+\frac{1}{3}\right)\left(t-\frac{1}{3}\right)(t-1) - \frac{27}{16}(t+1)\left(t-\frac{1}{3}\right)(t-1) \\ -\frac{27}{16}(t+1)\left(t+\frac{1}{3}\right)(t-1) - \frac{9}{16}(t+1)\left(t+\frac{1}{3}\right)\left(t-\frac{1}{3}\right), \quad (2.62)$$

while its monomial form is

$$f(t) = -\frac{9}{2}t^3 + \frac{7}{2}t, \quad (2.63)$$

and its Newton form, if the nodes are taken in the left-to-right order in which they are given, is

$$f(t) = -2 - 3t + \frac{9}{2}(t+1)\left(t+\frac{1}{3}\right) - \frac{9}{2}(t+1)\left(t+\frac{1}{3}\right)\left(t-\frac{1}{3}\right). \quad (2.64)$$

If instead we use the Leja ordering of the nodes (see Chap. 8, Exercise 4), namely, $[-1, 1, -1/3, 1/3]$, the form is

$$f(t) = -t + \frac{3}{2}(t+1)(t-1) - \frac{9}{2}(t+1)\left(t+\frac{1}{3}\right)(t-1). \quad (2.65)$$

For each of these, $B(t)$ is simply the sum of the absolute values of the terms. The results are displayed in Fig. 2.3. ◁

Remark 2.7. In Fig. 2.3, we see $B(t)$ plotted for all but that for Eq. (2.64), which is so large (going up to 25) that it would compress the graph. This example is hardly unique: The Newton basis is often poorly conditioned and, moreover, depends on the ordering of the nodes. We will pursue this in great detail in the exercises, and again in Chap. 8. This book differs from many numerical analysis texts in that it avoids use of the Newton basis for this reason and uses the Lagrange and Hermite interpolational bases instead, which are often better conditioned. This understanding in a broad popular sense is a relatively recent development and is due to the papers Berrut and Trefethen (2004) and Higham (2004), although in a more limited sense, it was known previously. ◁

Example 2.4. Let us continue Example 2.3 with another basis, namely, the Bernstein–Bézier basis, given by

$$\phi_k(z) = (b-a)^{-n}\binom{n}{k}(z-a)^k(b-z)^{n-k}. \quad (2.66)$$

For polynomials of degree 3 on the interval $-1 \le z \le 1$, we just let $a = -1$, $b = 1$, and $n = 3$, so that the basis elements are easily computed. We then find that if

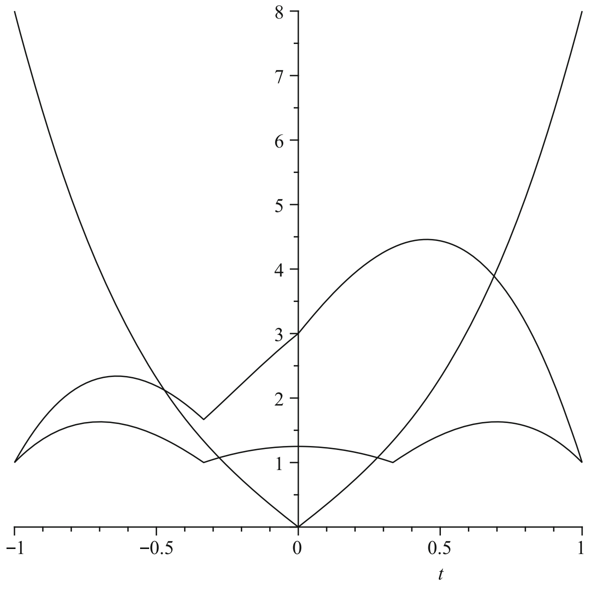

Fig. 2.3 Condition number of evaluation of a particular degree-3 polynomial in three different bases: Lagrange (which is best), Newton with the Leja ordering (which is next-best), and standard monomials (which is worst in this example)

$$f(z) = \sum_{k=0}^{3} c_k \phi_k(z)\,,$$

$c_0 = 1$, $c_1 = {}^{-17}\!/\!{}_3$, $c_2 = {}^{17}\!/\!{}_3$, and $c_3 = -1$. Then, the condition number

$$\sum_{k=0}^{3} |\, c_k \,| \cdot |\, \phi_k(z) \,|$$

is displayed in Fig. 2.4, where it is shown with the condition numbers from Fig. 2.3. ◁

Remark 2.8. After this discussion, it is clear that the same polynomial will have different condition numbers in different bases. It is shown in Farouki and Goodman (1996) that among all polynomial bases that are nonnegative on an interval, the Bernstein–Bézier basis has optimal condition numbers in a generic sense. Taken as a whole, one can expect a polynomial to have a smaller condition number in the Bernstein–Bézier basis than in any other nonnegative basis. The Farouki–Goodman theorem thus guarantees, for example, that the Bernstein–Bézier basis is better than the monomial basis in general. For a *particular* polynomial, however, this need not be true, as we have seen in the previous example. In Chap. 8, we will show,

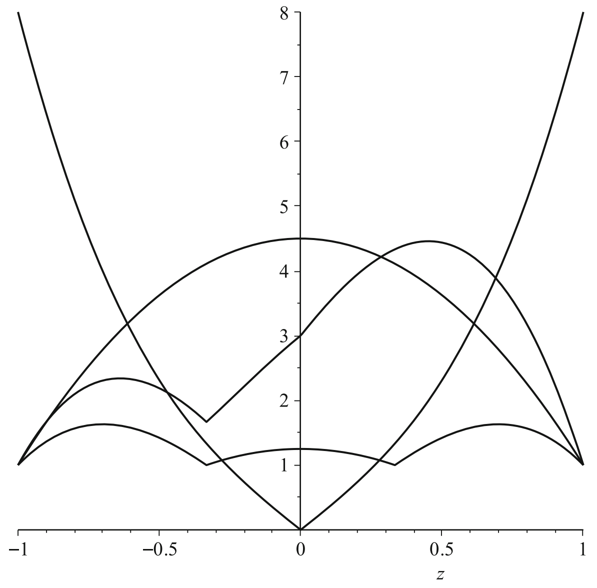

Fig. 2.4 Condition numbers from Fig. 2.3, together with the condition number of the same polynomial expressed in the Bernstein–Bézier basis. This graph shows that, for this example, the Bernstein–Bézier basis is worse than the Lagrange basis and comparable to the Newton basis with Leja ordering

using the same techniques as Farouki and Goodman (1996) used, that if we relax the nonnegativity condition, then the Lagrange basis has the same optimality property. This in some sense explains why the Lagrange basis did so well in Example 2.3. ◁

Remark 2.9. The question of "which basis is best overall?" is somewhat vexed. The answer is, "It really depends on the problem, and what information you want." In the simple example above, it is clear that the Lagrange basis has a lower condition number than either of the Newton bases or the Bernstein–Bézier bases, over the entire interval. The monomial basis, however, is better than the Lagrange basis, for all z "near enough" to the origin. For this problem, the conditioning of the Lagrange basis expression is better for "most" of the z in this restricted interval.

The picture would change if we took a different interval, or if we considered instead the complex disk $|z| < 1$ (where, in fact, the monomial basis would show itself to good advantage). It is the position of this book that the Lagrange basis is *generally* to be preferred over other bases, on the principle that you probably have sampled your polynomial where you know it best; while the Bernstein–Bézier basis is provably the best on an interval for generic polynomials (and is widely used in CAGD in part because of that); and that the monomial basis is likely overused—that is, often used where it shouldn't be—but can be the best tool for the job.

The relative condition number ${}^{|B(z)|}\!/\!{}_{|f(z)|}$ is also of interest (perhaps of more interest). Since this example polynomial has a zero at $t = 0$, the monomial basis condition number shows itself to be best there—$B_{\text{monomial}}(0) = 0$ as well, whereas all the other absolute condition numbers are nonzero at zero, and so the relative condition of the monomial basis is the only finite one there.

Finally, the flexibility and uniform approximation properties of discrete Chebyshev bases—that is, Lagrange interpolation on nodes that are the zeros of Chebyshev polynomials—make them extremely interesting to the computational scientist. See the Chebfun package as described in Battles and Trefethen (2004) and subsequent papers. $\triangleleft$

Example 2.5. As we have seen in Chap. 1, the Airy function has a Taylor series that converges for all z; it can be written as

$$\mathrm{Ai}(z) = 3^{-2/3} \sum_{n=0}^{\infty} \frac{z^{3n}}{9^n n!(n - 1/3)!} - 3^{-4/3} \sum_{n=0}^{\infty} \frac{z^{3n+1}}{9^n n!(n + 1/3)!}\,. \tag{2.67}$$

Consider using the degree-127 truncation of this Taylor series as a way of approximating $\mathrm{Ai}(z)$ for various z. In applications, for instance, the geometric optics of the rainbow, the zeros of $\mathrm{Ai}(z)$ are often important, so we want to accurately assess it there. So let us focus on values near $z = -7.94$, which is somewhat close to a zero of $\mathrm{Ai}(z)$. A preliminary analysis shows that, in theory, the degree-127 truncation has *more* than enough terms for convergence—here, we ought to be able to get about 25 significant figures of $\mathrm{Ai}(-7.94)$ if we want. We are using so many terms here in part to show that the mathematical theory of convergence is not at issue for this example. Write the degree-127 polynomial in Horner form (Exercise 2.29 contains two programs that implement an efficient, specific variation of Horner form for this particular polynomial).

If we use only 8 digits in our computation, because we only want 8 digits in our answer, we get $\mathrm{Ai}(-7.94) = 0.00359469$. Here is how the (general) Horner form begins, with 8-digit coefficients:

$$0.35502806 + z(-0.25881940 + z(0.059171345 + \cdots))\,. \tag{2.68}$$

If we use 16 digits, we get a different answer, beginning with

$$\mathrm{Ai}(-7.94) = 0.0039158060872139\,.$$

Only a single significant digit was right the first time! If we don't use Horner's form, the answer is worse, by the way. In order to understand what has gone wrong with the 8-digit computation, we need to plot the $B(z)$ function, which is the same Taylor series but with all positive signs and with powers of $|z|$, not z. This is plotted in Fig. 2.5. We see that the $B(z)$ function becomes very large, for large z: We say that the (monomial basis) polynomial approximation to the Airy function that we derived from Taylor series is ill-conditioned to evaluate for large $|z|$. This point deserves emphasis: Taylor series about the origin are often *impractical to use* for large $|z|$ because the resulting polynomial expression, although adequate in theory to deliver accuracy, is far too *ill-conditioned to use*. The condition number we see at the right is about 10^{15}—10 rounding errors, and we cannot count on any accuracy in the result (and since we are adding up hundreds of terms, we will make many more than 10 rounding errors). Near $z = -7.94$, the condition number is about 10^9. This phenomenon is sometimes known as “the hump” (see Exercise 2.16).

There is a bit more to say, for this example, though: If we take each separate series in Eq. (2.67), the one multiplied by $3^{-2/3}$ (call it $f_1(z)$) and the other multiplied by $3^{-4/3}$ (call it $f_2(z)$), and plot *their* condition numbers, we see that each of them has condition number 1 for $z > 0$ (because all terms are positive). So we can say that each of them is accurately evaluated for $z > 0$. (This is how the programs in Exercise 2.29 do it, by the way.) Yet the condition number for $\mathrm{Ai}(z) = 3^{-2/3} f_1(z) - 3^{-4/3} f_2(z)$ grows very large, even for positive z. This is because each of $f_1(z)$ and $f_2(z)$ grows very large for large positive z, while $\mathrm{Ai}(z)$ gets very small indeed—that is, we are computing $\mathrm{Ai}(z)$ as the tiny difference of two large numbers. This is a recipe for catastrophic cancellation. Notice that this shows up automatically in the condition number analysis: we have discovered directly that the condition number of this polynomial is very large. We defer analysis of the condition number of $\mathrm{Ai}(z)$ itself to Chap. 3 (Exercise 3.6). $\triangleleft$

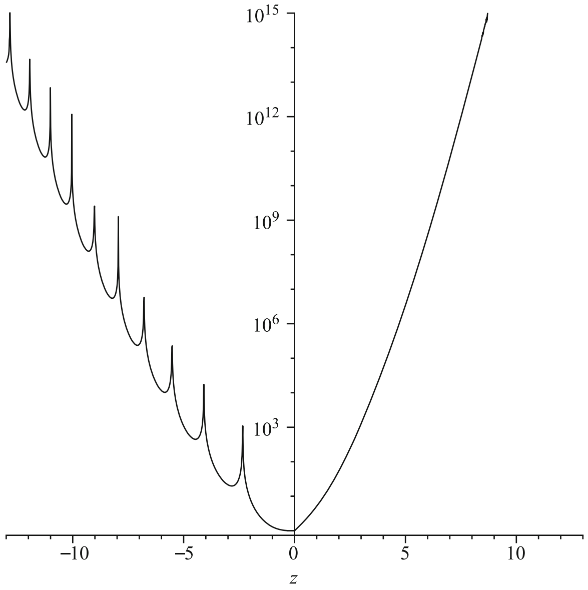

Fig. 2.5 The condition number for evaluation of the degree-127 Taylor polynomial for the Airy function $\mathrm{Ai}(z)$ on $-13 \le z \le 13$. Note that it goes to infinity in very narrow spikes (the graph only shows a portion of each spike since the singularity is so narrow) around each zero, but even away from zeros, the condition number grows very rapidly with $|z|$

2.4 Pseudozeros

We now look at the relationship between the condition number for *evaluation* of polynomials and the condition number for *rootfinding* for polynomials. In mathematical terms, given $\varepsilon > 0$ and weights $w_i \geq 0$, $0 \leq i \leq n$, not all zero, define the set of *pseudozeros*

$$\Lambda_\varepsilon(f) = \left\{ z \,\middle|\, (f + \Delta f)(z) = 0, \text{ where } \Delta f = \sum_{i=0}^{n} \Delta c_i \phi_i(z) \text{ and each } |\Delta c_i| \leq \varepsilon w_i \right\}.$$

These are the roots of the polynomials that are near f. Studying this set will help us to understand what happens if the coefficients of our polynomials are changed somehow (perhaps due to measurement error, or to numerical errors in computation). To do so, we make use of the following theorem:

Theorem 2.9. *Let $\Lambda_\varepsilon(f)$ be defined as above. Then*

$$\Lambda_\varepsilon(f) = \left\{ z \,\middle|\, \left|(f(z))^{-1}\right| \geq (\varepsilon B(z))^{-1} \right\}, \tag{2.69}$$

where $B(z) = \sum_{i=0}^{n} w_i |\phi_i(z)|$ or, equivalently for scalar polynomials,

$$\Lambda_\varepsilon(f) = \left\{ z \,\middle|\, |f(z)| \leq \varepsilon B(z) \right\}. \tag{2.70}$$

Proof. First, suppose $z \in \Lambda_\varepsilon(f)$. Moreover, if $|\Delta c_i| \leq \varepsilon w_i$, and $\Delta f(z) = \sum_{i=0}^{n} \Delta c_i \phi_i(z)$, then

$$|\Delta f(z)| \leq \sum_{i=0}^{n} |\Delta c_i||\phi_i(z)| \leq \sum_{i=0}^{n} \varepsilon w_i |\phi_i(z)| = \varepsilon B(z),$$

so that $\Lambda_\varepsilon(f) \subseteq \left\{ z \,\middle|\, |f(z)| \leq \varepsilon B(z) \right\}$. Now, suppose $|f(z)| \leq \varepsilon B(z)$. Take

$$\Delta c_i = -\mathrm{signum}\left(\overline{\phi_i(z)}\right) w_k \frac{f(z)}{B(z)}.$$

Then it follows that

$$|\Delta c_i| = \left| w_i \frac{f(z)}{B(z)} \right| \leq \varepsilon w_i \frac{B(z)}{B(z)} = \varepsilon w_i.$$

Also, observe that

$$f(z) + \sum_{i=0}^{n} \Delta c_i \phi(z) = f(z) + \sum_{i=0}^{n} \frac{-w_i |\phi_i(z)| f(z)}{B(z)}$$

$$= f(z) - \frac{f(z)}{B(z)} \sum_{i=0}^{n} w_i |\phi_i(z)|$$
$$= f(z) - f(z) = 0\,.$$

Thus, the set identity is obtained. ♮

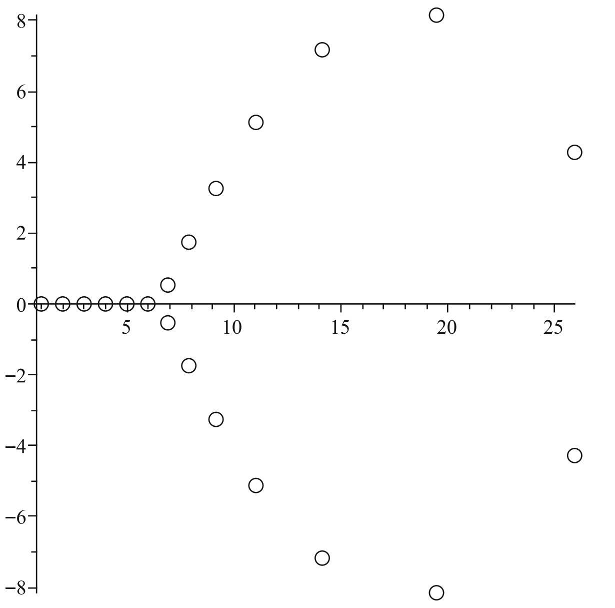

Fig. 2.6 Zeros of a small perturbation of the Wilkinson polynomial $W(z) = \prod_{k=1}^{20}(z-k)$, after first having been expanded into the monomial basis $W_k(z) = z^{20} - 210z^{19} + \cdots$

Remark 2.10. It is no coincidence that the condition number of Theorem 2.8 appears as the expansion factor in equations in the proof of Theorem 2.9. An ill-conditioned polynomial, with large B, will have its roots spread widely when the coefficients are changed.

This is a very useful and important result: It says that if $|f(z)|$ is *small*, then z is the exact root of a *nearby* polynomial $f(z) + \Delta f(z)$. Note that this works for only one root at a time. ◁

Example 2.6. The Wilkinson polynomial[11] can be written as

$$W(z) = \prod_{k=1}^{20}(z-k) = z^{20} - 210z^{19} + \cdots + 20! \tag{2.71}$$

[11] See, for instance, Wilkinson (1984)

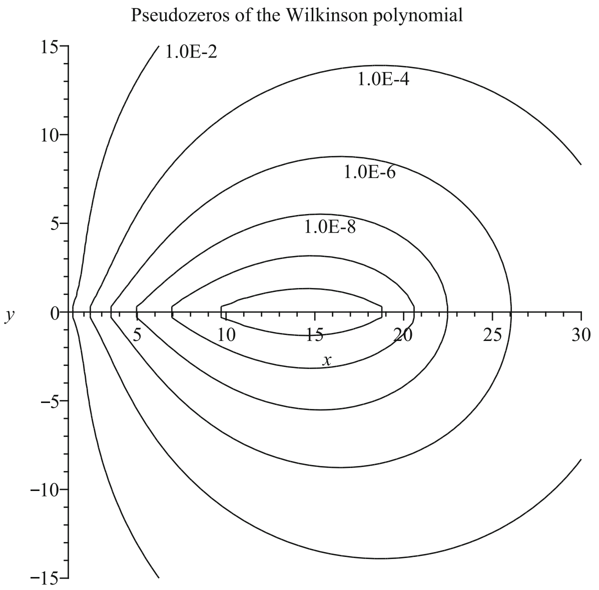

Fig. 2.7 Pseudozeros of the Wilkinson polynomial expressed in the monomial basis: a plot of the contours of $\frac{|f(x+iy)|}{B(x+iy)}$

when expanded into the monomial basis. Compare Figs. 2.6 and 2.7. In the latter case, we see contours bounding the sets of zeros of *all* perturbations of the Wilkinson polynomial expressed in the monomial basis—as we see, quite small perturbations can have dramatic effects on the location of the zeros. In the former, we have an explicit perturbation of one coefficient (again in the monomial basis), and the perturbed zeros lie pretty much along one of the contours of the latter. In contrast, if we *don't* expand it, then the natural basis to recognize in which it is written is the Lagrange basis on the nodes 1, 2, ..., 20, and (say) 0, so $W(z) = 20!\, L_0(z)$ has only one nonzero coefficient. The condition number in *this* basis is, remarkably, 0 at all the roots, in an absolute sense; in a relative sense, the condition number is just 1. Of course, this is unsurprising, and uninformative: If we know the roots, then they are easy to find. However, there is something more useful in this observation than just that, and we return to it later. ◁

2.5 Partial Fractions

Every reader of this book will have encountered the partial fraction decomposition. However, it is all too common to encounter only the hand practice, and not the theory or a practical algorithm. We give the theory here, and later we give a practical algorithm for the easy case that we need for interpolation. The basic object under study here is the class of rational functions and a simple representation for them.

Definition 2.5 (Rational function). A function $f:\mathbb{C}\to\mathbb{C}$ is called a *rational function* if there exist polynomials $p(z)$ and $q(z)$ such that

$$f(z)=\frac{p(z)}{q(z)} \qquad \forall z\in\mathbb{C}\,, \tag{2.72}$$

except possibly at the zeros of $q(z)$. If the degree of $p(z)$ is n and the degree of $q(z)$ is m, then we say that $f(z)$ is here represented as an $[n,m]$-degree rational function. By convention, the identically zero function $f(z)\equiv 0$ is also called a rational function, for example, taking $q(z)=1$. $\triangleleft$

Rational functions often arise in approximation theory. One class of these are called Padé approximants:

Definition 2.6 (Padé approximant). A *Padé approximant* to a function $f(z)$ is a rational function whose coefficients are wholly determined by matching the Taylor series of $f(z)$. $\triangleleft$

We can find a unique representative $^{p(z)}/_{q(z)}$ for a rational function $f(z)$ by insisting that $p(z)$ and $q(z)$ have no common factors (dividing out the GCD) and normalizing one of $f(z)$ and $q(z)$ in some way—often by making $f(z)$ monic by dividing the numerator and the denominator by the leading coefficient of $q(z)$, or by making the norm of the vector of coefficients of $q(z)$ equal to 1 and insisting that one particular coefficient be positive, or simply by insisting that $\mathbf{a}\cdot\mathbf{q}=1$ for some given nonzero vector $\mathbf{a}$, where $\mathbf{q}$ means the vector of coefficients of the polynomial $q(z)$ expanded in the monomial basis. Moreover, since the polynomials are linear in their coefficients, we may divide the numerator and denominator by a constant in order to make this dot product 1.

We also usually insist that $\deg p<\deg q$, by first doing polynomial division if necessary: $p=Qq+R$ and so $^{p}/_{q}=Q+{}^{R}/_{q}$, separating out a polynomial part $Q(z)$ and leaving a rational part $^{R(z)}/_{q(z)}$ with the desired "proper form."

We now state and prove the key theorem of this section:

Theorem 2.10 (Partial fraction decomposition). *Suppose we have already found machine number representations of all the roots of the denominator,*[12] *and that*

$$q(z)=\prod_{k=0}^{n}(z-\tau_k)^{s_k} \tag{2.73}$$

is the unique factorization of monic $q(z)$ into distinct factors over $\mathbb{C}$. That is, $\tau_i=\tau_j \Leftrightarrow i=j$, and each integer $s_k\geq 1$. Let $m=\sum_{k=0}^{n}s_k$. Note that the degree of $q(z)$ is exactly m and that $q(z)$ is not identically zero (if the product is empty with $n<0$, then $q(z)=1$ by convention). Suppose that $m\geq 1$. Suppose $p(z)$ is a polynomial

[12] Of course, this avoids the hard questions of how to do this if we start with $q(z)$ expressed in some other basis, and also the hard question of what are the consequences of approximating polynomial roots by machine numbers. But for the interpolation applications that we need in this book, this assumption suffices.

with $\deg p < m$ *having no common factor with* $q(z)$. *Then there exist* m *numbers* $\alpha_{i,j}$ *(with* $0 \le i \le n$ *and* $0 \le j \le s_i - 1$*) such that* $\forall z \notin \{\tau_0, \tau_1, \dots \tau_n\}$,

$$\frac{p(z)}{q(z)} = \sum_{i=0}^{n} \sum_{j=0}^{s_i-1} \frac{\alpha_{i,j}}{(z-\tau_i)^{j+1}} . \tag{2.74}$$

The numbers $\alpha_{i,j}$ *provide the decomposition.*

Proof. We proceed by induction on the degree $m \ge 1$, which gives a perfectly satisfactory algorithm to use in hand computation. The base of the induction, $m = 1$, is trivial: $\alpha_{0,0} = p_0 = p(z)$, because $\deg p = 0$ and there is nothing to prove. Suppose now that the theorem is true for all polynomials with degrees $m-1$ or less. Let

$$\alpha_{0,s_0-1} = p(\tau_0) \prod_{k=1}^{n} (\tau_0 - \tau_k)^{-s_k}$$

and consider

$$\frac{p(z)}{q(z)} - \frac{\alpha_{0,s_0-1}}{(z-\tau_0)^{s_0}} = \frac{p(z) - \alpha_{0,s_0-1} \prod_{k=1}^{n} (z-\tau_k)^{s_k}}{q(z)} .$$

We claim that the numerator and denominator on the right have a nontrivial common divisor, $z - \tau_0$. Since $s_0 \ge 1$, it is clear that $(z-\tau_0) \mid q(z)$. It only remains to show that this factor divides the numerator. It is equivalent to show that τ_0 is a zero of the numerator. But this is obvious, because the polynomial at $z = \tau_0$ has the value $p(\tau_0) - p(\tau_0) \prod_{k=1}^{n} (\tau_0 - \tau_k)^{-s_k} \prod_{k=1}^{n} (\tau_0 - \tau_k)^{s_k} = 0$.

On dividing the numerator and denominator on the right by $z - \tau_0$ (as many times as necessary but certainly at least once), we are left with a proper rational function on the right with denominator of the form (2.73) and having degree strictly less than m. By the induction hypothesis, this can be expressed uniquely in partial fraction form, thus completing the proof of the theorem. $\natural$

Remark 2.11. This proof provides an excellent hand algorithm: see Scott and Peeples (1988). It is also "self-checking": If exact division does *not* occur at the second step, we know that we have made an arithmetic blunder. $\triangleleft$

Example 2.7. Consider

$$\frac{1}{z^2(z-1)^2} = \frac{1}{z^2} + \mathrm{Rest}(z)$$

on taking out the leading term in z as $z \to 0$. Rearranging as in the proof of the theorem, we get

$$\begin{aligned} \mathrm{Rest}(z) &= \frac{1}{z^2(z-1)^2} - \frac{1}{z^2} = \frac{1-(z-1)^2}{z^2(z-1)^2} = \frac{1-z^2+2z-1}{z^2(z-1)^2} \\ &= \frac{z(2-z)}{z^2(z-1)^2} \qquad (2.75) \\ &= \frac{2-z}{z(z-1)^2} \qquad (2.76) \end{aligned}$$

and this is a proper rational function with a degree-3 denominator, one less than we started with. As stated previously, the exact cancellation from Eq. (2.75) to Eq. (2.76) is necessary, and if it doesn't happen, then we know that we have made an arithmetic blunder.

The process can be continued, to get

$$\frac{1}{z^2(z-1)^2} = \frac{1}{z^2} + \frac{2}{z} - \frac{2}{z-1} + \frac{1}{(z-1)^2}\,. \tag{2.77}$$

The numerators on the right-hand side are the $\alpha_{i,j}$ desired. ◁

Remark 2.12. Later we will need this particular partial fraction decomposition many times: It is the foundation for cubic Hermite interpolation. The reader is urged to complete the computation above and confirm Eq. (2.77). ◁

This algorithm can be implemented recursively, once an algorithm for division of polynomials by linear factors $z-\tau_k$ has been made available (and, of course, this can be done in any polynomial basis). For our purposes in this book, however, there is a more practical algorithm for partial fractions, based on *local Taylor series*. In order to develop that algorithm (and indeed for many other numerical purposes), we need to learn to manipulate formal power series, and so we turn to this in the next section.

Remark 2.13. MAPLE has several commands to compute partial fraction decompositions. Using exact arithmetic, the command

```
convert( R, parfrac, z, true );
```

does the trick (the `true` flag means that the rational function R has already had its denominator factored). However, at the time of this writing, for floating-point arithmetic, this command is not always satisfactory, because it converts internally to a monomial basis centered at 0, and this can induce numerical instability in the algorithm because the intermediate monomial basis representations are ill-conditioned. See Exercise 2.28. ◁

2.6 Formal Power Series Algebra

Numerical methods rely heavily on Taylor series. In this section we give a short generalized reminder[13] of how to operate on them. Suppose that, instead of being given a function, we are directly given the series and that we want to do standard operations with it, such as adding it, multiplying it, or dividing it by another series, differentiating or integrating it, exponentiating it, and so on. We will examine how to do so in this section. To begin with, suppose we have two series, given by

$$u = \sum_{k=0}^{N} u_k (x-a)^k + O(x-a)^{N+1} \tag{2.78}$$

[13] This generalization includes $O(n^2)$ algorithms for computation.

$$v = \sum_{k=0}^{N} v_k (x-a)^k + O(x-a)^{N+1} . \tag{2.79}$$

The scalar linear combination is defined to be

$$\alpha u + \beta v = \sum_{k=0}^{N} (\alpha u_k + \beta v_k)(x-a)^k + O(x-a)^{N+1} . \tag{2.80}$$

In other words, to add, subtract, and scalar-multiply series, we simply add, subtract, and scalar-multiply the corresponding coefficients. We examine the other operations in the following subsections.

2.6.1 Multiplication of Series and the Cauchy Convolution

The product $w = uv$ of two series u and v, as in Eqs. (2.78) and (2.79), can be written

$$w = uv = \sum_{k=0}^{N} w_k (x-a)^k + O(x-a)^{N+1} \tag{2.81}$$

as any other series. The problem, then, is to express the coefficients w_k in terms of the coefficients of u and v. The relationship in question is simply

$$w_k = \sum_{j=0}^{k} u_{k-j} v_j = \sum_{j=0}^{k} u_j v_{k-j} . \tag{2.82}$$

This is the Cauchy product formula or *convolution product*. It is crucial in what follows. It can be done faster than the direct sum formula above, by using the fast Fourier transform: See Henrici (1979b). See also Chap. 9 in this book.

Example 2.8. If we are given the series

$$u = 1 + 2(x-a) + 3(x-a)^2 + 4(x-a)^3 + 5(x-a)^4 + 6(x-a)^5 + O\left((x-a)^6\right)$$

and

$$v = 2 - 3(x-a) + 4(x-a)^2 - 5(x-a)^3 + 6(x-a)^4 - 7(x-a)^5 + O\left((x-a)^6\right) ,$$

then their product $w = uv$ has the series starting

$$uv = 2 + (x-a) + 4(x-a)^2 + 2(x-a)^3 + 6(x-a)^4 + 3(x-a)^5 + O\left((x-a)^6\right) .$$

These series were computed using the `series` command in MAPLE, which, among other things, implements the Cauchy convolution product. ◁

Is the computation of the Cauchy convolution numerically stable? If $N = 0$, so that we are multiplying constants, then (obviously) Cauchy convolution is numerically stable: $u_0v_0(1+\delta)$ can be interpreted as the exact product of $u_0(1+\delta/2)$ and $v_0(1+\delta)(1+\delta/2)^{-1} \approx v_0(1+\delta/2)$ by the IEEE standard. Cauchy convolution is *normwise* forward stable, as we will see, for any fixed N; but it is not componentwise stable for $N > 1$, as we will also see. But it is stable for $N = 1$.

Theorem 2.11. *Cauchy convolution is componentwise stable if $N = 1$.*

Proof. Suppose $N = 1$, so that $u = u_0 + u_1z + O(z^2)$, and $v = v_0 + v_1z + O(z^2)$. Then $uv = u_0v_0 + (u_0v_1 + u_1v_0)z + O(z^2)$ in exact arithmetic. If we are using floating-point arithmetic instead, then as we just saw, we may choose perturbations in u_0 and v_0 that allow us to interpret the first term as the exact product of perturbed u_0 and v_0. Suppose that we have done so, with $\hat{u}_0 = u_0(1+\delta_1)$ and $\hat{v}_0 = v_0(1+\delta_2)$, where δ_1 and δ_2 are such that $(1+\delta_1)(1+\delta_2) = (1+\delta_0)$ with $\hat{u}_0\hat{v}_0 = u_0v_0(1+\delta_0)$ and $|\delta_0| \le \mu_M$. As shown above, we may choose δ_1 and δ_2 so that each is also smaller than μ_M in magnitude. Now we wish to interpret the floating-point value $u_0 \otimes v_1 \oplus u_1 \otimes v_0$ as $\hat{u}_0\hat{v}_1 + \hat{u}_1\hat{v}_0$ with $\hat{u}_1 = u_1(1+\delta_3)$ and $\hat{v}_1 = v_1(1+\delta_4)$, with each of δ_3 and δ_4 small. We break the proof up into cases.

In the first case, suppose that $v_1 = 0$. Then we are multiplying u by a constant, and obviously each term in the product is the exact product of v_0 with a relatively minor change in u_0 and u_1: We have $\hat{u}_0 = u_0(1+\delta_0)$ and can take $\hat{u}_1 = u_1(1+\delta_3)$ directly, and leave $\hat{v}_0 = v_0$, and similarly if $u_1 = 0$.

Now suppose we are in the second case, where neither u_1 nor v_1 is zero. Then

$$\begin{aligned} u_0 \otimes v_1 \oplus u_1 \otimes v_0 &= (u_0v_1(1+\delta_5) + u_1v_0(1+\delta_6))(1+\delta_7) \\ &= (\hat{u}_0v_1(1+\delta_0)^{-1}(1+\delta_5) + u_1\hat{v}_0(1+\delta_6))(1+\delta_7) \\ &= \hat{u}_0\hat{v}_1 + \hat{u}_1\hat{v}_0, \end{aligned} \tag{2.83}$$

where $\hat{v}_1 = v_1(1+\delta_0)^{-1}(1+\delta_5)(1+\delta_7)$ is only three rounding errors different to v_1 and $\hat{u}_1 = u_1(1+\delta_6)(1+\delta_7)$ is only two rounding errors different to u_1. That is, the computed Cauchy convolution if $N = 1$ is the exact product of two series that differ only minutely in a relative sense to each multiplicand series. ♮

However, for $N = 2$, this kind of analysis cannot succeed.

Theorem 2.12. *For $N = 2$, Cauchy convolution can be componentwise unstable: That is, the computed product of two series of order $O(z^3)$ is not necessarily the exact product of any two nearby series, where "nearby" means each coefficient is relatively close to the original.*

Proof. Using a more systematic notation to help with the bookkeeping, suppose to the contrary that we may choose relative perturbations $\hat{u}_k = u_k(1+\delta_k^u)$ and $\hat{v}_k = v_k(1+\delta_k^v)$ in order to match the rounding errors in the computation, which we will denote by ε_k. We would then have

$$u_0v_0(1+\delta_0^u)(1+\delta_0^v) = u_0v_0(1+\varepsilon_0)$$

$$u_0 v_1(1+\delta_0^u)(1+\delta_1^v) + u_1 v_0(1+\delta_1^u)(1+\delta_0^v) = u_0 v_1(1+\varepsilon_1)(1+\varepsilon_3) \\ + u_1 v_0(1+\varepsilon_2)(1+\varepsilon_3),$$

where each $|\varepsilon_j| < \mu_M$, the unit roundoff. As we saw in the previous theorem, if we stop here, we may choose small δ's in order to satisfy these constraints: For $N = 1$, we may interpret the rounding errors as small relative backward errors. However, we need one more equation for $N = 2$:

$$u_2 v_0(1+\delta_2^u)(1+\delta_0^v) + u_1 v_1(1+\delta_1^u)(1+\delta_1^v) + u_0 v_2(1+\delta_0^u)(1+\delta_2^v) \\ = u_2 v_0(1+\varepsilon_4)(1+\varepsilon_6)(1+\varepsilon_8) + u_1 v_1(1+\varepsilon_5)(1+\varepsilon_6)(1+\varepsilon_8) + u_0 v_2(1+\varepsilon_7)(1+\varepsilon_8).$$

Because the u_k and v_k are independent variables, each monomial gives an equation for the unknown perturbations, so that we have

$$(1+\delta_0^u)(1+\delta_0^v) = (1+\varepsilon_0) \tag{2.84}$$
$$(1+\delta_0^u)(1+\delta_1^v) = (1+\varepsilon_1)(1+\varepsilon_3) \tag{2.85}$$
$$(1+\delta_1^u)(1+\delta_0^v) = (1+\varepsilon_2)(1+\varepsilon_3)\,, \tag{2.86}$$

and from the $O(z^2)$ term, we will only need

$$(1+\delta_1^u)(1+\delta_1^v) = (1+\varepsilon_5)(1+\varepsilon_6)(1+\varepsilon_8) \tag{2.87}$$

to arrive at a contradiction. Multiply Eqs. (2.85) and (2.86) together and divide by Eq. (2.84) to get

$$(1+\delta_1^u)(1+\delta_1^v) = \frac{(1+\varepsilon_1)(1+\varepsilon_3)^2(1+\varepsilon_2)}{(1+\varepsilon_0)}\,. \tag{2.88}$$

For this to hold simultaneously with (2.87) requires that the rounding errors ε_5, ε_6, and ε_8 be perfectly correlated with the earlier rounding errors ε_0, ε_1, ε_2, and ε_3. In general, this does not happen. Therefore, there is no possible set of perturbations δ_k^u and δ_k^v that allows rounding errors in Cauchy convolution for $N > 1$ to be interpreted as a small relative backward error. ♮

Remark 2.14. In the *forward* error sense, this computation also shows that the componentwise relative error may be infinite. Take an example where $u_0 v_2 + u_1 v_1 + u_2 v_0 = 0$ in exact arithmetic. Then the rounding errors, which will be proportional to $|u_0 v_2| + |u_1 v_1| + |u_2 v_0|$, will be infinitely large in comparison with the reference result of 0. ◁

However, there is a forward accuracy result for all N in the normwise sense, as follows. If $c_n = \sum_{j=0}^{n} u_j v_{n-j}$, then Eq. (3.5) in (Higham 2002 section 3.1), which gives us a general result on the forward accuracy of inner products, tells us that

$$|\hat{c}_n - c_n| \leq \sqrt{2}\gamma_{n+1} \sum_{j=0}^{n} |u_j||v_{n-j}|\,. \tag{2.89}$$

If all terms u_j and v_j are positive, this is a decent relative accuracy (and the constant in front can be improved with minor modifications of how the sum is done and in which order). If, however, c_n is very small while some u_j and v_j are large in magnitude, then there must be cancellation, and the error bound will then be large to reflect this.

Remark 2.15. This difficulty may be mitigated by performing this recurrence relation in higher precision, or by using certain compensated summation techniques as described in Higham (2002). However, the inaccuracy is often of little consequence in computation with the resulting series, anyway, even if the recurrence relation is performed in a naive way. The reason is simply that the errors grow at worst in c_k like $O((k+1)\|\mathbf{u}\|\|\mathbf{v}\|\mu_M)$, and thus for the low-order terms, the error is small in any case; and the high-order terms are used only together with high powers of $(z-a)$, which is presumed small. Thus, the total error in the *computed sum* $\sum_{j=0}^{N} c_j(z-a)^j$ will be small enough: The terms with larger errors will not contribute much to the total sum. ◁

Finally, we leave aside the question of whether the Cauchy convolution is well-conditioned, which we will take it up in the exercises in Chap. 3.

2.6.2 Division of Series

Let us now consider the case of division. If we consider

$$r = \frac{u}{v} = \sum_{k=0}^{N} r_k (x-a)^k + O(x-a)^{N+1}\,, \tag{2.90}$$

then we must have $u = rv$, so that

$$u_k = \sum_{j=0}^{k} r_j v_{k-j} = r_k v_0 + \sum_{j=0}^{k-1} r_j v_{k-j}, \tag{2.91}$$

and we see that for r_k to be defined we must have $v_0 \neq 0$. Then

$$r_k = \frac{1}{v_0}\left(u_k - \sum_{j=0}^{k-1} r_j v_{k-j}\right) \tag{2.92}$$

and the base of the recurrence is (if $v_0 \neq 0$)

$$r_0 = \frac{u_0}{v_0}. \tag{2.93}$$

However, if $v_0 = 0$, then no series for r exists, unless perhaps also $u_0 = 0$ and we may cancel a factor in $^u/_v$.

Example 2.9. With the same u and v that we used in Example 2.8, we find that

$$\frac{u}{v} = \frac{1}{2} + \frac{7}{4}(x-a) + \frac{25}{8}(x-a)^2 + \frac{71}{16}(x-a)^3 + \frac{185}{32}(x-a)^4 + O\left((x-a)^5\right),$$

while

$$\frac{v}{u} = 2 - 7(x-a) + 12(x-a)^2 - 16(x-a)^3 + 20(x-a)^4 + O\left((x-a)^5\right).$$

Again these series were computed in MAPLE, which knows how to do series algebra including division, and cancels common factors in order to avoid division by zero wherever possible. MAPLE also knows how to work with several generalizations of Taylor series, including *Laurent* series, which allow negative integer powers of $(x-a)$, and *Puiseux* series, which allow fractional powers. The algebra of these is a straightforward extension of that for Taylor series. We will occasionally have need for these generalizations. ◁

2.6.3 Differentiation and Integration

Differentiation of power series is very straightforward. If we are given a series

$$u = \sum_{k=0}^{N} u_k (x-a)^k + O(x-a)^{N+1},$$

then it is easy to see that its derivative is

$$\begin{aligned}\frac{du}{dx} &= \sum_{k=0}^{N} k u_k (x-a)^{k-1} + O(x-a)^N \\ &= \sum_{k=0}^{N-1} (k+1) u_{k+1} (x-a)^k + O(x-a)^N.\end{aligned}$$

Moreover, its integral is also directly seen to be

$$\begin{aligned}\int_a^x u(\xi)d\xi &= u_0(x-a) + u_1\frac{(x-a)^2}{2} + \ldots \\ &= \sum_{k=0}^{N} \frac{u_k}{k+1}(x-a)^{k+1} + O(x-a)^{N+2}.\end{aligned}$$

These two simple operations will be applied to many problems in this book.

2.6.4 *The Algebra of Series*

The rules we have examined already give us the series for all polynomials and rational functions. Using these rules, we see how to square a series,

$$\begin{aligned} x^2 &= \left(a + (x-a) + O(x-a)^{N+1}\right)^2 \\ &= a^2 + 2a(x-a) + (x-a)^2 + O(x-a)^{N+1}, \end{aligned}$$

or to take higher powers. Moreover, we see that

$$\begin{aligned} \frac{1}{x} &= \frac{1 + O(x-a)^{N+1}}{a + (x-a) + O(x-a)^{N+1}} \\ &= \sum_{k=0}^{N} \frac{(-1)^k}{a^{k+1}}(x-a)^k + O(x-a)^{N+1}. \end{aligned}$$

As a result, we can also find the series for $\ln(x)$ by using the integration rule. Observe that

$$\ln(x) = \int_1^x \frac{dt}{t} = \int_1^a \frac{dt}{t} + \int_a^x \frac{dt}{t}.$$

From this, we obtain

$$\begin{aligned} \ln(x) &= \ln(a) + \int_a^x \left(\sum_{k=0}^{N} \frac{(-1)^k}{a^{k+1}}(x-a)^k + O(x-a)^{N+1} \right) dx \\ &= \ln(a) + \sum_{k=0}^{N} \frac{(-1)^k}{(k+1)a^{k+1}}(x-a)^{k+1} + O(x-a)^{N+2}. \end{aligned}$$

Algebraically, the set of truncated power series (TPS) of order N forms an *integral domain*: The sum, difference, and product of TPS are TPS, but there are zero divisors, and not every element has a reciprocal—indeed, each element with a zero leading coefficient fails to have a TPS reciprocal. If we allow negative integer powers of $(x-a)$, then we have *truncated Laurent series*, which are indeed useful.

2.6.5 *The Exponential of a Series*

To find the series for e^x, we may introduce series reversion (see Exercise 2.24) or look at the differential equation

$$\frac{dy}{dx} = y, \qquad y(a) = e^a, \tag{2.94}$$

which is, of course, satisfied by e^x. Now, let y be given by

$$y = \sum_{k=0}^{N} y_k (x-a)^k + O(x-a)^{N+1} . \tag{2.95}$$

We have $y_0 = e^a$ and, by differentiation, we also have

$$\sum_{k=0}^{N} k y_k (x-a)^{k-1} + O(x-a)^N = \sum_{k=0}^{N} y_k (x-a)^k + O(x-a)^{N+1},$$

or, by rearranging the summation indices,

$$\sum_{k=0}^{N-1} (k+1) y_{k+1} (x-a)^k + O(x-a)^N = \sum_{k=0}^{N} y_k (x-a)^k + O(x-a)^{N+1} . \tag{2.96}$$

By the uniqueness of power series, we can identify the coefficients of corresponding powers, thereby obtaining the relation

$$(k+1) y_{k+1} = y_k \,, \qquad k = 0, 1, 2, 3, \ldots, N-1 . \tag{2.97}$$

Using our initial condition and this recursive relation, we find that

$$\begin{aligned} y_1 &= y_0 = e^a \\ 2y_2 &= y_1 = e^a \\ 3y_3 &= y_2 = \frac{1}{2} e^a \\ 4y^4 &= y_3 = \frac{1}{6} e^a \,, \end{aligned}$$

and so on, so that the series for the exponential itself is

$$e^x = \sum_{k=0}^{N} \frac{e^a}{k!} (x-a)^k + O(x-a)^{N+1} . \tag{2.98}$$

However, that was really too easy for such a powerful trick. How about $y = e^u$, where

$$u = \sum_{k=0}^{N} u_k (x-a)^k + O(x-a)^{N+1}$$

instead? It still works! Let y be as in Eq. (2.95). Then, because

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = y \frac{du}{dx} \,,$$

we find that

$$\sum_{k=0}^{N-1}(k+1)y_{k+1}(x-a)^k+O(x-a)^N = \\ \left(\sum_{k=0}^{N}y_k(x-a)^k+O(x-a)^{N+1}\right)\left(\sum_{k=0}^{N}u_k(x-a)^k+O(x-a)^{N+1}\right).$$

Now, applying the Cauchy convolution rule to the right-hand side gives us

$$\sum_{k=0}^{N-1}(k+1)y_{k+1}(x-a)^k+O(x-a)^N=\sum_{k=0}^{N}c_k(x-a)^k+O(x-a)^{N+1}, \tag{2.99}$$

where

$$c_k=\sum_{j=0}^{k}y_ju_{k-j}.$$

By the same method, we thus find the relation

$$(k+1)y_{k+1}=\sum_{j=0}^{k}y_ju_{k-j}. \tag{2.100}$$

Also, it is obviously the case that the recurrence starts with $y_0=e^{u_0}$. This recurrence relation allows us to compute the exponential of any series. We will later solve differential equations with this technique.

Example 2.10. If $u(x)$ has the following series,

$$u=\frac{\sqrt{2}}{2}+\frac{\sqrt{2}}{2}\left(x-\frac{\pi}{4}\right)-\frac{\sqrt{2}}{4}\left(x-\frac{\pi}{4}\right)^2-\frac{\sqrt{2}}{12}\left(x-\frac{\pi}{4}\right)^3 \\ +\frac{\sqrt{2}}{48}\left(x-\frac{\pi}{4}\right)^4+O\left(\left(x-\frac{\pi}{4}\right)^5\right),$$

then $\exp(u)$ has the series beginning

$$e^u=e^{\sqrt{2}/2}+\frac{1}{2}e^{\sqrt{2}/2}\sqrt{2}\left(x-\frac{\pi}{4}\right)+e^{\sqrt{2}/2}\left(\frac{1}{4}-\frac{\sqrt{2}}{4}\right)\left(x-\frac{\pi}{4}\right)^2 \\ -e^{\sqrt{2}/2}\left(\frac{1}{4}+\frac{\sqrt{2}}{24}\right)\left(x-\frac{\pi}{4}\right)^3-e^{\sqrt{2}/2}\left(\frac{1}{96}+\frac{\sqrt{2}}{24}\right)\left(x-\frac{\pi}{4}\right)^4+O\left(\left(x-\frac{\pi}{4}\right)^5\right).$$

Again, MAPLE was used with its series command, which implements the algorithms discussed here. Specifically, once the series for u was defined, the command

```
series( exp(u), u=Pi/4 )
```

generated the above result. ◁

You may notice that *convergence* has not entered the discussion. Since we work only with truncated, finite power series, this is not a serious omission. Truncation error formulæ, on the other hand, are very useful, even if the series don't converge. You may be familiar with the Lagrange form of the remainder (that is, truncation error) for *real* Taylor series:

$$f(x) = f(a) + f'(a)(x-a) + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n + R_{n+1}(x;a), \qquad (2.101)$$

where

$$R_{n+1}(x;a) = \frac{f^{(n+1)}(a+\theta x)}{(n+1)!}(x-a)^{n+1}. \qquad (2.102)$$

Here θ is some number between 0 and 1, which we don't know exactly. Knowledge of bounds on the $(n+1)$st derivative allows us to estimate how much accuracy we have in our real Taylor series when we truncate at n terms. This formula doesn't work over the complex numbers, however: Instead, we have (replacing x by z everywhere above)

$$R_{n+1}(z;a) = \frac{(z-a)^{n+1}}{2\pi i}\oint_C \frac{f(\zeta)}{(\zeta-a)^{n+1}(\zeta-z)}\,d\zeta. \qquad (2.103)$$

Here C is a contour enclosing a and z. This integral can be interpreted as an "average value" of the $(n+1)$st derivative; in the complex plane, however, this average value is not always attained at some point $a+\theta z$. We will see a generalization of this formula to the case of interpolation error in Eq. (8.40) in Chap. 8.

We may also need to worry about whether the computed series is *well-conditioned* with respect to the data. Again this is taken up in Chap. 3.

2.7 A Partial Fraction Decomposition Algorithm Using Local Taylor Series

We return to the problem of computing the partial fraction decomposition of $p(z)/q(z)$, where $q(z)$ has been completely factored down to distinct linear (complex) factors $(z-\tau_k)^{s_k}$ for $0 \le k \le n$. We will need, first, the local Taylor series of p for each τ_k:

$$p(z) = p_{k,0} + p_{k,1}(z-\tau_k) + \cdots + p_{k,s_k-1}(z-\tau_k)^{s_k-1} + O(z-\tau_k)^{s_k}. \qquad (2.104)$$

In other words, we need to reexpress $p(z)$ in each of the n local (i.e., shifted) monomial bases 1, $(z-\tau_k)$, $(z-\tau_k)^2$, …, $(z-\tau_k)^d$, except that we only need the first s_k coefficients in each case. This can be done using synthetic division, as discussed earlier. Assume that this has been done. Then, if

$$q(z) = \prod_{k=0}^{n} (z-\tau_k)^{s_k} , \tag{2.105}$$

then the rational function we wish to decompose into partial fractions, $p(z)/q(z)$, can be written as follows. We choose τ_0 as being special, for the moment, and let $w_0(z) = \prod_{k=1}^{n} (z-\tau_k)^{-s_k}$, which is analytic at τ_0 because all the τ_k are distinct by hypothesis (confluency is explicitly known). Thus, we can compute *its* local Taylor series by the methods of the previous section. In general, that is, not just for $k=0$, let

$$w_i(t) = \prod_{\substack{k=0\\k\neq i}}^{n} (t-\tau_k)^{-s_k} .$$

Then, we obtain the local Taylor series

$$w_i(t) = w_{i,0} + w_{i,1}(t-\tau_i) + \cdots = \sum_{\ell\geq 0} w_{i,\ell}(t-\tau_i)^{\ell} .$$

Then, observe that

$$\frac{p(z)}{q(z)} = \frac{p(z)}{\prod\limits_{k=0}^{n}(z-\tau_k)^{s_k}} = \frac{p(z)}{(z-\tau_i)^{s_i}\prod\limits_{\substack{k=0\\k\neq i}}^{n}(z-\tau_k)^{s_k}} = \frac{p(z)w_i(z)}{(z-\tau_i)^{s_i}} . \tag{2.106}$$

Now, this is exactly the form required for a partial fraction decomposition. As a result, the partial fraction decomposition we want may be obtained by Cauchy convolution with the local series for $p(z)$.

There are many ways to do this. The following is one method, and it has been implemented in MATLAB.[14] Begin by taking logarithms of $w_i(z)$:

$$\ln w_i(z) = \sum_{\substack{k=0\\k\neq i}}^{n} -s_k \ln(z-\tau_k) + \text{complex piecewise constants} . \tag{2.107}$$

When we take derivatives with respect to z, all the piecewise constants (multiples of $2\pi i$) disappear:

$$\frac{w_i'(z)}{w_i(z)} = \sum_{\substack{k=0\\k\neq i}} \frac{-s_k}{z-\tau_k} . \tag{2.108}$$

[14] The program is discussed in Chap. 8.

Note that

$$\frac{1}{z-\tau_k} = \frac{1}{\tau_i - \tau_k + z - \tau_i}$$

and so the summands in the right-hand sum of Eq. (2.108) can be expressed as follows:

$$\frac{-s_k}{z-\tau_k} = \frac{s_k}{\tau_k - \tau_i} \frac{1}{1 - \frac{z-\tau_i}{\tau_k-\tau_i}} = \frac{s_k}{\tau_k - \tau_i} \sum_{\ell \ge 0} \left(\frac{z-\tau_i}{\tau_k - \tau_i} \right)^{\ell} .$$

Therefore,

$$\frac{w_i'(z)}{w_i(z)} = \sum_{\substack{k=0\\k\neq i}}^{n} \sum_{\ell\ge 0} \frac{s_k}{(\tau_k - \tau_i)^{\ell+1}} (z-\tau_i)^{\ell} = \sum_{\ell \ge 0} \left(\sum_{\substack{k=0\\k\neq i}}^{n} \frac{s_k}{(\tau_k - \tau_i)^{\ell+1}} \right) (z-\tau_i)^{\ell} . \quad (2.109)$$

If we define $u_{i,\ell}$ as follows,

$$u_{i,\ell} = \sum_{\substack{k=0\\k\neq i}}^{n} \frac{s_k}{(\tau_k - \tau_i)^{\ell+1}} , \quad (2.110)$$

then Eq. (2.109) becomes

$$\frac{w_i'(z)}{w_i(z)} = \sum_{\ell \ge 0} u_k (z-\tau_i)^{\ell} . \quad (2.111)$$

To simplify things a bit more, let $v_{i,m} = {}^{w_{i,m}}/_{w_{i,0}}$ (so $v_{i,0} = 1$) and note that

$$w_{i,0} = \prod_{\substack{k=0\\k\neq i}}^{n} (\tau_i - \tau_k)^{-s_k} .$$

Now, it follows that

$$w_i(z) = \sum_{m\ge 0} w_{i,m} (z-\tau_i)^m = w_{i,0} \sum_{m \ge 0} v_{i,m} (z-\tau_i)^m .$$

Taking derivatives (using $'$ to denote d/dz), we have

$$\begin{aligned} w_i'(z) = w_{i,0} \sum_{m\ge 0} m v_{i,m} (z-\tau_i)^{m-1} &= w_{i,0} \sum_{m\ge 1} m v_{i,m} (z-\tau_i)^{m-1} \\ &= w_{i,0} \sum_{m\ge 0} (m+1) v_{i,m+1} (z-\tau_i)^m . \end{aligned}$$

Putting these in (2.111) and rearranging in order to more easily compare coefficients, we get the following:

$$\begin{aligned}\sum_{m\geq 0}(m+1)v_{i,m+1}(z-\tau_i)^m &= \left(\sum_{m\geq 0}v_{i,m}(z-\tau_i)^m\right)\left(\sum_{\ell\geq 0}u_{i,\ell}(z-\tau_i)^\ell\right)\\ &= \sum_{m\geq 0}c_m(z-\tau_i)^m,\end{aligned}$$

where

$$c_m = \sum_{\ell=0}^{m} v_{i,m-\ell}u_{i,\ell}$$

is Cauchy's convolution formula. Equating coefficients gives (remember $v_{i,0}=1$)

$$\begin{aligned}v_{i,m+1} &= \frac{1}{m+1}c_m\\ &= \frac{1}{m+1}\sum_{\ell=0}^{m} v_{i,m-\ell}u_{i,\ell}\,. \end{aligned}\tag{2.112}$$

Recall that (2.110) defines $u_{i,\ell}$. The recurrence relation (2.112) is the heart of the local Taylor series algorithm for partial fractions. Once we have the $v_{i,k}$, then we have the desired $\beta_{i,j}$.

Algorithm 2.4 Partial fraction decomposition by local Taylor series

```
Require: A positive integer $n$, a list of positive integers $s_k$, a list of $n$ distinct zeros $\tau_k$ of the
    denominator $q(z)=\prod_{k=0}^{n}(z-\tau_k)^{s_k}$, and the $n+1$ lists of local series coefficients $p_{k,j}$, $0\leq j\leq$
    $s_k-1$ of $p(z)$.
  for $i$=0:$n$ do
    for $j$=$i+1$:$n$ do
      $\Delta\tau_{i,j}=\tau_i-\tau_j$
    end for
    $v_{i,0}=1$
    for $m$=0:$s_i-1$ do
      $u_{i,m}=\sum_{k=0, k\neq i}^{n}\Delta\tau_{i,k}^{-m-1}$
      $v_{i,m+1}=\frac{1}{m+1}\sum_{k=0}^{m}u_{i,k}v_{i,m-k}$
    end for
    $\beta_i=\prod_{k=0, k\neq i}^{n}(\tau_i-\tau_k)^{-s_k}$
    for $m$=1:$s_i$ do
      $w_{i,m}=\beta_i v_{i,s_i-m}$
    end for
    for $m$=1:$s_i$ do
      $\alpha_{i,m}=\sum_{k=0}^{m}p_{i,k}w_{i,m-k}$
    end for
  end for
  return The coefficients $\alpha_{k,j}$ in the partial fraction decomposition
```

$$\frac{p(z)}{\prod_{k=0}^{n}(z-\tau_k)^{s_k}} = \sum_{k=0}^{n}\sum_{j=0}^{s_k-1}\frac{\alpha_{k,j}}{(z-\tau_k)^{j+1}} \tag{2.113}$$

Algorithm 2.4 has been implemented in the MATLAB program `genbarywts` and in the MAPLE program `BHIP`, for the case where the numerator is just 1. You will be asked to show in the exercises that this algorithm costs $O(d^2)$ flops, when proper care is taken to avoid redundancy. In the case when all $s_k = 1$, the algorithm reduces merely to the computation of β_i for $1 \le i \le n$. In that case, the computation was proved to be numerically stable by Higham (2004). If any $s_k > 1$, then the algorithm is *not* backward stable, in the case when the nodes τ_k are symmetric about zero (for example) and some of the partial fraction decomposition coefficients $\alpha_{i,j}$ are exactly zero. However, the algorithm is stable *enough* for many purposes.

Example 2.11. Suppose the nodes τ_k are the Chebyshev–Lobatto nodes $\tau_k = \cos(\pi k/n)$ for $0 \le k \le n$. Take first the case $n = 5$, and execute this code:

```
tau = cos( pi*k/n );
[w,D] = genbarywts( tau, 1 )
```

It returns the values

$$w = 1.6000, -3.2000, 3.2000, -3.2000, 3.2000, -1.6000\,.$$

By comparison with MAPLE, these answers are correct up to $O(\mu_M)$. When $n = 50$, the numbers are larger, $O(10^{13})$, but still have relative forward error only about 2×10^{-14}.

When we make each node have confluency $s = 2$, the situation changes a bit, but not much, for $n = 5$:

$$\alpha = \begin{bmatrix} -43.520 & 2.5600 \\ 23.978 & 10.240 \\ 3.4984 & 10.240 \\ -3.4984 & 10.240 \\ -23.978 & 10.240 \\ 43.520 & 2.5600 \end{bmatrix},$$

and again the forward error is $O(\mu_M)$. For $n = 50$, this is again true. We detect no instability in this example. For higher confluencies, we expect more trouble. $\triangleleft$

2.8 Asymptotic Series in Scientific Computation

Niels Henrik Abel (1802–1829) wrote

> The divergent series are the invention of the devil, and it is a shame to base on them any demonstration whatsoever. By using them, one may draw any conclusion [s]he pleases and that is why these series have produced so many fallacies and paradoxes [. . .]. (cited in Hoffman 1998 p. 218)

Nowadays, the attitude is different, and closer to what Heaviside meant when he said

The series is divergent; therefore we may be able to do something with it. (cited in Hardy 1949)

In fact, asymptotic series will be used a lot in this book, and we will often not care too much whether they converge. This is because, in many contexts, the first few terms contain all the numerical information one needs; there's no need to ponder on what happens in the tail end of the series.

The key to understanding asymptotic series is to realize that there are *two* limits to choose from, with a series. Suppose we have, for example,

$$f(z) = \sum_{k=0}^{N} \frac{f^{(k)}(a)}{k!}(z-a)^k \quad + \quad R_N(z)(z-a)^{N+1}\,, \tag{2.114}$$

as the usual truncated Taylor series for $f(z)$ near $z = a$. We can take the first limit, $N \to \infty$, to get the familiar mathematical object of the *infinite series*. This only makes sense if the limit exists. (There is some freedom to alter the definition of limit that we use in this case; we do not pursue this here.) If that limit exists, we say the series is *convergent*. However, there is another limit to be considered here, which often leads to very useful results. Namely, do *not* let $N \to \infty$, but rather keep it fixed (perhaps even at $N = 1$ or $N = 2$). Instead, consider the limit as $z \to a$. Even if the series is divergent in the first sense, this second limit often gives enormously useful information, typically because $R_N(z)$ (as it is written above) is well behaved near $z = a$, and so the term $(z-a)^{N+1}$ ensures that the remainder term vanishes more quickly than do the terms that are kept. The rest of this section explores that simple idea.

We often want to consider the behavior of a function $y(x)$ in the presence of some perturbations. Then, instead of studying the original function $y(x)$, we study the asymptotic behavior of a two-parameter function $y(x,\varepsilon)$, where ε is considered "small." An asymptotic expansion for the function $y(x,\varepsilon)$ has the form

$$y(x,\varepsilon) = y_0(x)\phi_0(\varepsilon) + y_1(x)\phi_1(\varepsilon) + y_2(x)\phi_2(\varepsilon) + \ldots = \sum_{k=0}^{\infty} y_k(x)\phi_k(\varepsilon)\,, \tag{2.115}$$

where $\phi_k(\varepsilon)$ are referred to as *gauge functions*; that is, they are a sequence of functions $\{\phi_k(\varepsilon)\}$ such that, for all k,

$$\lim_{\varepsilon\to 0} \frac{\phi_{k+1}(\varepsilon)}{\phi_k(\varepsilon)} = 0\,.$$

The type of gauge function we will use the most often is the power of the perturbation ε, namely, $\phi_k(\varepsilon) = \varepsilon^k$, in which case we simply have a formal power series:

$$y(x,\varepsilon) = y_0(x) + y_1(x)\varepsilon + y_2(x)\varepsilon^2 + \ldots = \sum_{k=0}^{\infty} y_k(x)\varepsilon^k.$$

We then have to solve for the $y_k(x)$, $k = 0, 1, \dots, N$. To find the first coefficient $y_0(x)$, divide Eq. (2.115) by $\phi_0(\varepsilon)$, and then take the limit as $\varepsilon \to 0$:

$$\frac{y(x,\varepsilon)}{\phi_0(\varepsilon)} = y_0(x) + \frac{1}{\phi_0(\varepsilon)} \sum_{k=1}^{\infty} y_k(x)\phi_k(\varepsilon)$$
$$\lim_{\varepsilon \to 0} \frac{y(x,\varepsilon)}{\phi_0(\varepsilon)} = y_0(x)\,.$$

All the higher-order terms vanish since $\phi_k(\varepsilon)$ is a gauge function. This gives us $y_0(x)$. Now, subtract $y_0(x)\phi_0(\varepsilon)$ from both sides in Eq. (2.115); we then divide both sides by $\phi_1(\varepsilon)$ and take the limit as $\varepsilon \to 0$:

$$\frac{y(x,\varepsilon) - y_0(x)\phi_0(\varepsilon)}{\phi_1(\varepsilon)} = y_1(x) + \frac{1}{\phi_1(\varepsilon)} \sum_{k=2}^{\infty} y_k(x)\phi_k(\varepsilon),$$

so

$$\lim_{\varepsilon \to 0} \frac{y(x,\varepsilon) - y_0(x)\phi_0(\varepsilon)}{\phi_1(\varepsilon)} = y_1(x)\,. \tag{2.116}$$

As we see, we will in general have

$$y_k(x) = \lim_{\varepsilon \to 0} \frac{1}{\phi_k(\varepsilon)} \left(y(x,\varepsilon) - \sum_{\ell=0}^{k-1} y_\ell(x)\phi_\ell(\varepsilon) \right)\,. \tag{2.117}$$

Convergence of a series is all about the tail, which requires an infinite amount of work. What we want instead is gauge functions that go to zero very fast; in other words, the speed at which they go to zero is asymptotically faster from one term to the next.

Example 2.12. Consider the (convergent) integral and the (divergent) asymptotic series

$$\int_0^\infty \frac{e^{-t}}{1+xt} dt = \sum_{k=0}^{n} (-1)^k k! x^k + O(x^{n+1})\,.$$

One can discover that series by replacing $1/(1+xt)$ with the finite sum $1 - xt + x^2t^2 + \cdots (xt)^n + \frac{(-xt)^{n+1}}{(1+xt)}$, giving

$$\int_0^\infty \frac{e^{-t}}{1+xt} dt = \sum_{k=0}^{n} (-1)^k x^k \int_0^\infty t^k e^{-t}\, dt + (-1)^{n+1} x^{n+1} \int_0^\infty \frac{t^{k+1} e^{-t}}{1+xt}\, dt\,.$$

This provides a perfectly definite meaning to each of the entries in the asymptotic series. Notice that the series diverges for any $x \neq 0$, if we take the limit as $n \to \infty$.

Nonetheless, taking (say) $n = 5$ allows us to evaluate the integral perfectly accurately for small enough x, say $x = 0.03$: summing the six terms gives 0.9716545240, whereas the exact value (found by the methods of Chap. 10) begins 0.9716549596, which differs by about $5 \cdot 10^{-7}$. $\triangleleft$

Remark 2.16. In the previous example, we have used a *divergent* series to give us a good approximation to the correct answer. Heaviside was right, and asymptotic series are extremely useful in numerical analysis. The reason this works is that it is the limit as $x \to 0$ that is dominating here: If we had wanted an accurate answer for $x = 10$, we would have been out of luck. We will often be concerned with the asymptotics of the error as the *average mesh width* (call it h) goes to zero, for example, and methods will be designed to be accurate in that limit. $\triangleleft$

2.9 Chebyshev Series and Chebfun

This generalized review chapter is not the right place to begin explaining the underlying methods of the Chebfun package. Here we mention only that the package does not use Taylor series, but rather *interpolation at Chebyshev points* (we expand on this in Chap. 8), which is closely related to *Chebyshev series*: One can convert back and forth using the FFT, in a stable and efficient fashion (see Chap. 9). What, then, are Chebyshev series? Just as with Taylor series, one can find convergent series for elementary functions, but where now the gauge functions are not shifted monomials but rather Chebyshev polynomials[15]; for example,

$$e^x = J_0(i)T_0(x) + 2\sum_{k=1}^{\infty} i^k J_k(-i)T_k(x)\,. \tag{2.118}$$

The coefficients are evaluations of the Bessel functions $J_k(z)$ at particular arguments (complex arguments, as it happens, although the results are real). This series is not expressed in powers of x or of $x - a$, but rather in higher- and higher-degree Chebyshev polynomials. One could do this on other intervals by the linear transformation $x = {}^{2(t-a)}/_{(b-a)} - 1$, so $a \le t \le b$; the coefficients would be different, of course. When one has evaluated $J_0(i) \approx 1.266\ldots$ and $J_k(-i)$ for several k, this series (and series like this) can provide a quite effective method for evaluating the function under consideration. See, for example, Boyd (2002) for applications to computing zeros of functions. For example, taking the first 15 terms here gives us

$$\begin{aligned} e^x = {} & 1.266065877752\,01 T_0(x) + 1.13031820798497\,T_1(x) \\ & + 0.271495339534077\,T_2(x) + 0.0443368498486638\,T_3(x) \\ & + 0.00547424044209373\,T_4(x) + 0.000542926311913944\,T_5(x) \\ & + 0.0000449773229542951\,T_6(x) + 0.00000319843646240199\,T_7(x) \end{aligned}$$

[15] See Rivlin (1990 Chapter 3).

$$
\begin{aligned}
&+0.00000199212480667280\,T_8(x)+0.000000110367717255173\,T_9(x)\\
&+0.00000000550589607967375\,T_{10}(x)+2.49795661698498\times10^{-11}\,T_{11}(x)\\
&+1.03915223067857\times10^{-12}\,T_{12}(x)+3.99126335641440\times10^{-14}\,T_{13}(x)\\
&+1.42375801082566\times10^{-15}\,T_{14}(x)
\end{aligned}
\tag{2.119}
$$

and this approximation has the relative error (on the interval $-1 \le x \le 1$) shown in Fig. 2.8. We will pursue this concept further in later chapters. For now, note that

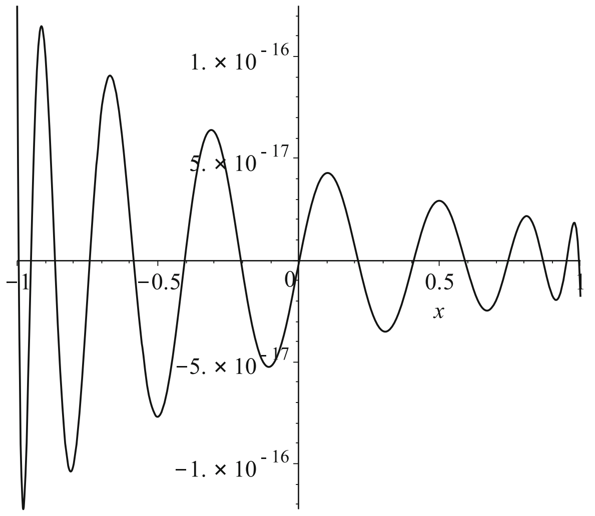

Fig. 2.8 The relative error $S\cdot\exp(-x)-1$ in the truncated Chebyshev series (2.119), computed in high precision in MAPLE

$|T_k(x)| \le 1$, and so the size of the coefficients tells us directly how much each term contributes (at most) to the sum.

In Chebfun itself, this series can be computed as follows, assuming the Chebfun package has been installed.

```
x = chebfun('x',[-1,1]);
y = exp(x);
co = chebpoly(y);
format long e
co(end:-1:1)'
% ans =
%
%    1.266065877752008e+000
%    1.130318207984970e+000
%    2.714953395340767e-001
%    4.433684984866388e-002
%    5.474240442093829e-003
%    5.429263119140007e-004
%    4.497732295430233e-005
%     3.198436462443460e-006
%     1.992124806757106e-007
%     1.103677179109604e-008
%     5.505895456820691e-010
%     2.497954620928056e-011
%     1.039121170062377e-012
%     4.003147020219850e-014
%     1.395708945243054e-015
%
t = linspace(-1,1,3011);
reler = exp(-t).*y(t)-1;
plot( t, reler, 'k-' );
set(gca, 'fontsize', 16);
axis([-1,1,-1.5E-15,1.5E-15]);
set(gca, 'YTick', -1.5E-15:5E-16:1.5E-15);
```

As you can see, the numbers do not quite match (although the largest three do): The series at the beginning of this section was computed using MAPLE in 60 digits of precision, and then the coefficients were rounded to 15 digits. They are different from the Chebfun series coefficients printed above, but not in any important way because the differences in the smallest coefficient (which are the greatest, relatively speaking) matter the least to the sum. The source of the difference is not a numerical error, but rather a difference in type of approximation. We will return to this later, but for now, observe that Chebfun is doing what it is supposed to—something similar to the Chebyshev series above, but not exactly the same thing. As we see in Fig. 2.9, it produces a perfectly acceptable (and even somewhat similar, ignoring the discrete-level effect of being so close to unit roundoff) error curve.

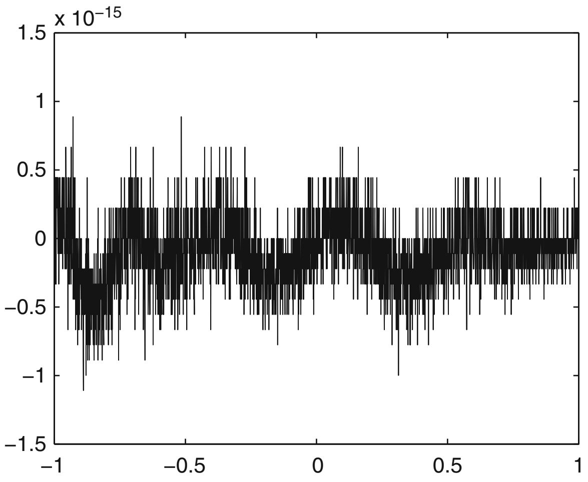

Fig. 2.9 The relative error $y \cdot \exp(-x) - 1$ in the `chebfun` for $y = \exp(x)$, as computed in normal precision in MATLAB

2.10 Notes and References

This chapter was originally intended for self-study, although the importance of the material suggests that it should be more formally included in any course using this book. A more elementary introduction to the theory of univariate polynomials can be found in Barbeau (2003). A more detailed introduction to the computation of Taylor series can be found in Henrici (1974). For a statement and discussion of the fundamental theorem of algebra, see, for instance, Levinson and Redheffer (1970).

Variations of Theorem 2.9 can be found throughout the literature, so many that Stetter (1999) says that it is misleading to cite any; the paper (Rezvani and Corless 2005) points out that it is really just an application of Hölder's inequality.

Algorithm 2.1, our version of the synthetic division algorithm, is an adaptation of Algorithm 5.2 in Higham (2002 p. 96). We modify that algorithm here to return the local Taylor coefficients, that is, $p^{(k)}(a)/k!$ instead of multiplying by factorials as done there to return values of the derivatives $p^{(k)}(a)$.

The special case $\tau_k = -k$ or $\tau_k = k$ for $0 \le k \le n-1$ of Newton polynomials is useful in combinatorics and is sometimes called the Pochhammer basis. We have already seen this, but called it $z^{\underline{k}}$, z to the k falling.

For a thorough treatment of Chebyshev polynomials, see Rivlin (1990). See Salzer (1972) for more discussion of useful properties of the Chebyshev–Lobatto points η_k. For a discussion of Chebfun and Chebyshev polynomials, see Battles and Trefethen (2004) and `http://www2.maths.ox.ac.uk/chebfun/`.

Some other orthogonal bases are discussed in the venerable book (Abramowitz and Stegun 1972). That book has been substantially revised to become the Digital Library of Mathematical Functions from the National Institute of Standards and Technology (`http://dlmf.nist.gov/`). A similar INRIA project, the Dynamic Dictionary of Mathematical Functions, may be found at `http://ddmf.msr-inria.inria.fr/1.6/ddmf`. More details on many orthogonal polynomials can be found in Andrews et al. (1999), and some important algorithms in Wilf (1962), available for free for educational purposes from `http://www.math.upenn.edu/~wilf/website/Mathematics_for_the_Physical_Sciences.html`. A discussion of MAPLE's methods for othogonal series can be found in Rebillard (1997) and Ronveaux and Rebillard (2002).

Faster methods of partial fraction decomposition than the one advocated here are certainly available: Kung and Tong (1977) and Chin (1977) use FFT methods, which we believe are unstable because they implicitly convert to the monomial basis; the divided-difference algorithm of Schneider and Werner (1991), which, although not asymptotically fast, is twice as fast as the algorithm given here, has an unhelpful dependence on node ordering and again can produce $\beta_{i,j}$ that do not accurately reproduce 1. Their method is much more stable than methods that convert to the monomial basis, however.

Problems

Theory and Practice

2.1. Prove Theorem 2.1 on page 44.

2.2. Prove Theorem 2.2 on page 45.

2.3. Show that the roots of $z^n - 1 = 0$ are $z_k = \exp({}^{2\pi ik}/_{n})$.

2.4. Show that the roots of $T_n(z)$ are

$$\xi_k = \cos\left(\frac{\pi(2k-1)}{2n}\right), \qquad 1 \le k \le n\,.$$

2.5. Consider the polynomial $p(x) = x^3 - 2x - 5$. Plot this on $0 \le x \le 4$ and see thereby that there is a root near $x = 2$. Shift the basis to 1, $(x-2)$, $(x-2)^2$, and $(x-2)^3$. By neglecting all but the first two terms, get an improved approximate root. Shift the basis again to 1, $(x-r)$, $(x-r)^2$, and $(x-r)^3$, where r is your estimated root. Neglect all but the first two terms again, and solve to get an even more improved root. Repeat the process until you have identified the root to machine accuracy. As discussed in the text, this is how Newton conceived of what we now call Newton's method.

2.6. Prove the discrete orthogonality of the Chebyshev polynomials on the zeros of $T_n(x)$, and show thereby that Eq. (2.31) gives the Chebyshev coefficients of a given $p(x)$ with degree at most $n-1$.

2.7. Download and install the Chebfun package. Execute the following commands.

```
close all
plot( chebpoly(0), 'k' );
hold on
for i=1:30,
  plot( chebpoly(i), 'k' );
end;
axis('square')
```

Explain what you see. The discussion by Rivlin (1990) is very extensive, but for this problem a simple description will suffice (this problem is more about syntax than anything).

2.8. Use the change of variables $x = \cos\theta$ to show that

$$\int T_n(x)\,dx = \frac{1}{2(n+1)}T_{n+1}(x) - \frac{1}{2(n-1)}T_{n-1}(x) + C$$

for $n \ge 1$, for some constant C. Since $\int T_0(x)\,dx = T_1(x) + C$, this gives a beautiful short formula for integrals of Chebyshev polynomials. The formula for derivatives of T_n, expressed in terms of lower-degree Chebyshev T polynomials, is not so elegant, but useful nonetheless. It is found in Rivlin (1990), and also as an MAPLE program in Corless (2002).

2.9. Show that the Chebyshev–Lobatto points $\eta_k^{(n)} = \cos(k\pi/n)$ are the zeros of the polynomial

$$(1-x^2)\frac{\sin n\theta}{\sin\theta},$$

where $x = \cos\theta$. The polynomial $U_{n-1}(x) := \sin n\theta / n\sin\theta$ (note the extra n in the denominator) is called the Chebyshev polynomial of the second kind, and the η_k are sometimes called the Chebyshev points of the second kind.

2.10. Show that the Chebyshev–Lobatto points $\eta_k^{(n)} = \cos(k\pi/n)$ are (also) the zeros of the monic polynomial

$$w(x) = \prod_{k=0}^{n}\left(x-\eta_k^{(n)}\right) = 2^{-n}\left(T_{n+1}(x) - T_{n-1}(x)\right)$$

if $n \geq 1$. A discussion of this result can be found in Trefethen (2013).

2.11. Show that Horner's method recursively applied to $p(z) = p(\tau_k) + q(z)(z-\tau_k)$ gives Algorithm 2.1.

2.12. Prove Lemma 2.1 on page 61.

2.13. Show that for every pair of sets of polynomial bases $\phi_k(x)$ and $\psi_k(x)$, $0 \leq k \leq n$, there exists a nonsingular matrix $\mathbf{A}_{\psi\phi}$ for which

$$\left[\psi_0(x), \psi_1(x), \ldots, \psi_n(x)\right] = \left[\phi_0(x), \phi_1(x), \ldots, \phi_n(x)\right]\mathbf{A}_{\psi\phi}\,.$$

Show $\mathbf{A}_{\phi\psi} = \mathbf{A}_{\psi\phi}^{-1}$.

2.14. Show that Algorithm 2.4 costs $O(d^2)$ flops to execute. Discuss the varying cases when all s_k are small and the opposite case when only one or two nodes have high confluency.

2.15. Plot the condition numbers for evaluating the scaled Wilkinson polynomial

$$W_{20}(x) = \prod_{k=1}^{20}\left(x - \frac{k}{21}\right),$$

in each of the following bases:

1. monomial basis $\phi_k(x) = x^k$;
2. Bernstein–Bézier on $[0,1]$,

$$\phi_k(x) = \binom{20}{k}x^k(1-x)^{20-k};$$

3. Lagrange basis on $\tau_k = \frac{k}{21}$, $0 \le k \le 20$;
4. Lagrange basis on random nodes τ_k chosen from a uniform distribution on $[0,1]$.

2.16. Consider the Taylor series for the Airy function $\mathrm{Ai}(z)$. Think about each term as $a_k z^k$, for $0 \le k \le 127$. For $z = 10$, at which a 127-degree Taylor polynomial *ought* to give an accurate answer, compute all 128 of these terms separately, and plot them on a graph, with k on the horizontal axis. Verify that the largest term occurs at about $k = 30$, and has size about 10^7. This picture is why the phenomenon is known as "the hump." What does this have to do with our condition number analysis in the text?

2.17. Compute $e^x = \sum_{k=0}^{\infty} x^k/k!$ for various values of x and truncate the series at various values of N. Does this series converge, in theory? Why, then, does it compute (say) $\exp(-30)$ to such poor relative accuracy? Compare with Problems 2.16 and 1.7.

2.18. In this problem, we examine exact formulæ for finding zeros of polynomials of low degree. To begin with, the zero polynomial $f_0(z) \equiv 0$ is exceptional; with $\deg f_0(z) = -\infty$, it is zero no matter what z is. Also, degree-0 polynomials, of the form $f_0(z) = a_0$, $a_0 \neq 0$, are never zero; they have no roots. Moreover, degree-1 polynomials, of the form $f_1(z) = a_0 + a_1 z$, $a_1 \neq 0$, have one root, which is given by $z = -a_0/a_1$. Notice that, as $a_1 \to 0$, unless $a_0 = 0$, this root goes to (complex) infinity. These are very straightforward cases. Degree-2 polynomials are already more interesting:

1. Show that $f_2(z) = a_0 + a_1 z + a_2 z^2$ with $a_2 \neq 0$ may be written as

$$f_2(z) = a_2 \left(z + \frac{a_1}{2a_2} \right)^2 - \frac{1}{4a_2} \left(a_1^2 - 4a_2 a_0 \right)$$

(because $a_2 \neq 0$), and that therefore the two roots of $f_2(z)$ are

$$z = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_2 a_0}}{2a_2} \,,$$

and that as $a_2 \to 0$, if $a_1 \neq 0$ that one root tends to $-a_0/a_1$ and the other tends to ∞. If $a_1 = 0$, then both roots tend to ∞ (remember: $a_k \in \mathbb{C}$).
2. What is the absolute condition number of the roots?

Now, let us turn to degree-3 polynomials and, in particular, Cardano's method:

1. Consider a third-degree polynomial $f_3(z) = a_0 + a_1 z + a_2 z^2 + a_3 z^3 = 0$. Show that, using $z = t - a_2/3a_3$, this is equivalent to solving $t^3 + pt + q = 0$.
2. Let $t = u + v$ and gather terms so that

$$u^3 + v^3 + q + (3uv + p)(u + v) = 0 \,.$$

Conclude that if one can find u and v such that

$$\begin{aligned} 3uv + p &= 0 \\ u^3 + v^3 + q &= 0 \end{aligned}$$

simultaneously, then one can solve any cubic equation.

3. By solving $u^3v^3 = -p^3/27$ and $u^3 + v^3 = -q$ simultaneously for, say, u^3 first, and then using $uv = -p/3$ to find v unambiguously, show that you really can solve cubics.
4. The "condition numbers" $\frac{\partial x}{\partial a_k}$, $\left[\frac{\partial t}{\partial p}, \frac{\partial t}{\partial q}\right]$ and $\left[\frac{\partial u}{\partial p}, \frac{\partial u}{\partial q}, \frac{\partial v}{\partial p}, \frac{\partial v}{\partial q}\right]$ are different but related. Discuss. In particular, is the use of Cardano's formula always numerically stable?

Finally, let's have a look at degree-4 polynomials (encountered in quartic equations), and in particular Descartes' method:

1. Convert $f(z) = a_0 + a_1z + a_2z^2 + a_3z^3 + a_4z^4 = 0$, with $a_4 \neq 0$, to $F(t) = t^4 + pt^2 + qt + r = 0$.
2. Show that if

$$\begin{aligned} v + w - u^2 &= p \\ u(w - v) &= q \\ vw &= r\,, \end{aligned}$$

then $F(t) = (t^2 + ut + v)(t^2 - ut + u)$. Eliminate v and w to find a cubic equation for u^2.
3. Discuss the conditioning of the transformed problems.

2.19. Show that if

$$f(z) = \sum_{k=0}^{n} c_k\phi_k(z) \qquad \text{and} \qquad (f + \Delta f)(z) = \sum_{k=0}^{n} c_k(1 + \delta_k)\phi_k(z)$$

with $|\delta_k| \leq \varepsilon w_k$, that for each simple root $\hat{z}$ of f, when $\varepsilon > 0$ is small enough, that there is a simple root $\tilde{z}$ of $f + \Delta f$ such that $\tilde{z} = \hat{z} + \Delta z$ and

$$|\Delta z| \leq \frac{\varepsilon B(\hat{z})}{|f'(\hat{z})|} + O(\varepsilon^2)\,,$$

where $B(z) = \sum_{k=0}^{n} w_k|\phi_k(z)|$.

2.20. Find a recurrence relation for the series coefficients of $y = \ln u$ if

$$u = \sum_{k=0}^{N} u_k(x - a)^k + O(x - a)^{N+1}$$

and $u_0 \neq 0$.

2.21. Find a recurrence relation for the series coefficients of s and c, where $s = \sin(u)$ and $c = \cos(u)$, if

$$u = \sum_{k=0}^{N} u_k (x-a)^k + O(x-a)^{N+1} .$$

2.22. The JCP Miller formula. If $y = u^\alpha$ (for constant α), find a recurrence relation for the series coefficients of y by use of ${}^{dy}\!/_{dx}$, where

$$u = \sum_{k=0}^{N} u_k (x-a)^k + O(x-a)^{N+1} .$$

2.23. The Airy function $\mathrm{Ai}(z)$ satisfies the differential equation

$$\frac{d^2y}{dz^2} = zy(z) \tag{2.120}$$

with the initial conditions $y(0) = 3^{1/3}/(3\Gamma(2/3))$ and $y'(0) = -3^{1/6}\Gamma(2/3)/(2\pi)$. Use the methods of this section to generate a recurrence relation that determines the Taylor coefficients of $\mathrm{Ai}(z)$ in its series about 0. (That recurrence is used in the programs in Exercise 2.29.) As for the exponential, sine and cosine, and logarithm, this can be extended to allow you to generate recurrence relations for the series coefficients of $\mathrm{Ai}(u(z))$, where $u(z)$ is known by a truncated power series.

2.24. This problem is on series reversion. Suppose that

$$x = x_0 + x_1(y-y_0) + x_2(y-y_0)^2 + \dots ,$$

that we know the x_k, and that $x_1 \neq 0$. We wish to find the coefficients y_k so that

$$y = y_0 + y_1(x-x_0) + y_2(x-x_0)^2 + \dots .$$

Proceed as follows. Take

$$\begin{aligned} x = x_0 + x_1(y_1(x-x_0) + y_2(x-x_0)^2 + \dots) + x_2(y_1(x-x_0) + y_2(x-x_0)^2 + \dots)^2 \\ + x_3(y_1(x-x_0) + \dots)^3 + \dots , \end{aligned}$$

expand, and solve for y_1, y_2 and y_3 in turn. This is the brute force approach. See Henrici (1974) for a discussion of the Lagrange–Bürmann theorem, which explores an elegant connection (perhaps originally due to Lambert, if we are to believe his claims about his *Acta Helvetica* paper) to powers of the series being reverted.

2.25. Using your answer: For Problem 2.24, find the first three terms of the series for $\tan(x)$ about $x = 0$ from the series for $\arctan(x) = \int_0^x \frac{dt}{1+t^2}$.

2.26. Let $f(x,y,t) = 0$ with $x = x(t), y = y(t)$, $g(x,y,t) = 0$, and

$$f(x_0, y_0, t_0) = g(x_0, y_0, t_0) = 0\,.$$

Show that if the Jacobian determinant

$$\det \begin{bmatrix} f_x(x_0,y_0,t_0) & f_y(x_0,y_0,t_0) \\ g_x(x_0,y_0,t_0) & g_y(x_0,y_0,t_0) \end{bmatrix} \tag{2.121}$$

is not zero, and f and g are analytic in all variables, then $x(t) = x_0 + x_1(t - t_0) + \ldots$ and $y(t) = y_0 + y_1(t - t_0) + \ldots$ may be constructively developed in Taylor series to as many terms as one likes. (Hint: differentiate. This is the implicit function theorem.)

2.27. If you have access to MAPLE, solve

$$\begin{aligned} x^2 + y^2 &= t^2 \\ 25xy - 12 &= 0 \end{aligned}$$

in series for x and y near $t = 1$, when $x = 3/5$ and $y = 4/5$ (there are three other intersections also; just follow this one). (Hint: You can `dsolve/series`, which implements the ideas of this chapter, but differentiate first.)

2.28. If you have access to MAPLE, consider the command

```
convert( R, parfrac, z, true );
```

when R is a simple factored rational function, say $\prod_{i=0}^{n}(x - \tau_i)^{-1}$, with (say) Chebyshev–Lobatto nodes computed to 16 digits of precision, via commands similar to

```
Digits := 16;
n := 5;
tau := [ seq( evalf( cos(Pi*j/n) ), j = 0..n ) ];
R := 1/mul( z-tau[1+j], j = 0..n);
PF := convert( R, parfrac, z, true );
ONE := PF/R;
plots[logplot]( abs(ONE-1), z = -1 .. 1, colour = BLACK, style=
    POINT );
```

Try these commands for various n. How well does MAPLE do, in your version? At this time of writing, MAPLE 15 makes acceptable plots for n as large as 15, but it's already bad for $n = 20$.

Investigations and Projects

2.29. The following MAPLE program uses a handwritten version of Horner's method to evaluate the degree-N Taylor polynomial at $z = 0$ for the Airy function $\mathrm{Ai}(z)$.

```
#
# Horner form of Taylor polynomial approximation to AiryAi(z)
#
TaylorAi := proc( z, N )
  local Ai0, Aip0, f1, f2, k, n, z3, zsq;
  Ai0  := evalf( 3^(1/3)/(3*GAMMA(2/3)) );
  Aip0 := evalf( -3^(1/6)*GAMMA(2/3)/(2*Pi) );
  z3   := evalf( z*z*z );
  n    := max( floor( (N-2)/3 ), 0 );
  f1 := 1;
  f2 := 1;
  for k from n by -1 to 1 do
     f1 := evalf( 1 + z3*f1/((3*k)*(3*k-1)) );
     f2 := evalf( 1 + z3*f2/((3*k+1)*(3*k)) );
  end do;
  return Ai0*f1 + Aip0*z*f2
end;
```

When this is translated into MATLAB via the CodeGeneration[Matlab] feature of MAPLE, and the resulting code is polished a bit by hand, the result is

```
%
% Automatic translation of TaylorAi.mpl
% which was written by RMC 2011, using
% CodeGeneration[Matlab] in Maple
% plus fixups GAMMA --> gamma
%             vectorized multiplications
%             added "end" to function
function TaylorAireturn = TaylorAi( z, N )
  Ai0 = ((3 ^ (0.1e1 / 0.3e1) / gamma(0.2e1 / 0.3e1)) / 0.3e1);
  Aip0 = (-(3 ^ (0.1e1 / 0.6e1)) * gamma(0.2e1 / 0.3e1) / pi /
      0.2e1);
  z3 = (z .* z .* z);
  n = max( floor(N / 0.3e1 - 0.2e1 / 0.3e1), 0 );
  f1 = 1;
  f2 = 1;
  for k = n:-1:1
    f1 = (0.1e1 + z3 .* f1 / k / (3 * k - 1) / 0.3e1);
    f2 = (0.1e1 + z3 .* f2 / (3 * k + 1) / k / 0.3e1);
  end
  TaylorAireturn = Ai0 * f1 + Aip0 * z .* f2;
end
```

The automatically generated and curiously ugly `0.3e1` meaning `3.0`, and its ilk, were left as-is. Notice also that this automatically generated code does not follow the MATLAB style guidelines of Johnson (2010). The following MATLAB commands

```
z = linspace( -13, 13, 40012 );
y = TaylorAi( z, 127 );
relerr = y./airy(z) - 1;
semilogy( z, abs(relerr), 'k.' )
xlabel( 'z' ), ylabel( 'relative␣error' )
axis([-15 15 10E-21 10E14]);
set(gca, 'YTick', [10.^-20, 10.^-15, 10.^-10, 10.^-5, 10.^0,
    10.^5, 10.^10, 10.^15]);
```

produce the plot in Fig. 2.10. Explain this plot in general. Can you explain the curious horizontal line starting at about $z = 7$? If you have access to MAPLE, you might consider running the original program at varying levels of precision, say `Digits` equal to 5, 10, 15, 20, and 25, in order to help.

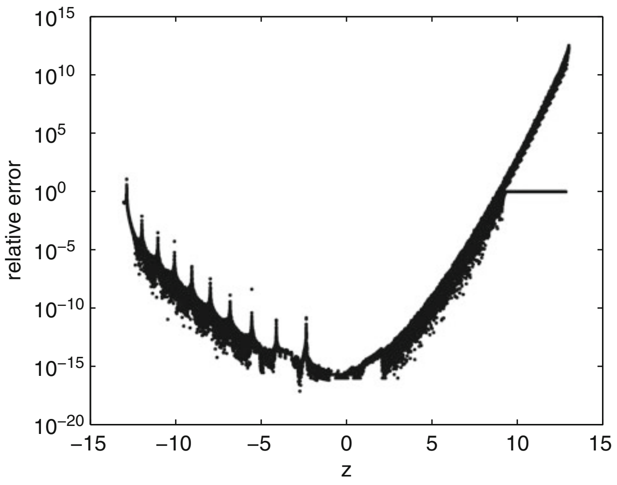

Fig. 2.10 The output of the program in Problem 2.29

2.30. Prove that you may algorithmically compute the Taylor coefficients of any function or set of functions defined by a system of polynomial or rational differential equations such as this:

$$\frac{dy_1}{dx} = f_1(x, y_1, y_2, \ldots, y_n) \tag{2.122}$$

$$\frac{dy_2}{dx} = f_1(x, y_1, y_2, \ldots, y_n) \tag{2.123}$$

$$\vdots$$

$$\frac{dy_n}{dx} = f_1(x, y_1, y_2, \ldots, y_n) \tag{2.124}$$

with $\mathbf{y}(a) = y_a$ given, and each f_i polynomial or rational in its arguments (with no poles at a, $\mathbf{y}(a)$ in any f_i). This is quite a large class of functions!

2.31. We know of one function, the Γ function (and its derivatives), that does not fall into the class of functions in Problem 2.30. Can you think of any others? Describe some.

2.32. Draw the pseudozero sets for the following polynomials as in Fig. 2.7. Choose interesting contour levels. Use weights equal to the polynomial coefficients.

1. $T_{20}(x)$. Compare with the Wilkinson polynomial of degree 20.
2. $q(x) = (x-1)^{30}(x-2)^{18}(x-3)^{12}$ (Zeng 2004).
3. $p(x) = x^{17} - (4x-1)^3$ (Bini and Mourrain 1996).
4. The Fibonacci polynomials $f_n(x) = x^n - \sum_{k=0}^{n-1} x^k$ for, say, $n = 5$ and $n = 10$.
5. For any of the Zeng (2004) test polynomials that you fancy.
6. One of the paper by Wilkinson (1959a).

2.33. Draw the first 30 Chebyshev polynomials on the same graph (like Fig. 2.1 but with more of them). You should see several smooth curves suggested by the gaps in the graph; these curves are called “ghost curves.” They can be described analytically. See Rivlin (1990).

2.34. Functions containing square roots or other radicals may not have Taylor series at the branch point. A useful extension is *Puiseux* series, that is, series in terms of powers of $(z-a)^{1/p}$ for some p. Compute five terms of the Puiseux series of $\sin(\exp(\sqrt{x}) - 1)$ about $x = 0$.

2.35. The Mandelbrot polynomials are defined by $p_0(z) = 1$ and

$$p_{k+1}(z) = z p_k^2(z) + 1\,.$$

Expanding these polynomials in the monomial basis is a bad idea. Demonstrate this by proving that the coefficients are all positive, the leading coefficient is 1 as is the trailing coefficient, and at least one coefficient grows doubly exponentially with k (the degree is exponential in k, so the coefficients grow exponentially in the degree). Explain why this makes the condition number $B(z)$ of the monomial basis expression very large on the interval $-2 \le z \le 0$.

2.36. Implement and test the Clenshaw algorithm (see Algorithm 2.2) for the Chebyshev polynomials, which have $\alpha_k(z) = 2z$ and $\beta_k = 1$ for $k \ge 2$.

2.37. The first barycentric form is

$$p(z) = w(z) \sum_{k=0}^{n} \frac{\beta_k \rho_k}{z - \tau_k},$$

where the ρ_k are the values of $p(z)$ at $z = \tau_k$; that is, $\rho_k = p(\tau_k)$. Since this is true for all $p(z)$, it is in particular true for the constant polynomial 1:

$$1 = w(z) \sum_{k=0}^{n} \frac{\beta_k \cdot 1}{z - \tau_k}\,.$$

Dividing these two gives us the *second* barycentric form,

$$p(z) = \frac{\sum_{k=0}^{n} \frac{\beta_k \rho_k}{z - \tau_k}}{\sum_{k=0}^{n} \frac{\beta_k}{z - \tau_k}},$$

about which we will learn more in Chap. 8. By cross-multiplying and using the product rule, find an expression for the derivative of a polynomial expressed in the second barycentric form of the Lagrange basis. What is the cost to evaluate this, supposing that the β_k are available?

2.38. Once one has found an approximate root r of a polynomial, one usually wants to deflate, that is, find a new polynomial $q(z) = p(z)/(z-r)$ that has the same roots as the other roots of $p(z)$ but is one degree less. Done incorrectly, this can lead to instability; Wilkinson advocated deflating roots from smallest magnitude to largest, but it has since been realized that by reversing the polynomial, that is, considering the polynomial $P(z) = z^n p(1/z)$, which has as roots the reciprocals of the roots of p, one can instead deflate from largest to smallest. Use Newton's method, synthetic division, and deflation to find all roots of Newton's example polynomial $p(z) = z^3 - 2z - 5$.

2.39. Show how to reverse polynomials $P(z) = z^n p(1/z)$ that are expressed in a Lagrange basis. Do not convert to the monomial basis.

2.40. Show that if the forward error $e_k = r_k - z^*$ in an approximate root to a polynomial $p(z) = 0$ is small, and $p'(z) \neq 0$ nearby, then the next iteration r_{k+1} has forward error proportional to the square of e_k. This is called quadratic convergence.

2.41. We said in the text that elements of the sequence of Newton iterates $r_{k+1} = r_k - p(r_k)/p'(r_k)$ were each *exact* solutions of the polynomials $p(z) - p(r_k) = 0$. This is trivial, in one sense, and very useful in another if $p(r_k)$ is small enough to be ignored. There is another way to look at this that is also useful. Given an approximate root r_k for $p(z)$, we can ask, "What is the *closest* polynomial $p(z) + \Delta p(z)$ for which r_k is an exact root?"

We know that the size of $\Delta p(z)$ is at most $|p(r_k)|$ by the previous "trivial" statement. But are there closer polynomials? The answer is usually yes, and there is an analytical formula for the coefficients of the optimal $\Delta p(z)$ that we can find using the Hölder inequality, as follows.

Given $p(z) = \sum_{k=0}^{n} c_k \phi_k(z)$, weights $w_k \geq 0$ for $0 \leq k \leq n$ not all zero, and a putative root r, find the minimum ε such that $(p + \Delta p)(r) = 0$ with $\Delta p(z) = \sum_{k=0}^{n} (\Delta c_k)\phi_k(z)$ such that each $|\Delta c_k| \leq w_k \varepsilon$. Then ε is the "minimal backward error" of the root r; you should find that ε is proportional to $|p(r)|$, the residual. (Hint: Reread Theorem 2.9 on page 70 and then use (C.2) in Appendix C.)

2.42. Following the discussion in Sect. 2.2.6.3 and the solution of the previous problem, find an expression for the nearest polynomial of lower degree.