Introduction to Numerical Analysis

Hector D. Ceniceros

Contents

List of Figures

List of Tables

Preface

Numerical analysis is a discipline of mathematics with applications in practically every area of science and engineering. There is a large collection of classical and modern introductory texts on this broad subject. This book is an attempt to focus on the basic principles of numerical analysis rather than a presentation of a comprehensive list of numerical methods for different mathematical problems. To this effect, this book contains a non-traditional arrangement of a selection of traditional topics and some topics are covered more extensively than usual. It is written from a mathematical perspective but always keeping in mind the modern practice of numerical analysis.

This book starts with an introduction to highlight some of the basic principles of numerical analysis with one illustrative example, the approximation of a definite integral. This is followed by some basic topics of approximation theory. After this, the different ideas of function approximation are used in the derivation and analysis of a selection of numerical methods. There is no attempt to cover all major topics of numerical analysis but rather to focus on a subset which I believe illustrate well fundamental principles and the power of numerical mathematics. This focused selection of topics was made to appropriately fit a year-long, undergraduate, introductory course on numerical analysis. In fact, it grew out of a set of lecture notes I prepared for a three-quarter, upper division undergraduate course of numerical analysis at the University of California at Santa Barbara.

This book is intended for undergraduate students with a solid mathematics background. The prerequisites are vector calculus, linear algebra, and an introductory course in analysis. Some rudimentary knowledge of differential equations and complex variables is desirable. It is also very important to have the ability to write simple computer codes to implement the numerical methods as this is an essential part of learning numerical analysis.

This book is not in finalized form and may contain errors, mis-

Acknowledgements:

Chapter 1

Introduction

1.1 What is Numerical Analysis?

This is an introductory course of numerical analysis, which *comprises the design, analysis, and implementation of constructive methods and algorithms for the solution of mathematical problems.*

Numerical analysis has vast applications both in mathematics and in modern science and technology. In the areas of the physical and life sciences, numerical analysis plays the role of a virtual laboratory by providing accurate solutions to the mathematical models representing a given physical or biological system in which the system's parameters can be varied at will, in a controlled way. The applications of numerical analysis also extend to more modern areas such as data analysis, web search engines, social networks, and just about anything where computation is involved.

1.2 An Illustrative Example: Approximating a Definite Integral

The purpose of this chapter is to illustrate with one example some of the main principles and objectives of numerical analysis. The example is the calculation of a definite integral:

$$I[f] = \int_a^b f(x)dx. \tag{1.1}$$

In most cases we cannot find an exact value of $I[f]$ and very often we only know the integrand f at a finite number of points in $[a, b]$. The problem is then to produce an approximation to $I[f]$ as accurate as we need and at a reasonable computational cost.

1.2.1 An Approximation Principle

One of the central ideas in numerical analysis is to approximate a given function or data by simpler functions which we can analytically evaluate, integrate, differentiate, etc. For example, we can approximate the integrand f in $[a, b]$ by the segment of the straight line, a polynomial of degree at most 1, that passes through $(a, f(a))$ and $(b, f(b))$

$$f(x) \approx p_1(x) = f(a) + \frac{f(b) - f(a)}{b - a}(x - a). \tag{1.2}$$

and approximate the integral of f by the integral of p_1, as Fig. 1.1 illustrates,

$$\begin{aligned} \int_a^b f(x)dx \approx \int_a^b p_1(x)dx &= f(a)(b - a) + \frac{1}{2}[f(b) - f(a)](b - a) \\ &= \frac{1}{2}[f(a) + f(b)](b - a). \end{aligned} \tag{1.3}$$

That is

$$\int_a^b f(x)dx \approx \frac{(b - a)}{2}[f(a) + f(b)]. \tag{1.4}$$

The right hand side is known as the *(simple)* [1] *trapezoidal quadrature rule*. A quadrature rule or quadrature formula is a method to approximate an integral. How accurate is this approximation? Clearly, if f is a linear polynomial or a constant, then the trapezoidal rule would give us the exact value of the integral. The underlying question is: how well does a polynomial of degree at most 1, p_1, satisfying

$$p_1(a) = f(a), \tag{1.5}$$
$$p_1(b) = f(b), \tag{1.6}$$

[1] There are simple and composite quadratures, as we will see shortly.

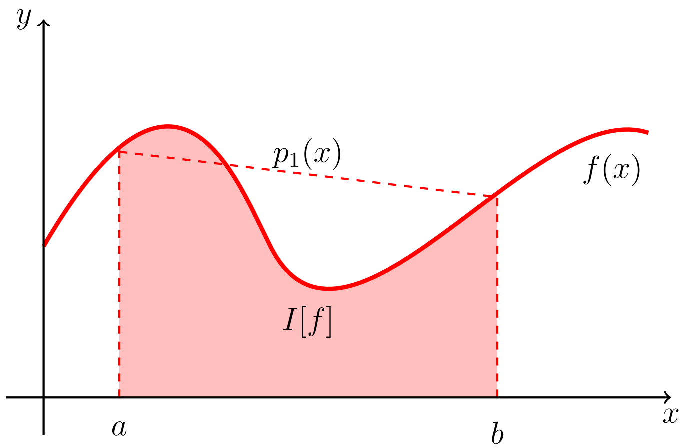

Figure 1.1: Trapezoidal rule approximation for definite integrals. The integrand f is approximated by p_1.

approximate f on the interval $[a, b]$? The approximation is exact at $x = a$ and $x = b$ because of (1.5)-(1.6) and is exact for all polynomials of degree ≤ 1. In fact, assuming $f \in C^2[a, b]$, we are going to prove that for $x \in [a, b]$

$$f(x) - p_1(x) = \frac{1}{2} f''(\xi(x))(x - a)(x - b), \tag{1.7}$$

for some $\xi(x) \in (a, b)$.

If $x = a$ or $x = b$, then (1.7) holds trivially. So let us take x in (a, b) and define the following function of a new variable t

$$\phi(t) = f(t) - p_1(t) - [f(x) - p_1(x)] \frac{(t - a)(t - b)}{(x - a)(x - b)}. \tag{1.8}$$

Then ϕ, as a function of t, is $C^2[a, b]$ and $\phi(a) = \phi(b) = \phi(x) = 0$. Since $\phi(a) = \phi(x) = 0$, by Rolle's theorem there is $\xi_1 \in (a, x)$ such that $\phi'(\xi_1) = 0$ and similarly there is $\xi_2 \in (x, b)$ such that $\phi'(\xi_2) = 0$. Because ϕ is $C^2[a, b]$ we can apply Rolle's theorem one more time, observing that $\phi'(\xi_1) = \phi'(\xi_2) = 0$, to conclude that there is a point $\xi(x)$ between ξ_1 and ξ_2 such that $\phi''(\xi(x)) = 0$. Consequently,

$$0 = \phi''(\xi(x)) = f''(\xi(x)) - [f(x) - p_1(x)] \frac{2}{(x - a)(x - b)} \tag{1.9}$$

and so

$$f(x) - p_1(x) = \frac{1}{2} f''(\xi(x))(x-a)(x-b), \qquad \xi(x) \in (a,b). \quad \square \tag{1.10}$$

We can now use (1.10) to find the accuracy of the simple trapezoidal rule. Assuming the integrand f is $C^2[a,b]$

$$\int_a^b f(x)dx = \int_a^b p_1(x)dx + \frac{1}{2}\int_a^b f''(\xi(x))(x-a)(x-b)dx. \tag{1.11}$$

Now, $(x-a)(x-b)$ does not change sign in $[a,b]$ and f'' is continuous so by the weighted mean value theorem for integrals, we have that there is $\eta \in (a,b)$ such that

$$\int_a^b f''(\xi(x))(x-a)(x-b)dx = f''(\eta)\int_a^b (x-a)(x-b)dx. \tag{1.12}$$

The last integral can be easily evaluated by shifting to the midpoint, i.e., changing variables to $x = y + \frac{1}{2}(a+b)$ then

$$\int_a^b (x-a)(x-b)dx = \int_{-\frac{b-a}{2}}^{\frac{b-a}{2}} \left[y^2 - \left(\frac{b-a}{2}\right)^2 \right] dy = -\frac{1}{6}(b-a)^3. \tag{1.13}$$

Collecting (1.11) and (1.13) we get

$$\int_a^b f(x)dx = \frac{(b-a)}{2}[f(a)+f(b)] - \frac{1}{12} f''(\eta)(b-a)^3, \tag{1.14}$$

where η is some point in (a,b). So in the approximation

$$\int_a^b f(x)dx \approx \frac{(b-a)}{2}[f(a)+f(b)].$$

we make the error

$$E[f] = -\frac{1}{12} f''(\eta)(b-a)^3. \tag{1.15}$$

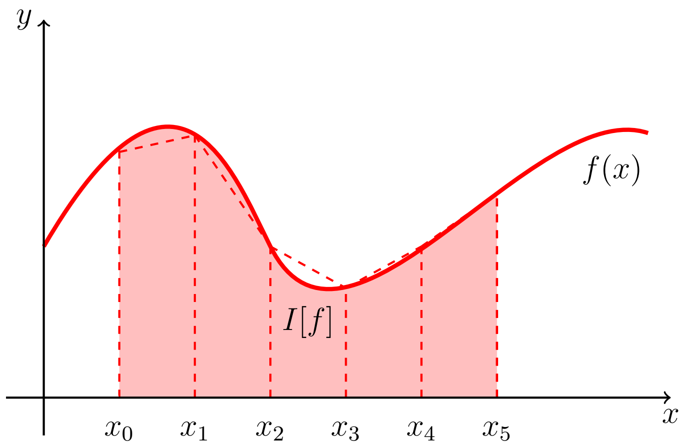

Figure 1.2: Composite trapezoidal rule for $N = 5$.

1.2.2 Divide and Conquer

The error (1.15) of the simple trapezoidal rule grows cubically with the length of the interval of integration so it is natural to divide $[a, b]$ into smaller subintervals, $[x_0, x_1], [x_1, x_2], \ldots [x_{N-1}, x_N]$, apply the trapezoidal rule on each of them, and sum up the result. Figure 1.2 illustrates the idea for $N = 5$. Let us take subintervals of equal length $h = \frac{1}{N}(b - a)$, determined by the points $x_0 = a, x_1 = x_0 + h, x_2 = x_0 + 2h, \ldots, x_N = x_0 + Nh = b$. Then

$$\begin{aligned}\int_a^b f(x)dx &= \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \ldots + \int_{x_{N-1}}^{x_N} f(x)dx \\ &= \sum_{j=0}^{N-1} \int_{x_j}^{x_{j+1}} f(x)dx.\end{aligned} \tag{1.16}$$

But we know

$$\int_{x_j}^{x_{j+1}} f(x)dx = \frac{1}{2}[f(x_j) + f(x_{j+1})]h - \frac{1}{12} f''(\xi_j) h^3 \tag{1.17}$$

for some $\xi_j \in (x_j, x_{j+1})$. Therefore, we get

$$\int_a^b f(x)dx = h\left[\frac{1}{2}f(x_0) + f(x_1) + \ldots + f(x_{N-1}) + \frac{1}{2}f(x_N)\right] - \frac{1}{12}h^3 \sum_{j=0}^{N-1} f''(\xi_j).$$

The first term on the right hand side is called the *composite trapezoidal rule*:

$$T_h[f] := h\left[\frac{1}{2}f(x_0) + f(x_1) + \ldots + f(x_{N-1}) + \frac{1}{2}f(x_N)\right]. \tag{1.18}$$

The error of this quadrature is

$$E_h[f] = -\frac{1}{12}h^3\sum_{j=0}^{N-1} f''(\xi_j) = -\frac{1}{12}(b-a)h^2\left[\frac{1}{N}\sum_{j=0}^{N-1} f''(\xi_j)\right], \tag{1.19}$$

where we have used that $h = (b-a)/N$. The term in brackets is a mean value of f'' (it is easy to prove that it lies between the maximum and the minimum of f''). Since f'' is assumed continuous ($f \in C^2[a,b]$), by the intermediate value theorem there is a point $\xi \in (a,b)$ such that

$$f''(\xi) = \frac{1}{N}\sum_{j=0}^{N-1} f''(\xi_j). \tag{1.20}$$

Thus, the error of the simple trapezoidal rule can be written as

$$E_h[f] = -\frac{1}{12}(b-a)h^2 f''(\xi), \tag{1.21}$$

for some $\xi \in (a,b)$.

1.2.3 Convergence and Rate of Convergence

We do not know what the point ξ is in (1.21). If we knew, the error could be evaluated and we would know the integral exactly, at least in principle, because

$$I[f] = T_h[f] + E_h[f]. \tag{1.22}$$

But (1.21) gives us two important properties of the approximation method in question. First, (1.21) tell us that $E_h[f] \to 0$ as $h \to 0$. That is, the quadrature rule $T_h[f]$ **converges** to the exact value of the integral as $h \to 0$. [2] Recall $h = (b-a)/N$, so as we increase N our approximation to the integral

[2]Neglecting round-off errors introduced by the computer finite precision representation of numbers and by computer arithmetic (Chapter 5).

gets better and better. Second, (1.21) tells us how fast the approximation converges, namely quadratically in h. This is the approximation's **rate of convergence**. If we double N (or equivalently halve h), the error decreases by a factor of 4. We also say that the error is order h^2 and write $E_h[f] = O(h^2)$. The Big 'O' notation is used frequently in numerical analysis.

Definition 1.1. *We say that $g(h)$ is order h^α, and write $g(h) = O(h^\alpha)$, if there is a constant C and h_0 such that $|g(h)| \leq Ch^\alpha$ for $0 \leq h \leq h_0$, i.e. for sufficiently small h.*

Example 1.1. *Let's check the composite trapezoidal rule approximation for an integral we can compute exactly. Take $f(x) = e^x$ in $[0, 1]$. The exact value of the integral is $e - 1$. The approximation for some values of N is shown in Table 1.1. Observe how the error $|I[f] - T_{1/N}[f]|$ decreases by a factor of*

Table 1.1: Composite Trapezoidal Rule for $f(x) = e^x$ in $[0, 1]$.

N	$T_{1/N}[f]$	$\lvert I[f] - T_{1/N}[f]\rvert$	Decrease factor
16	1.718841128579994	$5.593001209489579 \times 10^{-4}$	
32	1.718421660316327	$1.398318572816137 \times 10^{-4}$	0.250012206406039
64	1.718316786850094	$3.495839104861176 \times 10^{-5}$	0.250003051723810
128	1.718290568083478	$8.739624432374526 \times 10^{-6}$	0.250000762913303

(approximately) 1/4 as N is doubled, in accordance to (1.21).

1.2.4 Error Correction

We can get an upper bound for the error using (1.21) and the fact that f'' is bounded in $[a, b]$, i.e. $|f''(x)| \leq M_2$ for all $x \in [a, b]$ for some constant M_2. Then,

$$|E_h[f]| \leq \frac{1}{12}(b - a)h^2 M_2. \tag{1.23}$$

However, this bound might not be an accurate estimate of the actual error. This can be seen from (1.19). As $N \to \infty$, the term in brackets converges to a mean value of f'', i.e.

$$\frac{1}{N}\sum_{j=0}^{N-1} f''(\xi_j) \longrightarrow \frac{1}{b-a}\int_a^b f''(x)dx = \frac{1}{b-a}[f'(b) - f'(a)], \tag{1.24}$$

as $N \to \infty$, which could be significantly smaller than the maximum of $|f''|$. Take for example $f(x) = \frac{1}{2}x^2 - \sin 2\pi x$ on $[0,1]$. Then, $\max|f''| = 1 + 4\pi^2$, whereas the mean value (1.24) is equal to 1. Thus, (1.23) can overestimate the error significantly.

Equation (1.19) and (1.24) suggest that asymptotically, that is for sufficiently small h,

$$E_h[f] = C_2 h^2 + R(h), \tag{1.25}$$

where

$$C_2 = -\frac{1}{12}[f'(b) - f'(a)] \tag{1.26}$$

and $R(h)$ goes to zero faster than h^2 as $h \to 0$, i.e.

$$\lim_{h\to 0} \frac{R(h)}{h^2} = 0. \tag{1.27}$$

We say that $R(h) = o(h^2)$ (little 'o' h^2).

Definition 1.2. *A function $g(h)$ is little 'o' h^α if*

$$\lim_{h\to 0} \frac{g(h)}{h^\alpha} = 0$$

and we write $g(h) = o(h^\alpha)$.

We then have

$$I[f] = T_h[f] + C_2 h^2 + R(h) \tag{1.28}$$

and, for *sufficiently small* h, $C_2 h^2$ is an approximation of the error. If it is possible and computationally efficient to evaluate the first derivative of f at the end points of the interval then we can compute directly $C_2 h^2$ and use this leading order approximation of the error to obtain the improved approximation

$$\widetilde{T}_h[f] = T_h[f] - \frac{1}{12}[f'(b) - f'(a)]h^2. \tag{1.29}$$

This is called the (composite) *modified trapezoidal rule*. It then follows from (1.28) that error of this "corrected approximation" is $R(h)$, which goes to zero faster than h^2. In fact, we will prove later in Chapter 7 that the error of the modified trapezoidal rule is $O(h^4)$.

Often, we only have access to values of f and/or it is difficult to evaluate $f'(a)$ and $f'(b)$. Fortunately, we can compute a sufficiently good approximation of the leading order term of the error, C_2h^2, so that we can use the same *error correction* idea that we did for the modified trapezoidal rule. Roughly speaking, the error can be estimated by comparing two approximations obtained with different h.

Consider (1.28). If we halve h we get

$$I[f] = T_{h/2}[f] + \frac{1}{4}C_2h^2 + R(h/2). \tag{1.30}$$

Subtracting (1.30) from (1.28) we get

$$C_2h^2 = \frac{4}{3}\left(T_{h/2}[f] - T_h[f]\right) + \frac{4}{3}\left(R(h/2) - R(h)\right). \tag{1.31}$$

The last term on the right hand side is $o(h^2)$. Hence, for h sufficiently small, we have

$$C_2h^2 \approx \frac{4}{3}\left(T_{h/2}[f] - T_h[f]\right) \tag{1.32}$$

and this could provide a good, computable estimate for the error, i.e.

$$E_h[f] \approx \frac{4}{3}\left(T_{h/2}[f] - T_h[f]\right). \tag{1.33}$$

The key here is that h has to be sufficiently small to make the asymptotic approximation (1.32) valid. We can check this by working backwards. If h is sufficiently small, then evaluating (1.32) at $h/2$ we get

$$C_2\left(\frac{h}{2}\right)^2 \approx \frac{4}{3}\left(T_{h/4}[f] - T_{h/2}[f]\right) \tag{1.34}$$

and consequently the ratio

$$q(h) = \frac{T_{h/2}[f] - T_h[f]}{T_{h/4}[f] - T_{h/2}[f]} \tag{1.35}$$

should be approximately 4. Thus, $q(h)$ offers a reliable, computable indicator of whether or not h is sufficiently small for (1.33) to be an accurate estimate of the error.

We can now use (1.32) and the idea of error correction to improve the accuracy of $T_h[f]$ with the following approximation [3]

$$S_h[f] := T_h[f] + \frac{4}{3}\left(T_{h/2}[f] - T_h[f]\right) = \frac{4T_{h/2}[f] - T_h[f]}{3}. \tag{1.36}$$

1.2.5 Richardson Extrapolation

We can view the **error correction** procedure as a way to eliminate the leading order (in h) contribution to the error. Multiplying (1.30) by 4 and subtracting (1.28) to the result we get

$$I[f] = \frac{4T_{h/2}[f] - T_h[f]}{3} + \frac{4R(h/2) - R(h)}{3}. \tag{1.37}$$

Note that $S_h[f]$ is exactly the first term in the right hand side of (1.37) and that the last term converges to zero faster than h^2. This very useful and general procedure in which the leading order component of the asymptotic form of error is eliminated by a combination of two computations performed with two different values of h is called **Richardson's extrapolation**.

Example 1.2. *Consider again $f(x) = e^x$ in $[0, 1]$. With $h = 1/16$ we get*

$$q\left(\frac{1}{16}\right) = \frac{T_{1/32}[f] - T_{1/16}[f]}{T_{1/64}[f] - T_{1/32}[f]} \approx 3.9998 \tag{1.38}$$

and the improved approximation is

$$S_{1/16}[f] = \frac{4T_{1/32}[f] - T_{1/16}[f]}{3} = 1.718281837561771, \tag{1.39}$$

which gives us nearly 8 digits of accuracy (error $\approx 9.1\times10^{-9}$). $S_{1/32}$ yields an error $\approx 5.7 \times 10^{-10}$. It decreased by approximately a factor of $1/2^4 = 1/16$. This would correspond to fourth order *rate of convergence. We will see in Chapter 7 that indeed this is the case.*

$S_h[f]$ is superior than $T_h[f]$ in accuracy but apparently at roughly twice the computational cost. However, if we group together the common terms

[3] The symbol := means equal by definition.

in $T_h[f]$ and $T_{h/2}[f]$ we can compute $S_h[f]$ at about the same computational cost as that of $T_{h/2}[f]$:

$$
\begin{aligned}
4T_{h/2}[f] - T_h[f] &= 4\frac{h}{2}\left[\frac{1}{2}f(a) + \sum_{j=1}^{2N-1} f(a+jh/2) + \frac{1}{2}f(b)\right] \\
&\quad - h\left[\frac{1}{2}f(a) + \sum_{j=1}^{N-1} f(a+jh) + \frac{1}{2}f(b)\right] \\
&= \frac{h}{2}\left[f(a) + f(b) + 2\sum_{k=1}^{N-1} f(a+kh) + 4\sum_{k=1}^{N-1} f(a+kh/2)\right].
\end{aligned}
$$

Therefore

$$
S_h[f] = \frac{h}{6}\left[f(a) + 2\sum_{k=1}^{N-1} f(a+kh) + 4\sum_{k=1}^{N-1} f(a+kh/2) + f(b)\right]. \tag{1.40}
$$

The resulting quadrature formula $S_h[f]$ is known as the *composite Simpson's rule* and, as we will see in Chapter 7, can be derived by approximating the integrand by polynomials of degree ≤ 2. Thus, based on cost and accuracy, the composite Simpson's rule would be preferable to the composite trapezoidal rule, with one important exception: periodic smooth integrands integrated over their period (or multiple periods).

Example 1.3. *Consider the integral*

$$
\int_0^{2\pi} \frac{dx}{2+\sin x}. \tag{1.41}
$$

Using complex variables techniques (theory of residues) the exact integral can be computed and it is equal to $2\pi/\sqrt{3}$*. Note that the integrand is smooth (has an infinite number of continuous derivatives) and periodic in* $[0, 2\pi]$*. If we use the composite trapezoidal rule to find approximations to this integral we obtain the results shown in Table 1.2.*

The approximations converge amazingly fast. With $N = 32$*, we already reach machine precision (with double precision we get about 16 digits of accuracy).*

Table 1.2: Composite trapezoidal rule for $f(x) = 1/(2 + \sin x)$ in $[0, 2\pi]$.

N	$T_{2\pi/N}[f]$	$\left\vert I[f] - T_{2\pi/N}[f] \right\vert$
8	3.627791516645356	$1.927881769203665 \times 10^{-4}$
16	3.627598733591013	$5.122577029226250 \times 10^{-9}$
32	3.627598728468435	$4.440892098500626 \times 10^{-16}$

1.3 Super-Algebraic Convergence of the Composite Trapezoidal Rule for Smooth Periodic Integrands

Integrals of periodic integrands appear in many applications, most notably, in Fourier analysis.

Consider the definite integral

$$I[f] = \int_0^{2\pi} f(x)dx,$$

where the integrand f is periodic in $[0, 2\pi]$ and has $m > 1$ continuous derivatives, i.e. $f \in C^m[0, 2\pi]$ and $f(x + 2\pi) = f(x)$ for all x. Due to periodicity we can work in any interval of length 2π and if the function has a different period, with a simple change of variables, we can reduce the problem to one in $[0, 2\pi]$.

Consider the *equally spaced points* in $[0, 2\pi]$, $x_j = jh$ for $j = 0, 1, \ldots, N$ and $h = 2\pi/N$. Because f is periodic $f(x_0 = 0) = f(x_N = 2\pi)$. Then, the composite trapezoidal rule becomes

$$T_h[f] = h\left[\frac{f(x_0)}{2} + f(x_1) + \ldots + f(x_{N-1}) + \frac{f(x_N)}{2}\right] = h\sum_{j=0}^{N-1} f(x_j). \quad (1.42)$$

Being f smooth and periodic in $[0, 2\pi]$, it has a uniformly convergent Fourier series:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_k \cos kx + b_k \sin kx) \quad (1.43)$$

where

$$a_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos kx\, dx, \quad k = 0, 1, \ldots \tag{1.44}$$

$$b_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin kx\, dx, \quad k = 1, 2, \ldots \tag{1.45}$$

Using the Euler formula[4].

$$e^{ix} = \cos x + i \sin x \tag{1.46}$$

we can write

$$\cos x = \frac{e^{ix} + e^{-ix}}{2}, \tag{1.47}$$

$$\sin x = \frac{e^{ix} - e^{-ix}}{2i} \tag{1.48}$$

and the Fourier series can be conveniently expressed in complex form in terms of functions e^{ikx} for $k = 0, \pm 1, \pm 2, \ldots$ so that (1.43) becomes

$$f(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx}, \tag{1.49}$$

where

$$c_k = \frac{1}{2\pi}\int_0^{2\pi} f(x) e^{-ikx} dx. \tag{1.50}$$

We are assuming that f is real-valued so the complex Fourier coefficients satisfy $\bar{c}_k = c_{-k}$, where $\bar{c}_k$ is the complex conjugate of c_k. We have the relation $2c_0 = a_0$ and $2c_k = a_k - ib_k$ for $k = \pm 1, \pm 2, \ldots$, between the complex and real Fourier coefficients.

Using (1.49) in (1.42) we get

$$T_h[f] = h \sum_{j=0}^{N-1} \left(\sum_{k=-\infty}^{\infty} c_k e^{ikx_j} \right). \tag{1.51}$$

[4] $i^2 = -1$ and if $c = a + ib$, with a, $b \in \mathbb{R}$, then its complex conjugate $\bar{c} = a - ib$.

Justified by the uniform convergence of the series we can exchange the finite and the infinite sums to get

$$T_h[f] = \frac{2\pi}{N} \sum_{k=-\infty}^{\infty} c_k \sum_{j=0}^{N-1} e^{ik\frac{2\pi}{N}j}. \tag{1.52}$$

But

$$\sum_{j=0}^{N-1} e^{ik\frac{2\pi}{N}j} = \sum_{j=0}^{N-1} \left(e^{ik\frac{2\pi}{N}}\right)^j. \tag{1.53}$$

Note that $e^{ik\frac{2\pi}{N}} = 1$ precisely when k is an integer multiple of N, i.e. $k = \ell N$, $\ell \in \mathbb{Z}$ and if so

$$\sum_{j=0}^{N-1} \left(e^{ik\frac{2\pi}{N}}\right)^j = N \qquad \text{for } k = \ell N. \tag{1.54}$$

Otherwise,

$$\sum_{j=0}^{N-1} \left(e^{ik\frac{2\pi}{N}}\right)^j = \frac{1-\left(e^{ik\frac{2\pi}{N}}\right)^N}{1-\left(e^{ik\frac{2\pi}{N}}\right)} = 0 \qquad \text{for } k \neq \ell N. \tag{1.55}$$

Employing (1.54) and (1.55) we thus get that

$$T_h[f] = 2\pi \sum_{\ell=-\infty}^{\infty} c_{\ell N}. \tag{1.56}$$

On the other hand

$$c_0 = \frac{1}{2\pi}\int_0^{2\pi} f(x)dx = \frac{1}{2\pi} I[f]. \tag{1.57}$$

Therefore

$$T_h[f] = I[f] + 2\pi\left[c_N + c_{-N} + c_{2N} + c_{-2N} + \ldots\right], \tag{1.58}$$

that is

$$|T_h[f] - I[f]| \leq 2\pi\left[|c_N| + |c_{-N}| + |c_{2N}| + |c_{-2N}| + \ldots\right]. \tag{1.59}$$

So now, the relevant question is how fast the Fourier coefficients $c_{\ell N}$ of f decay with N. The answer is tied to the smoothness of f. Doing integration by parts in formula (1.50) for the Fourier coefficients of f we have

$$c_k = \frac{1}{2\pi}\frac{1}{ik}\left[\int_0^{2\pi} f'(x)e^{-ikx}dx - f(x)e^{-ikx}\Big|_0^{2\pi}\right] \qquad k \neq 0 \tag{1.60}$$

and the last term vanishes due to the periodicity of $f(x)e^{-ikx}$. Hence,

$$c_k = \frac{1}{2\pi}\frac{1}{ik}\int_0^{2\pi} f'(x)e^{-ikx}dx \qquad k \neq 0. \tag{1.61}$$

Integrating by parts m times we obtain

$$c_k = \frac{1}{2\pi}\left(\frac{1}{ik}\right)^m \int_0^{2\pi} f^{(m)}(x)e^{-ikx}dx \qquad k \neq 0, \tag{1.62}$$

where $f^{(m)}$ is the m-th derivative of f. Therefore, for $f \in C^m[0, 2\pi]$ and periodic

$$|c_k| \leq \frac{A_m}{|k|^m}, \tag{1.63}$$

where A_m is a constant (depending only on m). Using this in (1.59) we get

$$\begin{aligned} |T_h[f] - I[f]| &\leq 2\pi A_m \left[\frac{2}{N^m} + \frac{2}{(2N)^m} + \frac{2}{(3N)^m} + \ldots\right] \\ &= \frac{4\pi A_m}{N^m}\left[1 + \frac{1}{2^m} + \frac{1}{3^m} + \ldots\right], \end{aligned} \tag{1.64}$$

and so for $m > 1$ we can conclude that

$$|T_h[f] - I[f]| \leq \frac{C_m}{N^m}. \tag{1.65}$$

Thus, in this particular case, the rate of convergence of the composite trapezoidal rule at *equally spaced points* is not fixed (to 2). It depends on the number of derivatives of f and we say that the accuracy and convergence of the approximation is *spectral*. Note that if f is smooth, i.e. $f \in C^\infty[0, 2\pi]$ and periodic, the composite trapezoidal rule converges to the exact integral at a rate faster than any power of $1/N$ (or h)! This is called *super-algebraic convergence*.

1.4 Bibliographic Notes

Section 1.1 . In his 1964 textbook, Henrici [Hen64] defines numerical analysis as “the theory of constructive methods in mathematical analysis”. Ralston and Rabinowitz devote a section of their textbook [RR01] to the question What is numerical analysis? They describe numerical analysis as both a science and an art: “As a science, then, numerical analysis is concerned with the processes by which mathematical problems can be solved by the operations of arithmetic. ... As an art, numerical analysis is concerned with choosing that procedure (and suitably applying it) which is “‘best suited”’ to the solution of a particular problem”. In a 1992 article, which inspired this section and this chapter, Trefethen [Tre92] proposes to define numerical analysis as “the study of algorithms for the problems of continuous mathematics”. Gautschi [Gau11], in his excellent graduate textbook, defines numerical analysis as “the branch of mathematics that provides tools and methods for solving mathematical problems in numerical form”.

Section 1.2. The exposition of Richardson’s extrapolation and the explanation of the meaning of sufficiently small h by checking the $q(h)$ ratios was inspired by Section 5.5 (extrapolation to the limit; Romberg integration) in the classical textbook “Elementary Numerical Analysis” [CdB72] by Conte and de Boor. Richardson extrapolation is named after Lewis F. Richardson, who applied the technique to finite differences for partial differential equations [Ric11].

Section 1.3. While the impressive spectral convergence of the composite trapezoidal rule for periodic integrands is well-known, it appears only on relatively few numerical analysis textbooks. One book that does include this important case, and whose presentation inspired this section, is that by Schwarz [Sch89].

Chapter 2

Function Approximation

We saw in the introductory chapter that one key step in the construction of a numerical method to approximate a definite integral is the approximation of the integrand by a simpler function, which we can integrate exactly.

The problem of function approximation is central to many numerical methods. Given a continuous function f in a closed, bounded interval $[a, b]$, we would like to find a good approximation to it by functions from a certain class, for example algebraic polynomials, trigonometric polynomials, rational functions, radial functions, splines, neural networks, etc. We are going to measure the accuracy of an approximation using norms and ask whether or not there is a best approximation out of functions from a given family of functions. These are the main topics of this introductory chapter in approximation theory.

2.1 Norms

A norm on a vector space V over a field $\mathbb{F}$ ($\mathbb{R}$ or $\mathbb{C}$ for our purposes) is a mapping

$$\|\cdot\| : V \to [0, \infty),$$

which satisfy the following properties:

(i) $\|x\| \geq 0 \ \forall x \in V$ and $\|x\| = 0$ iff $x = 0$.

(ii) $\|x + y\| \leq \|x\| + \|y\| \ \forall x, y \in V$.

(iii) $\|\lambda x\| = |\lambda| \, \|x\| \ \forall x \in V, \ \lambda \in \mathbb{F}$.

If we relax (i) to just $\|x\| \geq 0$, we get a *semi-norm.*

We recall first some of the most important examples of norms in the finite dimensional case $V = \mathbb{R}^n$ (or $V = \mathbb{C}^n$):

$$\|x\|_1 = |x_1| + \ldots + |x_n|, \tag{2.1}$$

$$\|x\|_2 = \sqrt{|x_1|^2 + \ldots + |x_n|^2}, \tag{2.2}$$

$$\|x\|_\infty = \max\{|x_1|, \ldots, |x_n|\}. \tag{2.3}$$

These are all special cases of the l^p norm:

$$\|x\|_p = \left(|x_1|^p + \ldots + |x_n|^p\right)^{1/p}, \qquad 1 \leq p \leq \infty. \tag{2.4}$$

If we have weights $w_i > 0$ for $i = 1, \ldots, n$ we can also define a weighted l^p norm by

$$\|x\|_{w,p} = \left(w_1|x_1|^p + \ldots + w_n|x_n|^p\right)^{1/p}, \qquad 1 \leq p \leq \infty. \tag{2.5}$$

All norms in a finite dimensional space V are equivalent, in the sense that for any two norms in V, $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$, there are two constants c and C such that

$$\|x\|_\alpha \leq C\|x\|_\beta, \tag{2.6}$$

$$\|x\|_\beta \leq c\|x\|_\alpha, \tag{2.7}$$

for all $x \in V$.

If V is a space of functions defined on a interval $[a, b]$, for example $C[a, b]$, the corresponding norms to (2.1)-(2.4) are given by

$$\|u\|_1 = \int_a^b |u(x)| dx, \tag{2.8}$$

$$\|u\|_2 = \left(\int_a^b |u(x)|^2 dx\right)^{1/2}, \tag{2.9}$$

$$\|u\|_\infty = \sup_{x \in [a,b]} |u(x)|, \tag{2.10}$$

$$\|u\|_p = \left(\int_a^b |u(x)|^p dx\right)^{1/p}, \qquad 1 \leq p \leq \infty \tag{2.11}$$

and are called the L^1, L^2, L^∞, and L^p norms, respectively. Similarly to (2.5) we can defined a weighted L^p norm by

$$\|u\|_p = \left(\int_a^b w(x)|u(x)|^p dx \right)^{1/p}, \qquad 1 \leq p \leq \infty, \tag{2.12}$$

where w is a given positive weight function defined in $[a, b]$. If $w(x) \geq 0$, we get a semi-norm.

Lemma 2.1.1. *Let $\|\cdot\|$ be a norm on a vector space V then*

$$|\, \|x\| - \|y\| \,| \leq \|x - y\|. \tag{2.13}$$

This lemma implies that a norm is a continuous function (on V to $\mathbb{R}$).

Proof. $\|x\| = \|x - y + y\| \leq \|x - y\| + \|y\|$ which gives that

$$\|x\| - \|y\| \leq \|x - y\|. \tag{2.14}$$

By reversing the roles of x and y we also get

$$\|y\| - \|x\| \leq \|x - y\|. \tag{2.15}$$

□

2.2 Uniform Polynomial Approximation

There is a fundamental result in approximation theory: *any continuous function on a closed, bounded interval can be approximated uniformly, i.e. in the $\|\cdot\|_\infty$ norm, with arbitrary accuracy by a polynomial.* This is the celebrated Weierstrass approximation theorem. We are going to present a constructive proof due to S. Bernstein, which uses a class of polynomials that have found widespread applications in computer graphics and animation. Historically, the use of these so-called Bernstein polynomials in computer assisted design (CAD) was introduced by two engineers working in the French car industry: Pierre Bézier at Renault and Paul de Casteljau at Citroën.

2.2.1 Bernstein Polynomials and Bézier Curves

Given a function f on $[0,1]$, the Bernstein polynomial of degree $n \geq 1$ is defined by

$$B_n f(x) = \sum_{k=0}^{n} f(k/n) \binom{n}{k} x^k (1-x)^{n-k}, \tag{2.16}$$

where

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}, \quad k = 0, \ldots, n \tag{2.17}$$

are the binomial coefficients. Note that $B_n f(0) = f(0)$ and $B_n f(1) = f(1)$ for all n. The terms

$$b_{k,n}(x) = \binom{n}{k} x^k (1-x)^{n-k}, \quad k = 0, \ldots, n, \tag{2.18}$$

which are all nonnegative, are called the Bernstein basis polynomials and can be viewed as x-dependent weights that sum up to one:

$$\sum_{k=0}^{n} b_{k,n}(x) = \sum_{k=0}^{n} \binom{n}{k} x^k (1-x)^{n-k} = [x + (1-x)]^n = 1. \tag{2.19}$$

Thus, for each $x \in [0,1]$, $B_n f(x)$ represents a weighted average of the values of f at $0, 1/n, 2/n, \ldots, 1$. Moreover, as n increases the weights $b_{k,n}(x)$, for $0 < x < 1$, concentrate more and more around the points k/n close to x as Fig. 2.1 indicates for $b_{k,n}(0.5)$.

For $n = 1$, the Bernstein polynomial is just the straight line connecting $f(0)$ and $f(1)$, $B_1 f(x) = (1-x)f(0) + xf(1)$. Given two points $\mathbf{P_0}$ and $\mathbf{P_1}$ in the plane or in space, the segment of the straight line connecting them can be written in parametric form as

$$\mathbf{B}_1(t) = (1-t)\mathbf{P_0} + t\,\mathbf{P_1}, \qquad t \in [0,1]. \tag{2.20}$$

With three points, $\mathbf{P_0}, \mathbf{P_1}, \mathbf{P_2}$, we can employ the quadratic Bernstein basis polynomials to get a more useful parametric curve

$$\mathbf{B}_2(t) = (1-t)^2\mathbf{P_0} + 2t(1-t)\mathbf{P_1} + t^2\,\mathbf{P_2}, \qquad t \in [0,1]. \tag{2.21}$$

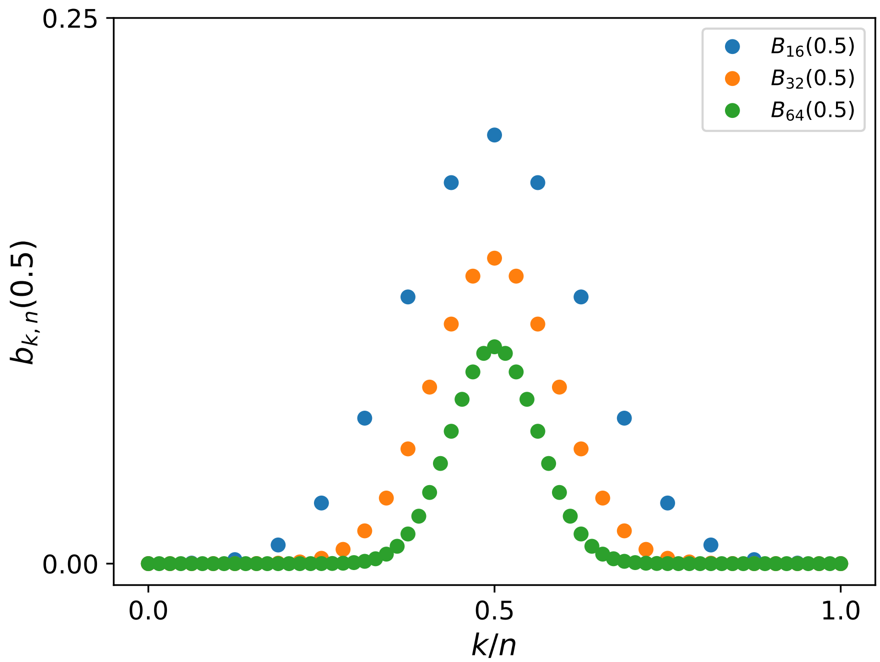

Figure 2.1: The Bernstein basis (weights) $b_{k,n}(x)$ for $x = 0.5$, $n = 16$, 32, and 64. Note how they concentrate more and more around $k/n \approx x$ as n increases.

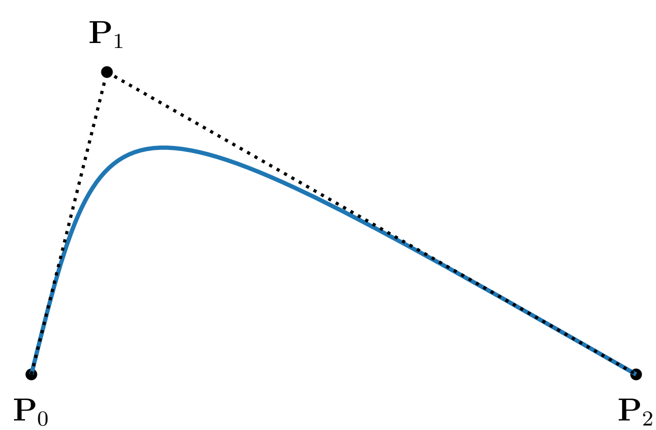

Figure 2.2: Quadratic Bézier curve.

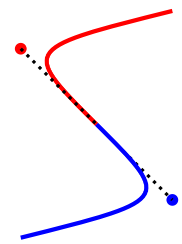

Figure 2.3: Example of a composite, quadratic C^1 Bézier curve with two pieces.

This curve connects again $\mathbf{P_0}$ and $\mathbf{P_2}$ but $\mathbf{P_1}$ can be used to control how the curve bends. More precisely, the tangents at the end points are $\mathbf{B}_2'(0) = 2(\mathbf{P_1} - \mathbf{P_0})$ and $\mathbf{B}_2'(1) = 2(\mathbf{P_2} - \mathbf{P_1})$, which intersect at $\mathbf{P_1}$, as Fig. 2.2 illustrates. These parametric curves formed with the Bernstein basis polynomials are called *Bézier curves* and have been widely employed in computer graphics, specially in the design of vector fonts, and in computer animation. A Bézier curve of degree $n \geq 1$ can be written in parametric form as

$$\mathbf{B}_n(t) = \sum_{k=0}^{n} b_{k,n}(t)\mathbf{P}_k, \qquad t \in [0, 1]. \tag{2.22}$$

The points $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n$ are called control points. Often, low degree (quadratic or cubic) Bézier curves are pieced together to represent complex shapes. These *composite Bézier curves* are broadly used in font generation. For example, the TrueType font of most computers today is generated with composite, quadratic Bézier curves while the Metafont used in these pages, via LaTeX, employs composite, cubic Bézier curves. For each character, many pieces of Bézier curves are stitched together. To have some degree of smoothness (C^1), the common point for two pieces of a composite Bézier curve has to lie on the line connecting the two adjacent control points on either side as Fig. 2.3 shows.

Let us now do some algebra to prove some useful identities of the Bernstein polynomials. First, for $f(x) = x$ we have,

$$\begin{aligned}
\sum_{k=0}^{n} \frac{k}{n} \binom{n}{k} x^k (1-x)^{n-k} &= \sum_{k=1}^{n} \frac{kn!}{n(n-k)!k!} x^k (1-x)^{n-k} \\
&= x \sum_{k=1}^{n} \binom{n-1}{k-1} x^{k-1} (1-x)^{n-k} \\
&= x \sum_{k=0}^{n-1} \binom{n-1}{k} x^k (1-x)^{n-1-k} \\
&= x \left[x + (1-x) \right]^{n-1} = x.
\end{aligned} \tag{2.23}$$

Now for $f(x) = x^2$, we get

$$\sum_{k=0}^{n} \left(\frac{k}{n} \right)^2 \binom{n}{k} x^k (1-x)^{n-k} = \sum_{k=1}^{n} \frac{k}{n} \binom{n-1}{k-1} x^k (1-x)^{n-k} \tag{2.24}$$

and writing

$$\frac{k}{n} = \frac{k-1}{n} + \frac{1}{n} = \frac{n-1}{n} \frac{k-1}{n-1} + \frac{1}{n}, \tag{2.25}$$

we have

$$\begin{aligned}
\sum_{k=0}^{n} \left(\frac{k}{n} \right)^2 \binom{n}{k} x^k (1-x)^{n-k} &= \frac{n-1}{n} \sum_{k=2}^{n} \frac{k-1}{n-1} \binom{n-1}{k-1} x^k (1-x)^{n-k} \\
&\quad + \frac{1}{n} \sum_{k=1}^{n} \binom{n-1}{k-1} x^k (1-x)^{n-k} \\
&= \frac{n-1}{n} \sum_{k=2}^{n} \binom{n-2}{k-2} x^k (1-x)^{n-k} + \frac{x}{n} \\
&= \frac{n-1}{n} x^2 \sum_{k=0}^{n-2} \binom{n-2}{k} x^k (1-x)^{n-2-k} + \frac{x}{n}.
\end{aligned}$$

Thus,

$$\sum_{k=0}^{n} \left(\frac{k}{n} \right)^2 \binom{n}{k} x^k (1-x)^{n-k} = \frac{n-1}{n} x^2 + \frac{x}{n}. \tag{2.26}$$

Now, expanding $\left(\frac{k}{n} - x\right)^2$ and using (2.19), (2.23), and (2.26) it follows that

$$\sum_{k=0}^{n} \left(\frac{k}{n} - x\right)^2 \binom{n}{k} x^k (1-x)^{n-k} = \frac{1}{n} x(1-x). \tag{2.27}$$

2.2.2 Weierstrass Approximation Theorem

Theorem 2.1. *(Weierstrass Approximation Theorem) Let f be a continuous function in a closed, bounded interval $[a, b]$. Given $\epsilon > 0$, there is a polynomial p such that*

$$\max_{a \le x \le b} |f(x) - p(x)| < \epsilon.$$

Proof. We are going to work on the interval $[0, 1]$. For a general interval $[a, b]$, we consider the change of variables $x = a + (b-a)t$ for $t \in [0, 1]$ so that $F(t) = f(a + (b-a)t)$ is continuous in $[0, 1]$.

Using (2.19), we have

$$f(x) - B_n f(x) = \sum_{k=0}^{n} \left[f(x) - f\left(\frac{k}{n}\right) \right] \binom{n}{k} x^k (1-x)^{n-k}. \tag{2.28}$$

Since f is continuous in $[0, 1]$, it is also uniformly continuous. Thus, given $\epsilon > 0$ there is $\delta = \delta(\epsilon) > 0$, independent of x, such that

$$|f(x) - f(k/n)| < \frac{\epsilon}{2} \quad \text{if } |x - k/n| < \delta. \tag{2.29}$$

Moreover,

$$|f(x) - f(k/n)| \le 2\|f\|_\infty \quad \text{for all } x \in [0, 1],\ k = 0, 1, \ldots, n. \tag{2.30}$$

We now split the sum in (2.28) in two sums, one over the points such that $|k/n - x| < \delta$ and the other over the points such that $|k/n - x| \ge \delta$:

$$\begin{aligned} f(x) - B_n f(x) = &\sum_{|k/n - x| < \delta} \left[f(x) - f\left(\frac{k}{n}\right) \right] \binom{n}{k} x^k (1-x)^{n-k} \\ &+ \sum_{|k/n - x| \ge \delta} \left[f(x) - f\left(\frac{k}{n}\right) \right] \binom{n}{k} x^k (1-x)^{n-k}. \end{aligned} \tag{2.31}$$

Using (2.29) and (2.19) it follows immediately that the first sum is bounded by $\epsilon/2$. For the second sum we have

$$
\begin{aligned}
\sum_{|k/n-x|\geq\delta} & \left| f(x) - f\left(\frac{k}{n}\right)\right| \binom{n}{k} x^k(1-x)^{n-k} \\
&\leq 2\|f\|_\infty \sum_{|k/n-x|\geq\delta} \binom{n}{k} x^k(1-x)^{n-k} \\
&\leq \frac{2\|f\|_\infty}{\delta^2} \sum_{|k/n-x|\geq\delta} \left(\frac{k}{n}-x\right)^2 \binom{n}{k} x^k(1-x)^{n-k} \qquad (2.32)\\
&\leq \frac{2\|f\|_\infty}{\delta^2} \sum_{k=0}^{n} \left(\frac{k}{n}-x\right)^2 \binom{n}{k} x^k(1-x)^{n-k} \\
&= \frac{2\|f\|_\infty}{n\delta^2} x(1-x) \leq \frac{\|f\|_\infty}{2n\delta^2}.
\end{aligned}
$$

Therefore, there is N such that for all $n \geq N$ the second sum in (2.31) is bounded by $\epsilon/2$ and this completes the proof. □

Figure 2.4 shows approximations of $f(x) = \sin(2\pi x)$ by Bernstein polynomials of degree $n = 10, 20, 40$. Observe that $\|f - B_n f\|_\infty$ decreases by roughly one half as n is doubled, suggesting a slow $O(1/n)$ convergence even for this smooth function.

2.3 Best Approximation

We just saw that any continuous function f on a closed, bounded interval can be approximated uniformly with arbitrary accuracy by a polynomial. Ideally, we would like to find the closest polynomial, say of degree at most n, to the function f when the distance is measured in the supremum (infinity) norm, or in any other norm we choose. There are three important elements in this general problem: the space of functions we want to approximate, the norm, and the family of approximating functions. The following definition makes this more precise.

Definition 2.1. *Given a normed, vector space V and a subspace W of V, $p^* \in W$ is called a best approximation of $f \in V$ by elements in W if*

$$
\|f - p^*\| \leq \|f - p\|, \quad \textit{for all } p \in W. \qquad (2.33)
$$

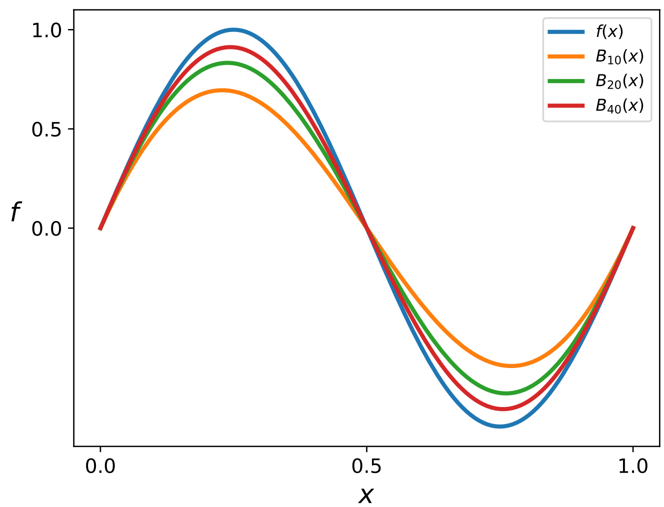

Figure 2.4: Approximation of $f(x) = \sin(2\pi x)$ on $[0, 1]$ by Bernstein polynomials.

For example, the normed, vector space V could be $C[a, b]$ with the supremum norm (2.10) and W could be the set of all polynomials of degree at most n, which henceforth we will denote by $\mathbb{P}_n$.

Theorem 2.2. *Let W be a finite-dimensional subspace of a normed, vector space V. Then, for every $f \in V$, there is at least one best approximation to f by elements in W.*

Proof. Since W is a subspace $0 \in W$ and for any candidate $p \in W$ for best approximation to f we must have

$$\|f - p\| \leq \|f - 0\| = \|f\|. \tag{2.34}$$

Therefore we can restrict our search to the set

$$F = \{p \in W : \|f - p\| \leq \|f\|\}. \tag{2.35}$$

F is closed and bounded and because W is finite-dimensional it follows that F is compact. Now, the function $p \mapsto \|f - p\|$ is continuous on this compact set and hence it attains its minimum in F. $\square$

If we remove the finite-dimensionality of W then we cannot guarantee that there is a best approximation as the following example shows.

Example 2.1. *Let $V = C[0, 1/2]$ and W be the space of all polynomials (clearly a subspace of V). Take $f(x) = 1/(1-x)$ for $x \in [0, 1/2]$ and note that*

$$\frac{1}{1-x} - (1 + x + x^2 + \ldots + x^N) = \frac{x^{N+1}}{1-x}. \tag{2.36}$$

Therefore, given $\epsilon > 0$ there is N such that

$$\max_{x\in[0,1/2]} \left| \frac{1}{1-x} - (1 + x + x^2 + \ldots + x^N) \right| = \left(\frac{1}{2}\right)^N < \epsilon. \tag{2.37}$$

Thus, if there is a best approximation p^* in the supremum norm, necessarily $\|f - p^*\|_\infty = 0$, which implies

$$p^*(x) = \frac{1}{1-x} \tag{2.38}$$

This is of course impossible since p is a polynomial.

Theorem 2.2 does not guarantee uniqueness of best approximation. Strict convexity of the norm gives us a sufficient condition.

Definition 2.2. *A norm $\|\cdot\|$ on a vector space V is strictly convex if for all $f \neq g$ in V with $\|f\| = \|g\| = 1$ then*

$$\|\theta f + (1-\theta)g\| < 1, \quad \textit{for all } 0 < \theta < 1.$$

In other words, a norm is strictly convex if its unit ball is strictly convex.

Note the use of the strict inequality $\|\theta f + (1-\theta)g\| < 1$ in the definition. The p-norm is strictly convex for $1 < p < \infty$ but not for $p = 1$ or $p = \infty$.

Theorem 2.3. *Let V be a vector space with a strictly convex norm, W a subspace of V, and $f \in V$. If p^* and q^* are best approximations of f in W then $p^* = q^*$.*

Proof. Let $M = \|f - p^*\| = \|f - q^*\|$. If $p^* \neq q^*$, by the strict convexity of the norm

$$\left\|\theta\left(\frac{f-p^*}{M}\right) + (1-\theta)\left(\frac{f-q^*}{M}\right)\right\| < 1, \quad \text{for all } 0 < \theta < 1. \tag{2.39}$$

That is,

$$\|\theta(f-p^*) + (1-\theta)(f-q^*)\| < M, \quad \text{for all } 0 < \theta < 1. \tag{2.40}$$

Taking $\theta = 1/2$ we get

$$\|f - \frac{1}{2}(p^* + q^*)\| < M, \tag{2.41}$$

which is impossible because $\frac{1}{2}(p^* + q^*)$ is in W and cannot be a better approximation. □

2.3.1 Best Uniform Polynomial Approximation

Given a continuous function f on a closed, bounded interval $[a, b]$ we know there is at least one best approximation p_n^* to f, in any given norm, by polynomials of degree at most n because the dimension of $\mathbb{P}_n$ is finite. The norm $\|\cdot\|_\infty$ is not strictly convex so Theorem 2.3 does not apply. However, due to a special property (called the *Haar property*) of the vector space $\mathbb{P}_n$, which is that the only element of $\mathbb{P}_n$ that has more than n roots is the zero element, we will see that the best uniform approximation out of $\mathbb{P}_n$ is unique and is characterized by a very peculiar property. Specifically, the error function

$$e_n(x) = f(x) - p_n^*(x), \quad x \in [a, b], \tag{2.42}$$

has to *equioscillate* at least $n+2$ points, between $+\|e_n\|_\infty$ and $-\|e_n\|_\infty$. That is, there are k points, $x_1, x_2, \ldots, x_k$, with $k \geq n+2$, such that

$$\begin{aligned}
e_n(x_1) &= \pm\|e_n\|_\infty \\
e_n(x_2) &= -e_n(x_1), \\
e_n(x_3) &= -e_n(x_2), \\
&\;\;\vdots \\
e_n(x_k) &= -e_n(x_{k-1}).
\end{aligned} \tag{2.43}$$

For if not, it would be possible to find a polynomial of degree at most n, with the same sign at the extremal points of e_n (at most n sign changes), and use this polynomial to decrease the value of $\|e_n\|_\infty$. This would contradict the fact that p_n^* is a best approximation. This is easy to see for $n = 0$ as it is impossible to find a polynomial of degree 0 (a constant) with one change of sign. This is the content of the next result.

Theorem 2.4. *The error $e_n = f - p_n^*$ has at least two extremal points, x_1 and x_2, in $[a, b]$ such that $|e_n(x_1)| = |e_n(x_2)| = \|e_n\|_\infty$ and $e_n(x_1) = -e_n(x_2)$ for all $n \geq 0$.*

Proof. The continuous function $|e_n(x)|$ attains its maximum $\|e_n\|_\infty$ in at least one point x_1 in $[a, b]$. Suppose $\|e_n\|_\infty = e_n(x_1)$ and that $e_n(x) > -\|e_n\|_\infty$ for all $x \in [a, b]$. Then, $m = \min_{x\in[a,b]} e_n(x) > -\|e_n\|_\infty$ and we have some room to decrease $\|e_n\|_\infty$ by shifting down e_n a suitable amount c. In particular, if we take c as one half the gap between the minimum m of e_n and $-\|e_n\|_\infty$,

$$c = \frac{1}{2}\left(m + \|e_n\|_\infty\right) > 0, \tag{2.44}$$

and subtract it to e_n, as shown in Fig. 2.5, we have

$$-\|e_n\|_\infty + c \leq e_n(x) - c \leq \|e_n\|_\infty - c. \tag{2.45}$$

Therefore, $\|e_n - c\|_\infty = \|f - (p_n^* + c)\|_\infty = \|e_n\|_\infty - c < \|e_n\|_\infty$ but $p_n^* + c \in \mathbb{P}_n$ so this is impossible since p_n^* is a best approximation. A similar argument can be used when $e_n(x_1) = -\|e_n\|_\infty$. $\square$

Before proceeding to the general case, let us look at the $n = 1$ situation. Suppose there are only two alternating extremal points x_1 and x_2 for e_1 as described in (2.43). We are going to construct a linear polynomial that has the same sign as e_1 at x_1 and x_2 and which can be used to decrease $\|e_1\|_\infty$. Suppose $e_1(x_1) = \|e_1\|_\infty$ and $e_1(x_2) = -\|e_1\|_\infty$. Since e_1 is continuous, we can find small closed intervals I_1 and I_2, containing x_1 and x_2, respectively, and such that

$$e_1(x) > \frac{\|e_1\|_\infty}{2} \quad \text{for all } x \in I_1, \tag{2.46}$$

$$e_1(x) < -\frac{\|e_1\|_\infty}{2} \quad \text{for all } x \in I_2. \tag{2.47}$$

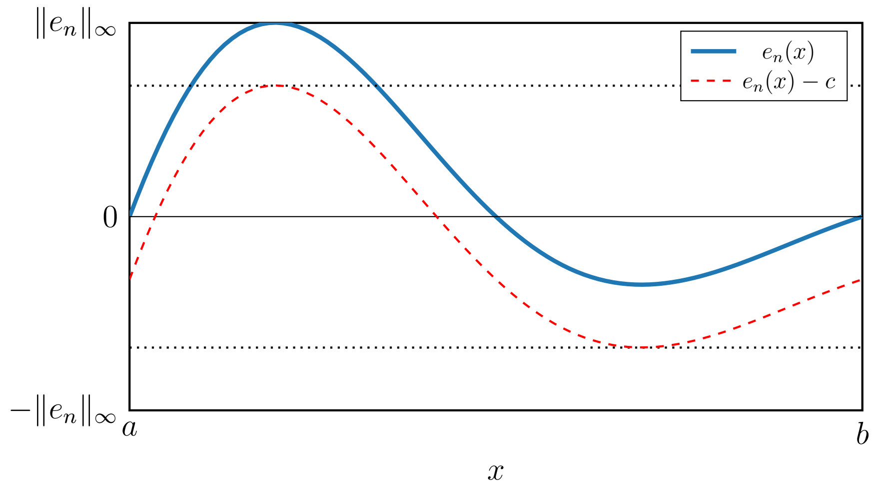

Figure 2.5: If the error function e_n does not equioscillate at least twice we could lower $\|e_n\|_\infty$ by an amount $c > 0$.

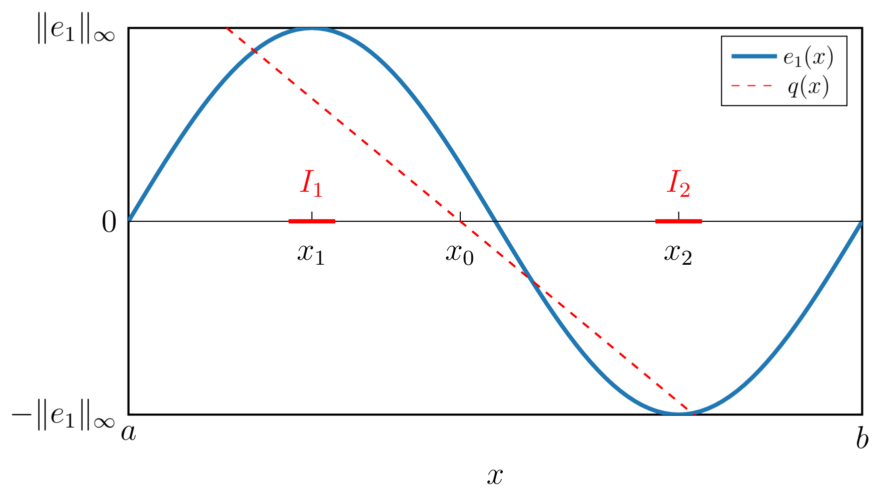

Figure 2.6: If e_1 equioscillates only twice, it would be possible to find a polynomial $q \in \mathbb{P}_1$ with the same sign around x_1 and x_2 as that of e_1 and, after a suitable scaling, use it to decrease the error.

Since I_1 and I_2 are disjoint sets, we can choose a point x_0 between the two intervals. Then, it is possible to find $q \in \mathbb{P}_1$ that passes through x_0 and that is positive in I_1 and negative in I_2 as Fig. 2.6 depicts. We are now going to pick a suitable constant $\alpha > 0$ such that $\|f - p_1^* - \alpha q\|_\infty < \|e_1\|_\infty$. Since $p_1^* + \alpha q \in \mathbb{P}_1$ this would be a contradiction to the fact that p_1^* is a best approximation.

Let $R = [a,b] \setminus (I_1 \cup I_2)$ and $d = \max_{x \in R} |e_1(x)|$. Clearly $d < \|e_1\|_\infty$. Choose α such that

$$0 < \alpha < \frac{1}{2\|q\|_\infty} \left(\|e_1\|_\infty - d\right). \tag{2.48}$$

On I_1, we have

$$0 < \alpha q(x) < \frac{1}{2\|q\|_\infty} \left(\|e_1\|_\infty - d\right) q(x) \leq \frac{1}{2} \left(\|e_1\|_\infty - d\right) < e_1(x). \tag{2.49}$$

Therefore

$$|e_1(x) - \alpha q(x)| = e_1(x) - \alpha q(x) < \|e_1\|_\infty, \quad \text{for all } x \in I_1. \tag{2.50}$$

Similarly, on I_2, we can show that $|e_1(x) - \alpha q(x)| < \|e_1\|_\infty$. Finally, on R we have

$$|e_1(x) - \alpha q(x)| \leq |e_1(x)| + |\alpha q(x)| \leq d + \frac{1}{2} \left(\|e_1\|_\infty - d\right) < \|e_1\|_\infty. \tag{2.51}$$

Therefore, $\|e_1 - \alpha q\|_\infty = \|f - (p_1^* + \alpha q)\|_\infty < \|e_1\|_\infty$, which contradicts the best approximation assumption on p_1^*.

Theorem 2.5. *(Chebyshev Equioscillation Theorem) Let $f \in C[a,b]$. Then, p_n^* in $\mathbb{P}_n$ is a best uniform approximation of f if and only if there are at least $n+2$ points in $[a,b]$, where the error $e_n = f - p_n^*$ equioscillates between the values $\pm\|e_n\|_\infty$ as defined in (2.43).*

Proof. We first prove that if the error $e_n = f - p_n^*$, for some $p_n^* \in \mathbb{P}_n$, equioscillates at least $n+2$ times then p_n^* is a best approximation. Suppose the contrary. Then, there is $q_n \in \mathbb{P}_n$ such that

$$\|f - q_n\|_\infty < \|f - p_n^*\|_\infty. \tag{2.52}$$

Let $x_1, \ldots, x_k$, with $k \geq n+2$, be the points where e_n equioscillates. Then

$$|f(x_j) - q_n(x_j)| < |f(x_j) - p_n^*(x_j)|, \quad j = 1, \ldots, k \tag{2.53}$$

and since

$$f(x_j) - p_n^*(x_j) = -[f(x_{j+1}) - p_n^*(x_{j+1})], \quad j = 1, \ldots, k-1 \tag{2.54}$$

we have that

$$q_n(x_j) - p_n^*(x_j) = f(x_j) - p_n^*(x_j) - [f(x_j) - q_n(x_j)] \tag{2.55}$$

changes signs $k-1$ times, i.e. at least $n+1$ times. But $q_n - p_n^* \in \mathbb{P}_n$. Therefore $q_n = p_n^*$, which contradicts (2.52), and consequently p_n^* has to be a best uniform approximation of f.

For the other half of the proof the idea is the same as for $n = 1$ but we need to do more bookkeeping. We are going to partition $[a, b]$ into the union of sufficiently small subintervals so that we can guarantee that $|e_n(t) - e_n(s)| \leq \|e_n\|_\infty/2$ for any two points t and s in each of the subintervals. Let us label by $I_1, \ldots, I_k$, the subintervals on which $|e_n(x)|$ achieves its maximum $\|e_n\|_\infty$. Then, on each of these subintervals either $e_n(x) > \|e_n\|_\infty/2$ or $e_n(x) < -\|e_n\|_\infty/2$. We need to prove that e_n changes sign at least $n+1$ times.

Going from left to right, we can label the subintervals $I_1, \ldots, I_k$ as a $(+)$ or $(-)$ subinterval depending on the sign of e_n. For definiteness, suppose I_1 is a $(+)$ subinterval then we have the groups

$$\begin{array}{ll} \{I_1, \ldots, I_{k_1}\}, & (+) \\ \{I_{k_1+1}, \ldots, I_{k_2}\}, & (-) \\ \vdots & \\ \{I_{k_m+1}, \ldots, I_k\}, & (-)^m. \end{array}$$

We have m changes of sign so let us assume that $m \leq n$. We already know $m \geq 1$. Since the sets, I_{k_j} and I_{k_j+1} are disjoint for $j = 1, \ldots, m$, we can select points $t_1, \ldots, t_m$, such that $t_j > x$ for all $x \in I_{k_j}$ and $t_j < x$ for all $x \in I_{k_j+1}$. Then, the polynomial

$$q(x) = (t_1 - x)(t_2 - x) \cdots (t_m - x) \tag{2.56}$$

has the same sign as e_n in each of the extremal intervals $I_1, \ldots, I_k$ and $q \in \mathbb{P}_n$. The rest of the proof is as in the $n = 1$ case to show that $p_n^* + \alpha q$ would be a better approximation to f than p_n^*. $\square$

Theorem 2.6. *Let $f \in C[a,b]$. The best uniform approximation p_n^* to f by elements of $\mathbb{P}_n$ is unique.*

Proof. Suppose q_n^* is also a best approximation, i.e.

$$\|e_n\|_\infty = \|f - p_n^*\|_\infty = \|f - q_n^*\|_\infty.$$

Then, the midpoint $r = \frac{1}{2}(p_n^* + q_n^*)$ is also a best approximation, for $r \in \mathbb{P}_n$ and

$$\begin{aligned} \|f - r\|_\infty &= \|\frac{1}{2}(f - p_n^*) + \frac{1}{2}(f - q_n^*)\|_\infty \\ &\leq \frac{1}{2}\|f - p_n^*\|_\infty + \frac{1}{2}\|f - q_n^*\|_\infty = \|e_n\|_\infty. \end{aligned} \tag{2.57}$$

Let $x_1, \ldots, x_{n+2}$ be extremal points of $f - r$ with the alternating property (2.43), i.e. $f(x_j) - r(x_j) = (-1)^{m+j}\ \|e_n\|_\infty$ for some integer m and $j = 1, \ldots n+2$. This implies that

$$\frac{f(x_j) - p_n^*(x_j)}{2} + \frac{f(x_j) - q_n^*(x_j)}{2} = (-1)^{m+j}\ \|e_n\|_\infty, \quad j = 1, \ldots, n+2. \tag{2.58}$$

But $|f(x_j) - p_n^*(x_j)| \leq \|e_n\|_\infty$ and $|f(x_j) - q_n^*(x_j)| \leq \|e_n\|_\infty$. As a consequence,

$$f(x_j) - p_n^*(x_j) = f(x_j) - q_n^*(x_j) = (-1)^{m+j}\ \|e_n\|_\infty, \quad j = 1, \ldots, n+2, \tag{2.59}$$

and it follows that

$$p_n^*(x_j) = q_n^*(x_j), \quad j = 1, \ldots, n+2. \tag{2.60}$$

Therefore, $q_n^* = p_n^*$. □

2.4 Chebyshev Polynomials

The best uniform approximation of $f(x) = x^{n+1}$ in $[-1,1]$ by polynomials of degree at most n can be found explicitly and the solution introduces one of the most useful and remarkable polynomials, the Chebyshev polynomials.

Let $p_n^* \in \mathbb{P}_n$ be the best uniform approximation to x^{n+1} in the interval $[-1, 1]$ and as before define the error function as $e_n(x) = x^{n+1} - p_n^*(x)$. Note that since e_n is a monic polynomial (its leading coefficient is 1) of degree $n+1$, the problem of finding p_n^* is equivalent to finding, among all monic polynomials of degree $n+1$, the one with the smallest deviation (in absolute value) from zero in $[-1, 1]$.

According to Theorem 2.5, there exist $n+2$ distinct points,

$$-1 \leq x_1 < x_2 < \cdots < x_{n+2} \leq 1, \tag{2.61}$$

such that

$$e_n^2(x_j) = \|e_n\|_\infty^2, \quad \text{for } j = 1, \ldots, n+2. \tag{2.62}$$

Now consider the polynomial

$$q(x) = \|e_n\|_\infty^2 - e_n^2(x). \tag{2.63}$$

Then, $q(x_j) = 0$ for $j = 1, \ldots, n+2$. Each of the points x_j in the interior of $[-1, 1]$ is also a local minimum of q, then necessarily $q'(x_j) = 0$ for $j = 2, \ldots n+1$. Thus, the n points $x_2, \ldots, x_{n+1}$ are zeros of q of multiplicity at least two. But q is a nonzero polynomial of degree $2n+2$ exactly. Therefore, x_1 and x_{n+2} have to be simple zeros and so $x_1 = -1$ and $x_{n+2} = 1$. Note that the polynomial $p(x) = (1 - x^2)[e_n'(x)]^2 \in \mathbb{P}_{2n+2}$ has the same zeros as q and so $p = cq$, for some constant c. Comparing the coefficient of the leading order term of p and q it follows that $c = (n+1)^2$. Therefore, e_n satisfies the ordinary differential equation

$$(1 - x^2)[e_n'(x)]^2 = (n+1)^2 \left[\|e_n\|_\infty^2 - e_n^2(x)\right]. \tag{2.64}$$

We know $e_n' \in \mathbb{P}_n$ and its n zeros are the interior points $x_2, \ldots, x_{n+1}$. Therefore, e_n' cannot change sign in $[-1, x_2]$. Suppose it is nonnegative for $x \in [-1, x_2]$ (we reach the same conclusion if we assume $e_n'(x) \leq 0$) then, taking square roots in (2.64) we get

$$\frac{e_n'(x)}{\sqrt{\|e_n\|_\infty^2 - e_n^2(x)}} = \frac{n+1}{\sqrt{1 - x^2}}, \quad \text{for } x \in [-1, x_2]. \tag{2.65}$$

We can integrate this ordinary differential equation using the trigonometric substitutions $e_n(x) = \|e_n\|_\infty \cos\phi$ and $x = \cos\theta$, for the left and the right hand side respectively, to obtain

$$-\cos^{-1}\left(\frac{e_n(x)}{\|e_n\|_\infty}\right) = -(n+1)\theta + C, \tag{2.66}$$

where C is a constant of integration. Choosing $C = 0$ (so that $e_n(1) = \|e_n\|_\infty$) we get

$$e_n(x) = \|e_n\|_\infty \cos\left[(n+1)\theta\right] \tag{2.67}$$

for $x = \cos\theta \in [-1, x_2]$ with $0 < \theta \leq \pi$. Recall that e_n is a polynomial of degree $n+1$ then so is $\cos[(n+1)\cos^{-1} x]$. Since these two polynomials agree in $[-1, x_2]$, (2.67) must also hold for all x in $[-1, 1]$.

Definition 2.3. *The Chebyshev polynomial (of the first kind) of degree n, T_n is defined by*

$$T_n(x) = \cos n\theta, \quad x = \cos\theta, \quad 0 \leq \theta \leq \pi. \tag{2.68}$$

Note that (2.68) only defines T_n for $x \in [-1, 1]$. However, once the coefficients of this polynomial are determined we can define it for any real (or complex) x.

Using the trigonometry identity

$$\cos(n+1)\theta + \cos(n-1)\theta = 2\cos n\theta \cos\theta, \tag{2.69}$$

we immediately get

$$T_{n+1}(\cos\theta) + T_{n-1}(\cos\theta) = 2T_n(\cos\theta)\cdot\cos\theta \tag{2.70}$$

and going back to the x variable we obtain the recursion formula

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x), \quad n \geq 1, \end{aligned} \tag{2.71}$$

which makes it more evident the T_n for $n = 0, 1, \ldots$ are indeed polynomials of exactly degree n. Let us generate a few of them.

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_2(x) &= 2x\cdot x - 1 = 2x^2 - 1, \\ T_3(x) &= 2x\cdot(2x^2-1) - x = 4x^3 - 3x, \\ T_4(x) &= 2x(4x^3-3x) - (2x^2-1) = 8x^4 - 8x^2 + 1 \\ T_5(x) &= 2x(8x^4-8x^2+1) - (4x^3-3x) = 16x^5 - 20x^3 + 5x. \end{aligned} \tag{2.72}$$

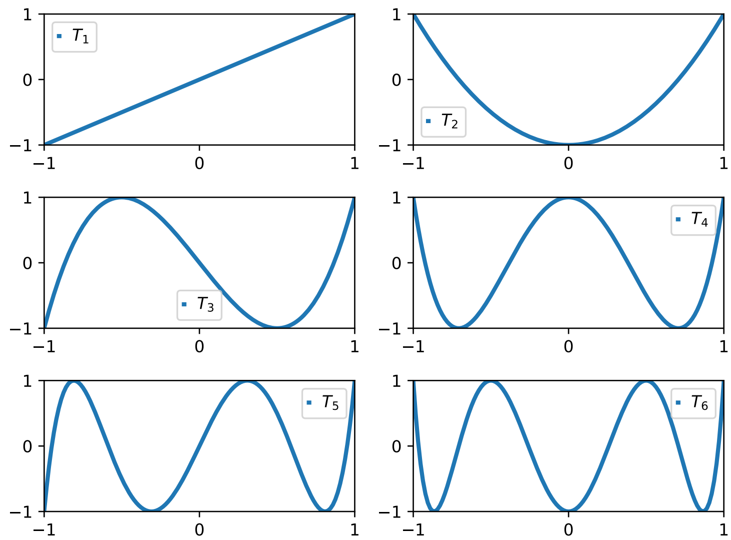

Figure 2.7: The Chebyshev polynomials T_n for $n = 1, 2, 3, 4, 5, 6$.

From these few Chebyshev polynomials, and from (2.71), we see that

$$T_n(x) = 2^{n-1}x^n + \text{lower order terms} \tag{2.73}$$

and that T_n is an even (odd) function of x if n is even (odd), i.e.

$$T_n(-x) = (-1)^n T_n(x). \tag{2.74}$$

The Chebyshev polynomials T_n, for $n = 1, 2, \ldots, 6$ are plotted in Fig. 2.7. Going back to (2.67), since the leading order coefficient of e_n is 1 and that of T_{n+1} is 2^n, it follows that $\|e_n\|_\infty = 2^{-n}$. Therefore

$$p_n^*(x) = x^{n+1} - \frac{1}{2^n}T_{n+1}(x) \tag{2.75}$$

is the best uniform approximation of x^{n+1} in $[-1, 1]$ by polynomials of degree at most n. Equivalently, as noted in the beginning of this section, the monic polynomial of degree n with smallest supremum norm in $[-1, 1]$ is

$$\tilde{T}_n(x) = \frac{1}{2^{n-1}}T_n(x). \tag{2.76}$$

Hence, for any other monic polynomial p of degree n

$$\max_{x\in[-1,1]} |p(x)| > \frac{1}{2^{n-1}}. \tag{2.77}$$

The zeros and extremal points of T_n are easy to find. Because $T_n(x) = \cos n\theta$ and $0 \leq \theta \leq \pi$, the zeros occur when θ is an odd multiple of $\pi/2$. Therefore,

$$\bar{x}_j = \cos\left(\frac{(2j+1)}{n}\frac{\pi}{2}\right) \quad j = 0, \ldots, n-1 \tag{2.78}$$

are the zeros of T_n.

The extremal points of T_n (the points x where $T_n(x) = \pm 1$) correspond to $n\theta = j\pi$ for $j = 0, 1, \ldots, n$, that is

$$x_j = \cos\left(\frac{j\pi}{n}\right), \quad j = 0, 1, \ldots, n. \tag{2.79}$$

These points are called Chebyshev, Chebyshev-Lobatto, or Gauss-Lobatto points and are very useful in applications. We will simply call them Chebyshev points or Chebyshev nodes. Figure 2.8 shows the Chebyshev nodes for $n = 16$. Note that they are more clustered at the end points of the interval and that x_j for $j = 1, \ldots, n-1$ are local extremal points. Therefore

$$T_n'(x_j) = 0, \quad \text{for } j = 1, \ldots, n-1. \tag{2.80}$$

In other words, the Chebyshev points (2.79) are the $n-1$ zeros of T_n' plus the end points $x_0 = 1$ and $x_n = -1$.

Using the Chain Rule we can differentiate T_n with respect to x:

$$T_n'(x) = -n \sin n\theta \frac{d\theta}{dx} = n\frac{\sin n\theta}{\sin\theta}, \quad (x = \cos\theta). \tag{2.81}$$

Therefore

$$\frac{T_{n+1}'(x)}{n+1} - \frac{T_{n-1}'(x)}{n-1} = \frac{1}{\sin\theta}\left[\sin(n+1)\theta - \sin(n-1)\theta\right] \tag{2.82}$$

and since $\sin(n+1)\theta - \sin(n-1)\theta = 2\sin\theta\cos n\theta$, we get that

$$\frac{T_{n+1}'(x)}{n+1} - \frac{T_{n-1}'(x)}{n-1} = 2T_n(x). \tag{2.83}$$

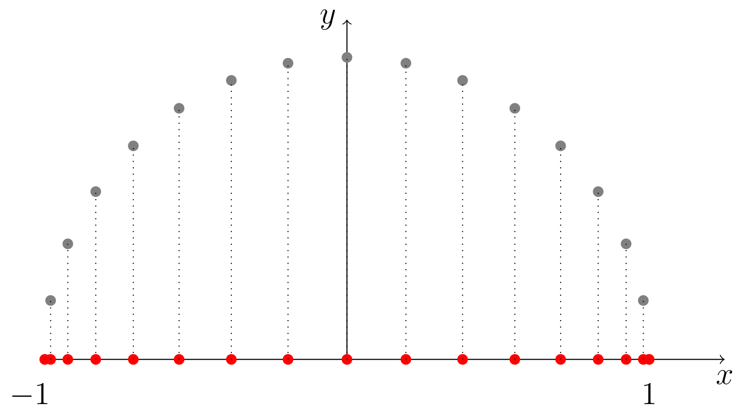

Figure 2.8: The Chebyshev nodes (red dots) $x_j = \cos(j\pi/n)$, $j = 0, 1, \ldots, n$ for $n = 16$. The gray dots on the semi-circle correspond to the equispaced angles $\theta_j = j\pi/n$, $j = 0, 1, \ldots, n$.

The polynomial

$$U_n(x) = \frac{T'_{n+1}(x)}{n+1} = \frac{\sin(n+1)\theta}{\sin\theta}, \quad (x = \cos\theta) \tag{2.84}$$

is called the second kind Chebyshev polynomial of degree n. Thus, the Chebyshev nodes (2.79) are the zeros of the polynomial

$$q_{n+1}(x) = (1 - x^2)U_{n-1}(x). \tag{2.85}$$

2.5 Bibliographic Notes

Section 2.1. A simple proof that all norms on a finite dimensional, vector space are equivalent can be found in [Hac94], Section 2.6.

Section 2.2. A historical account of the invention of Bezier curves and surfaces used in CAD is given by G. Farin [Far02]. The excellent book on approximation theory by Rivlin [Riv81] contains Berstein's proof of Weierstrass theorem. Other fine textbooks on approximation theory that are the main sources for this chapter and the next one are the classical books by Cheney [Che82] and Davis [Dav75]. There are many proofs of Weierstrass approximation theorem. One of great simplicity, due to H. Lebesgue, is masterfully presented by de la Vallée Poussin in his lectures on function approximation [dLVP19].

Section 2.3. This section follows the material on best approximation in [Riv81] (Introduction and Chapter 1) and in [Dav75] (Chapter 7). Example 2.1 is from Rivlin's book [Riv81].

Section 2.4. The construction of the solution to the best uniform approximation of x^{n+1} by polynomials of degree at most n, or equivalently the polynomial of degree $\leq n$ that deviates the least from zero, is given in [Riv81, Tim94]. In particular, Timan [Tim94] points out that Chebyshev arrived at his equi-oscillation theorem by considering this particular problem. An excellent reference for Chebyshev polynomials is the monograph by Rivlin [Riv20].

Chapter 3

Interpolation

One of the most useful tools for approximating a function or a given data set is interpolation, where the approximating function is required to coincide with a give set of values. In this chapter, we focus on (algebraic) polynomial and piece-wise polynomial interpolation (splines), and trigonometric interpolation.

3.1 Polynomial Interpolation

The polynomial *interpolation problem* can be stated as follows: Given $n+1$ data points, $(x_0, f_0), (x_1, f_1) ..., (x_n, f_n)$, where $x_0, x_1, \ldots, x_n$ are distinct, find a polynomial $p_n \in \mathbb{P}_n$, which satisfies the interpolation conditions:

$$\begin{aligned} p_n(x_0) &= f_0, \\ p_n(x_1) &= f_1, \\ &\vdots \\ p_n(x_n) &= f_n. \end{aligned}$$

The points $x_0, x_1, \ldots, x_n$ are called interpolation *nodes* and the values $f_0, f_1, \ldots, f_n$ are data supplied to us or they can come from a function f we would like to approximate, in which case $f_j = f(x_j)$ for $j = 0, 1, \ldots, n$. Figure 3.1 illustrates the interpolation problem for $n = 6$.

Let us represent such polynomial as $p_n(x) = a_0 + a_1 x + \cdots + a_n x^n$. Then, the interpolation conditions imply

$$a_0 + a_1 x_0 + \cdots + a_n x_0^n = f_0,$$

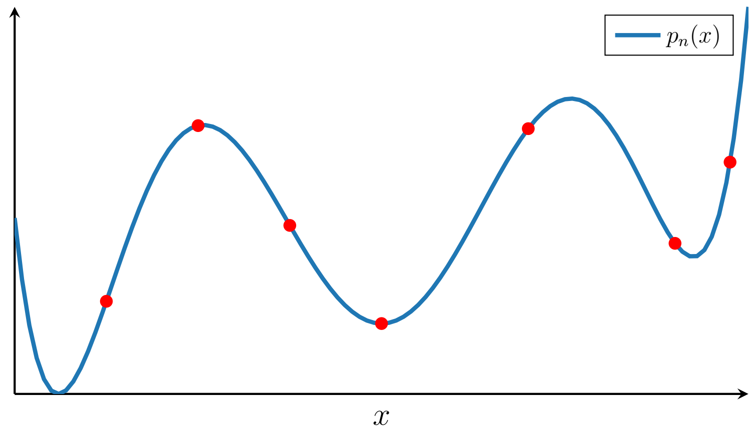

Figure 3.1: Given the data points $(x_0, f_0), \ldots, (x_n, f_n)$ ($\bullet$, $n = 6$), the polynomial interpolation problem consists in finding a polynomial $p_n \in \mathbb{P}_n$ such that $p_n(x_j) = f_j$, for $j = 0, 1, \ldots, n$.

$$
\begin{gathered}
a_0 + a_1 x_1 + \cdots + a_n x_1^n = f_1, \\
\vdots \\
a_0 + a_1 x_n + \cdots + a_n x_n^n = f_n.
\end{gathered}
$$

This is a linear system of $n+1$ equations in $n+1$ unknowns (the polynomial coefficients $a_0, a_1, \ldots, a_n$). In matrix form:

$$
\begin{bmatrix}
1 & x_0 & x_0^2 \cdots x_0^n \\
1 & x_1 & x_1^2 \cdots x_1^n \\
\vdots & & \\
1 & x_n & x_n^2 \cdots x_n^n
\end{bmatrix}
\begin{bmatrix}
a_0 \\ a_1 \\ \vdots \\ a_n
\end{bmatrix}
=
\begin{bmatrix}
f_0 \\ f_1 \\ \vdots \\ f_n.
\end{bmatrix}
\tag{3.1}
$$

Does this linear system have a solution? Is this solution unique? The answer is yes to both. Here is a simple proof. Take $f_j = 0$ for $j = 0, 1, \ldots, n$. Then $p_n(x_j) = 0$, for $j = 0, 1, ..., n$ but p_n is a polynomial of degree at most n, it cannot have $n+1$ zeros unless $p_n \equiv 0$, which implies $a_0 = a_1 = \cdots = a_n = 0$. That is, the homogenous problem associated with (3.1) has only the trivial solution. Therefore, (3.1) has a unique solution.

Example 3.1. *As an illustration let us consider interpolation by a polynomial* $p_1 \in \mathbb{P}_1$. *Suppose we are given* (x_0, f_0) *and* (x_1, f_1) *with* $x_0 \neq x_1$. *We* *wrote* p_1 *explicitly in* (1.2) *[with* $x_0 = a$ *and* $x_1 = b$*]. We write it now in a different form:*

$$p_1(x) = \left(\frac{x - x_1}{x_0 - x_1}\right) f_0 + \left(\frac{x - x_0}{x_1 - x_0}\right) f_1. \tag{3.2}$$

Clearly, this polynomial has degree at most 1 and satisfies the interpolation conditions:

$$p_1(x_0) = f_0, \tag{3.3}$$
$$p_1(x_1) = f_1. \tag{3.4}$$

Example 3.2. *Given* (x_0, f_0), (x_1, f_1), *and* (x_2, f_2), *with* x_0, x_1 *and* x_2 *distinct, let us construct* $p_2 \in \mathbb{P}_2$ *that interpolates these points. The form we have used for* p_1 *in (3.2) is suggestive of how we can write* p_2*:*

$$p_2(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f_0 + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f_1 + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f_2.$$

If we define

$$l_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}, \tag{3.5}$$
$$l_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}, \tag{3.6}$$
$$l_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}, \tag{3.7}$$

then we simply have

$$p_2(x) = l_0(x) f_0 + l_1(x) f_1 + l_2(x) f_2. \tag{3.8}$$

Note that each of the polynomials (3.5), (3.6), and (3.7) are exactly of degree 2 and they satisfy $l_j(x_k) = \delta_{jk}$ [1]*. Therefore, it follows that* p_2 *given by (3.8) satisfies the desired interpolation conditions:*

$$\begin{aligned} p_2(x_0) &= f_0, \\ p_2(x_1) &= f_1, \\ p_2(x_2) &= f_2. \end{aligned} \tag{3.9}$$

[1] δ_{jk} is the Kronecker delta, i.e. $\delta_{jk} = 0$ if $k \neq j$ and 1 if $k = j$.

We can now write down the polynomial p_n of degree at most n that interpolates $n+1$ given values, $(x_0, f_0), \ldots, (x_n, f_n)$, where the interpolation nodes $x_0, \ldots, x_n$ are assumed distinct. Define

$$\begin{aligned} l_j(x) &= \frac{(x-x_0)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n)}{(x_j-x_0)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)} \\ &= \prod_{\substack{k=0\\k\neq j}}^{n} \frac{(x-x_k)}{(x_j-x_k)}, \qquad \text{for } j=0,1,...,n. \end{aligned} \tag{3.10}$$

These polynomials are called (polynomial) cardinal functions or fundamental polynomials of degree n. For simplicity, we are omitting in the notation their dependence on the $n+1$ nodes $x_0, x_1, \ldots, x_n$. Since $l_j(x_k) = \delta_{jk}$,

$$p_n(x) = l_0(x)f_0 + l_1(x)f_1 + \cdots + l_n(x)f_n = \sum_{j=0}^{n} l_j(x)f_j \tag{3.11}$$

interpolates the given data, i.e., it satisfies $p_n(x_j) = f_j$ for $j = 0, 1, 2, \ldots, n$. Relation (3.11) is called the *Lagrange form* of the interpolating polynomial.

The following result summarizes our discussion.

Theorem 3.1. *Given the $n+1$ values $(x_0, f_0), \ldots, (x_n, f_n)$, for $x_0, x_1, ..., x_n$ distinct, there is a unique polynomial p_n of degree at most n such that $p_n(x_j) = f_j$ for $j = 0, 1, \ldots, n$.*

Proof. p_n in (3.11) is of degree at most n and interpolates the data. Uniqueness follows from the fundamental theorem of algebra, as noted earlier. Suppose there is another polynomial q_n of degree at most n such that $q_n(x_j) = f_j$ for $j = 0, 1, \ldots, n$. Consider $r = p_n - q_n$. This is a polynomial of degree at most n and $r(x_j) = p_n(x_j) - q_n(x_j) = f_j - f_j = 0$ for $j = 0, 1, 2, \ldots, n$, which is impossible unless $r \equiv 0$. This implies $q_n = p_n$. □

3.1.1 Equispaced and Chebyshev Nodes

There are two special sets of nodes that are particularly important in applications. The uniform or equispaced nodes in an interval $[a, b]$ are given by

$$x_j = a + jh, \quad j = 0, 1, \ldots, n \;\; \text{with } h = (b-a)/n. \tag{3.12}$$

These nodes yield very accurate and efficient *trigonometric* polynomial interpolation but are generally not good for (algebraic) polynomial interpolation as we will see later.

One of the preferred set of nodes for high order, accurate, and computationally efficient polynomial interpolation is the *Chebyshev* nodes, introduced in Section 2.4. In $[-1, 1]$, they are given by

$$x_j = \cos\left(\frac{j\pi}{n}\right), \qquad j = 0, \ldots, n, \tag{3.13}$$

and are the extremal points of the Chebyshev polynomial (2.68) of degree n. Note that these nodes are obtained from the equispaced points $\theta_j = j(\pi/n)$, $j = 0, 1, \ldots, n$ in $[0, \pi]$ by the one-to-one relation $x = \cos\theta$, for $\theta \in [0, \pi]$. As defined in (3.13), the nodes go from 1 to -1 so sometimes the alternative definition $x_j = -\cos(j\pi/n)$ is used. The Chebyshev nodes are not equally spaced and tend to cluster toward the end points of the interval (see Fig. 2.8). For a general interval $[a, b]$, we can do the simple change of variables

$$x = \frac{1}{2}(a+b) + \frac{1}{2}(b-a)t, \quad t \in [-1, 1], \tag{3.14}$$

to obtain the corresponding Chebyshev nodes in $[a, b]$.

3.2 Connection to Best Uniform Approximation

Given a continuous function f in $[a, b]$, its best uniform approximation p_n^* in $\mathbb{P}_n$ is characterized by an error, $e_n = f - p_n^*$, which equioscillates, as defined in (2.43), at least $n+2$ times. Therefore e_n has a minimum of $n+1$ zeros and consequently, there exists $x_0, \ldots, x_n$ such that

$$\begin{aligned} p_n^*(x_0) &= f(x_0), \\ p_n^*(x_1) &= f(x_1), \\ &\vdots \\ p_n^*(x_n) &= f(x_n). \end{aligned} \tag{3.15}$$

In other words, p_n^* is the polynomial of degree at most n that interpolates the function f at $n+1$ zeros of e_n. Rather than finding these zeros, a natural and more practical question is: given $(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))$, where $x_0, \ldots, x_n$ in $[a, b]$ are distinct, how close is the interpolating polynomial $p_n \in \mathbb{P}_n$ of f at these nodes to the best uniform approximation $p_n^* \in \mathbb{P}_n$ of f?

To obtain a bound for $\|p_n - p_n^*\|_\infty$ we note that $p_n - p_n^*$ is a polynomial of degree at most n which interpolates $f - p_n^*$. Therefore, we can use Lagrange formula to represent it:

$$p_n(x) - p_n^*(x) = \sum_{j=0}^{n} l_j(x)[f(x_j) - p_n^*(x_j)]. \tag{3.16}$$

It then follows that

$$\|p_n - p_n^*\|_\infty \leq \Lambda_n \|f - p_n^*\|_\infty, \tag{3.17}$$

where

$$\Lambda_n = \max_{a \leq x \leq b} \sum_{j=0}^{n} |l_j(x)| \tag{3.18}$$

is called the *Lebesgue constant* and depends only on the interpolation nodes, not on f. On the other hand, we have that

$$\|f - p_n\|_\infty = \|f - p_n^* - p_n + p_n^*\|_\infty \leq \|f - p_n^*\|_\infty + \|p_n - p_n^*\|_\infty. \tag{3.19}$$

Using (3.17) we obtain

$$\|f - p_n\|_\infty \leq (1 + \Lambda_n)\|f - p_n^*\|_\infty. \tag{3.20}$$

This inequality connects the interpolation error $\|f - p_n\|_\infty$ with the best approximation error $\|f - p_n^*\|_\infty$. What happens to these errors as we increase n? To make it more concrete, suppose we have a triangular array of nodes as follows:

$$\begin{array}{llll}
x_0^{(0)} & & & \\
x_0^{(1)} & x_1^{(1)} & & \\
x_0^{(2)} & x_1^{(2)} & x_2^{(2)} & \\
\vdots & & & \\
x_0^{(n)} & x_1^{(n)} & \ldots & x_n^{(n)} \\
\vdots & & &
\end{array} \tag{3.21}$$

where $a \leq x_0^{(n)} < x_1^{(n)} < \cdots < x_n^{(n)} \leq b$ for $n = 0, 1, \ldots$. Let p_n be the interpolating polynomial of degree at most n of f at the nodes corresponding to the $n+1$ row of (3.21). By the Weierstrass Approximation Theorem (p_n^* is a better approximation or at least as good as that provided by the Bernstein polynomial),

$$\|f - p_n^*\|_\infty \to 0 \quad \text{as } n \to \infty. \tag{3.22}$$

However, it can be proved that

$$\Lambda_n > \frac{2}{\pi^2} \log n - 1 \tag{3.23}$$

and hence the Lebesgue constant is not bounded in n. Therefore, we cannot conclude from (3.20) and (3.22) that $\|f - p_n\|_\infty \to 0$ as $n \to \infty$, i.e. that the interpolating polynomial, as we add more and more nodes, converges uniformly to f. In fact, given any distribution of points, organized in a triangular array (3.21), it is possible to construct a continuous function f for which its interpolating polynomial p_n (corresponding to the nodes on the n-th row of (3.21)) will not converge uniformly to f as $n \to \infty$.

Convergence of polynomial interpolation depends on both *the regularity of f and the distribution of the interpolation nodes*. We will discuss this further in Section 3.8.

3.3 Barycentric Formula

The Lagrange form of the interpolating polynomial

$$p_n(x) = \sum_{j=0}^{n} l_j(x) f_j$$

is not convenient for computations. The evaluation of each l_j costs $O(n)$ operations and there are n of these evaluations for a total cost of $O(n^2)$ operations. Also, if we want to increase the degree of the polynomial we cannot reuse the work done in getting and evaluating a lower degree one. However, we can obtain a more efficient formula by rewriting the interpolating polynomial in the following way. Let

$$\omega(x) = (x - x_0)(x - x_1) \cdots (x - x_n). \tag{3.24}$$

Then, differentiating this polynomial of degree $n+1$ and evaluating at $x = x_j$ we get

$$\omega'(x_j) = \prod_{\substack{k=0 \\ k\neq j}}^{n} (x_j - x_k), \quad \text{for } j = 0, 1, \ldots, n. \tag{3.25}$$

Therefore, each of the fundamental polynomials may be written as

$$l_j(x) = \frac{\dfrac{\omega(x)}{x - x_j}}{\omega'(x_j)} = \frac{\omega(x)}{(x - x_j)\omega'(x_j)}, \tag{3.26}$$

for $x \neq x_j$, $j = 0, 1, \ldots, n$ and $l_j(x_j) = 1$ follows from L'Hôpital rule.

Defining

$$\lambda_j = \frac{1}{\omega'(x_j)}, \quad \text{for } j = 0, 1, \ldots, n, \tag{3.27}$$

we can recast Lagrange formula as

$$p_n(x) = \omega(x) \sum_{j=0}^{n} \frac{\lambda_j}{x - x_j} f_j. \tag{3.28}$$

This *modified Lagrange formula* is computationally more efficient than the original formula if we need to evaluate p_n at more than one point. This is because the λ_j's only depend on the interpolation nodes and can be precomputed for a one-time cost of $O(n^2)$ operations. After that, each evaluation of p_n only costs $O(n)$ operations. Unfortunately, the λ_j's as defined in (3.27) grow exponentially with the length of the interpolation interval so that (3.28) can only be used for moderate size n, without having to rescale the interval. We can eliminate this problem by noting that from (3.11) with $f(x) \equiv 1$ it follows that

$$1 = \sum_{j=0}^{n} l_j(x) = \omega(x) \sum_{j=0}^{n} \frac{\lambda_j}{x - x_j}. \tag{3.29}$$

Dividing (3.28) by (3.29), we get the so-called *barycentric formula* for interpolation:

$$p_n(x) = \frac{\displaystyle\sum_{j=0}^{n} \frac{\lambda_j}{x - x_j} f_j}{\displaystyle\sum_{j=0}^{n} \frac{\lambda_j}{x - x_j}}, \qquad \text{for } x \neq x_j, \quad j = 0, 1, \ldots, n. \tag{3.30}$$

If x coincides with one of the nodes x_j, the interpolation property $p_n(x_j) = f_j$ should be used.

The numbers λ_j depend only on the nodes $x_0, x_1, ..., x_n$ and not on given values $f_0, f_1, ..., f_n$. We can obtain them explicitly for both the Chebyshev nodes (3.13) and for the equally spaced nodes (3.12) and can be precomputed efficiently for a general set of nodes.

3.3.1 Barycentric Weights for Chebyshev Nodes

The Chebyshev nodes are the zeros of $q_{n+1}(x) = (1 - x^2)U_{n-1}(x)$, where $U_{n-1}(x) = \sin n\theta / \sin\theta$, $x = \cos\theta$ is the Chebyshev polynomial of the second kind of degree $n - 1$, with leading order coefficient 2^{n-1} [see Section 2.4]. Since the λ_j's can be defined up to a multiplicative constant (which would cancel out in the barycentric formula) we can take λ_j to be proportional to $1/q'_{n+1}(x_j)$. Since

$$q_{n+1}(x) = \sin\theta \sin n\theta, \tag{3.31}$$

differentiating we get

$$q'_{n+1}(x) = -n\cos n\theta - \sin n\theta \cot\theta. \tag{3.32}$$

Thus,

$$q'_{n+1}(x_j) = \begin{cases} -2n, \ \text{for } j = 0, \\ -(-1)^j n, \ \text{for } j = 1, \ldots, n-1, \\ -2n \ (-1)^n \ \text{for } j = n. \end{cases} \tag{3.33}$$

We can factor out $-n$ in (3.33) to obtain the barycentric weights for the Chebyshev points

$$\lambda_j = \begin{cases} \frac{1}{2}, \ \text{for } j = 0, \\ (-1)^j, \ \text{for } j = 1, \ldots, n-1, \\ \frac{1}{2} \ (-1)^n \ \text{for } j = n. \end{cases} \tag{3.34}$$

Note that for a general interval $[a,b]$, the term $(a+b)/2$ in the change of variables (3.14) cancels out in (3.25) but we gain an extra factor of $[(b-a)/2]^n$. However, this factor can be omitted as it does not alter the barycentric formula. Therefore, the same barycentric weights (3.34) can also be used for the Chebyshev nodes in an interval $[a,b]$.

3.3.2 Barycentric Weights for Equispaced Nodes

For equispaced points, $x_j = x_0 + jh$, $j = 0, 1, \ldots, n$ we have

$$
\begin{aligned}
\lambda_j &= \frac{1}{(x_j - x_0)\cdots(x_j - x_{j-1})(x_j - x_{j+1})\cdots(x_j - x_n)} \\
&= \frac{1}{(jh)[(j-1)h]\cdots(h)(-h)(-2h)\cdots(j-n)h} \\
&= \frac{1}{(-1)^{n-j}h^n[j(j-1)\cdots 1][1\cdot 2\cdots(n-j)]} \\
&= \frac{1}{(-1)^{n-j}h^n n!}\,\frac{n!}{j!(n-j)!} \\
&= \frac{1}{(-1)^n h^n n!}\,(-1)^j\binom{n}{j}.
\end{aligned}
$$

We can omit the factor $1/((-1)^n h^n n!)$ because it cancels out in the barycentric formula. Thus, for equispaced nodes we can use

$$
\lambda_j = (-1)^j \binom{n}{j}, \quad j = 0, 1, \ldots n. \tag{3.35}
$$

Note that in this case the λ_j's grow very rapidly with n, limiting the use of the barycentric formula to only moderate size n for equispaced nodes. However, as we will see, equispaced nodes are not a good choice for accurate, high order polynomial interpolation in the first place.

3.3.3 Barycentric Weights for General Sets of Nodes

The barycentric weights for a general set of nodes can be computed efficiently by using the definition (3.27), i.e.

$$\lambda_j = \frac{1}{\prod_{\substack{k=0\\k\neq j}}^{n}(x_j - x_k)}, \quad j = 0, 1, \ldots n \tag{3.36}$$

and by noting the following. Suppose we have the barycentric weights for the nodes $x_0, x_1, \ldots, x_{m-1}$ and let us call these $\lambda_j^{(m-1)}$, for $j = 0, 1, \ldots, m-1$. Then, the barycentric weights $\lambda_j^{(m)}$ for the set of nodes $x_0, x_1, \ldots, x_m$ can be computed reusing the previous values:

$$\lambda_j^{(m)} = \frac{\lambda_j^{(m-1)}}{x_j - x_m}, \quad \text{for } j = 0, 1, \ldots m-1 \tag{3.37}$$

and for $j = m$ we employ directly the definition:

$$\lambda_m^{(m)} = \frac{1}{\prod_{k=0}^{m-1}(x_m - x_k)}. \tag{3.38}$$

Algorithm 3.1 shows the procedure in pseudo-code.

Algorithm 3.1 Barycentric weights for general nodes

$\lambda_0^{(0)} \leftarrow 1$
for $m = 1, \ldots, n$ **do**
  for $j = 0, \ldots, m-1$ **do**
    $\lambda_j^{(m)} \leftarrow \frac{\lambda_j^{(m-1)}}{x_j - x_m}$
  end for
  $\lambda_m^{(m)} \leftarrow \frac{1}{\prod_{k=0}^{m-1}(x_m - x_k)}$
end for

3.4 Newton's Form and Divided Differences

There is another representation of the interpolating polynomial p_n that is convenient for the derivation of some numerical methods and for the evaluation of relatively low order p_n. The idea of this representation, due to Newton, is to use successively lower order polynomials for constructing p_n.

Suppose we have gotten $p_{n-1} \in \mathbb{P}_{n-1}$, the interpolating polynomial of $(x_0, f_0), (x_1, f_1), \ldots, (x_{n-1}, f_{n-1})$ and we would like to obtain $p_n \in \mathbb{P}_n$, the interpolating polynomial of $(x_0, f_0), (x_1, f_1), \ldots, (x_n, f_n)$ by reusing p_{n-1}. The difference between these polynomials, $r = p_n - p_{n-1}$, is a polynomial of degree at most n. Moreover, for $j = 0, \ldots, n-1$

$$r(x_j) = p_n(x_j) - p_{n-1}(x_j) = f_j - f_j = 0. \tag{3.39}$$

Therefore, r can be factored as

$$r(x) = c_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}). \tag{3.40}$$

The constant c_n is called the *n-th divided difference* with respect to $x_0, x_1, ..., x_n$, and is usually denoted by $f[x_0, \ldots, x_n]$. Thus, we have

$$p_n(x) = p_{n-1}(x) + f[x_0, \ldots, x_n](x - x_0)(x - x_1)\cdots(x - x_{n-1}). \tag{3.41}$$

By the same argument, we have

$$p_{n-1}(x) = p_{n-2}(x) + f[x_0, \ldots, x_{n-1}](x - x_0)(x - x_1)\cdots(x - x_{n-2}), \tag{3.42}$$

etc. So we arrive at Newton's form of p_n:

$$p_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + \ldots + f[x_0, \ldots, x_n](x - x_0)\cdots(x - x_{n-1}). \tag{3.43}$$

Note that for $n = 1$,

$$p_1(x) = f[x_0] + f[x_0, x_1](x - x_0) \tag{3.44}$$

and the interpolation property gives

$$f_0 = p_1(x_0) = f[x_0], \tag{3.45}$$

$$f_1 = p_1(x_1) = f[x_0] + f[x_0, x_1](x_1 - x_0). \tag{3.46}$$

$$\tag{3.47}$$

Therefore

$$f[x_0] = f_0, \tag{3.48}$$

$$f[x_0, x_1] = \frac{f_1 - f_0}{x_1 - x_0}, \tag{3.49}$$

and

$$p_1(x) = f_0 + \left(\frac{f_1 - f_0}{x_1 - x_0}\right)(x - x_0). \tag{3.50}$$

Define $f[x_j] = f_j$ for $j = 0, 1, ...n$. The following identity will allow us to compute all the required divided differences, order by order.

Theorem 3.2.

$$f[x_0, x_1, ..., x_k] = \frac{f[x_1, x_2, ..., x_k] - f[x_0, x_1, ..., x_{k-1}]}{x_k - x_0}. \tag{3.51}$$

Proof. Let p_{k-1} be the interpolating polynomial of degree at most $k-1$ of $(x_0, f_0), \ldots, (x_{k-1}, f_{k-1})$ and q_{k-1} the interpolating polynomial of degree at most $k-1$ of $(x_1, f_1), \ldots, (x_k, f_k)$. Then

$$p(x) = q_{k-1}(x) + \left(\frac{x - x_k}{x_k - x_0}\right)[q_{k-1}(x) - p_{k-1}(x)]. \tag{3.52}$$

is a polynomial of degree at most k and for $j = 1, 2, ...k-1$

$$p(x_j) = f_j + \left(\frac{x_j - x_k}{x_k - x_0}\right)[f_j - f_j] = f_j.$$

Moreover, $p(x_0) = p_{k-1}(x_0) = f_0$ and $p(x_k) = q_{k-1}(x_k) = f_k$. Therefore, $p = p_k$, the interpolation polynomial of degree at most k of the points $(x_0, f_0), (x_1, f_1), \ldots, (x_k, f_k)$. From (3.43), the leading order coefficient of p_k is $f[x_0, ..., x_k]$. Equating this with the leading order coefficient of p

$$\frac{f[x_1, ..., x_k] - f[x_0, x_1, ...x_{k-1}]}{x_k - x_0},$$

gives (3.51). □

To obtain the divided differences of p_n we construct a table using (3.51), computing all first order divided differences, then the second order ones, etc. This process is illustrated in Table 3.1 for $n = 3$.

Table 3.1: Table of divided differences for $n = 3$.

x_j	f_j	1st order	2nd order	3rd order
x_0	f_0			
x_1	f_1	$f[x_0, x_1] = \frac{f_1 - f_0}{x_1 - x_0}$		
x_2	f_2	$f[x_1, x_2] = \frac{f_2 - f_1}{x_2 - x_1}$	$f[x_0, x_1, x_2] = \frac{f[x_1,x_2] - f[x_0,x_1]}{x_2 - x_0}$	
x_3	f_3	$f[x_2, x_3] = \frac{f_3 - f_2}{x_3 - x_2}$	$f[x_1, x_2, x_3] = \frac{f[x_2,x_3] - f[x_1,x_2]}{x_3 - x_1}$	$f[x_0, x_1, x_2, x_3] = \frac{f[x_1,x_2,x_3] - f[x_0,x_1,x_2]}{x_3 - x_0}$

Example 3.3. *Take the data set* $(0, 1), (1, 2), (2, 5), (3, 10)$. *Then*

x_j	f_j			
0	1			
1	2	$\frac{2-1}{1-0} = 1$		
2	5	$\frac{5-2}{2-1} = 3$	$\frac{3-1}{2-0} = 1$	
3	10	$\frac{10-5}{3-2} = 5$	$\frac{5-3}{3-1} = 1$	$\frac{1-1}{3-0} = 0$

so

$$p_3(x) = 1 + 1(x - 0) + 1(x - 0)(x - 1) + 0(x - 0)(x - 1)(x - 2) = 1 + x^2.$$

After computing the divided differences, we need to evaluate p_n at a given point x. This can be done efficiently by suitably factoring it. For example, for $n = 3$ we have

$$\begin{aligned} p_3(x) &= c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + c_3(x - x_0)(x - x_1)(x - x_2) \\ &= c_0 + (x - x_0)\left\{c_1 + (x - x_1)[c_2 + (x - x_2)c_3]\right\} \end{aligned}$$

For general n we can use the *Horner*-like scheme in Algorithm 3.2 to get $y = p_n(x)$, given the divided difference coefficients $c_0, c_1, \ldots, c_n$ and the evaluation point x.

Algorithm 3.2 Horner Scheme to evaluate p_n at x in Newton's form

```
y ← c_n
for k = n − 1, ..., 0 do
    y ← c_k + (x − x_k) * y
end for
```

3.5 Cauchy Remainder

We now assume the data $f_j = f(x_j)$, $j = 0, 1, \ldots, n$ come from a sufficiently smooth function f, which we are trying to approximate with an interpolating polynomial p_n, and we focus on the error $f - p_n$ of such approximation.

In Chapter 1 we proved that if x_0, x_1, and x are in $[a, b]$ and $f \in C^2[a, b]$ then

$$f(x) - p_1(x) = \frac{1}{2} f''(\xi(x))(x - x_0)(x - x_1),$$

where p_1 is the polynomial of degree at most 1 that interpolates $(x_0, f(x_0))$, $(x_1, f(x_1))$ and $\xi(x) \in (a, b)$. The general result about the interpolation error is the following theorem:

Theorem 3.3. *Let $f \in C^{n+1}[a, b]$, $x_0, x_1, ..., x_n \in [a, b]$ distinct, $x \in [a, b]$, and p_n be the interpolation polynomial of degree at most n of f at $x_0, ..., x_n$. Then,*

$$f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi(x))(x - x_0)(x - x_1) \cdots (x - x_n), \quad (3.53)$$

where $\min\{x_0, \ldots, x_n, x\} < \xi(x) < \max\{x_0, \ldots, x_n, x\}$.

Proof. The right hand side of (3.53) is known as the Cauchy remainder.

For x equal to one of the nodes x_j the result is trivially true. Take x fixed not equal to any of the nodes and define

$$\phi(t) = f(t) - p_n(t) - [f(x) - p_n(x)] \frac{(t - x_0)(t - x_1) \cdots (t - x_n)}{(x - x_0)(x - x_1) \cdots (x - x_n)}. \quad (3.54)$$

Clearly, $\phi \in C^{n+1}[a, b]$ and vanishes at $t = x_0, x_1, ..., x_n, x$. That is, ϕ has at least $n + 2$ distinct zeros. Applying Rolle's theorem $n + 1$ times we conclude that there exists a point $\xi(x) \in (a, b)$ such that $\phi^{(n+1)}(\xi(x)) = 0$ (see Fig. 3.2 for an illustration of the $n = 3$ case). Therefore,

$$0 = \phi^{(n+1)}(\xi(x)) = f^{(n+1)}(\xi(x)) - [f(x) - p_n(x)] \frac{(n+1)!}{(x - x_0)(x - x_1) \cdots (x - x_n)}$$

from which (3.53) follows. Note that the repeated application of Rolle's theorem implies that $\xi(x)$ is between $\min\{x_0, x_1, ..., x_n, x\}$ and $\max\{x_0, x_1, ..., x_n, x\}$.

$\square$

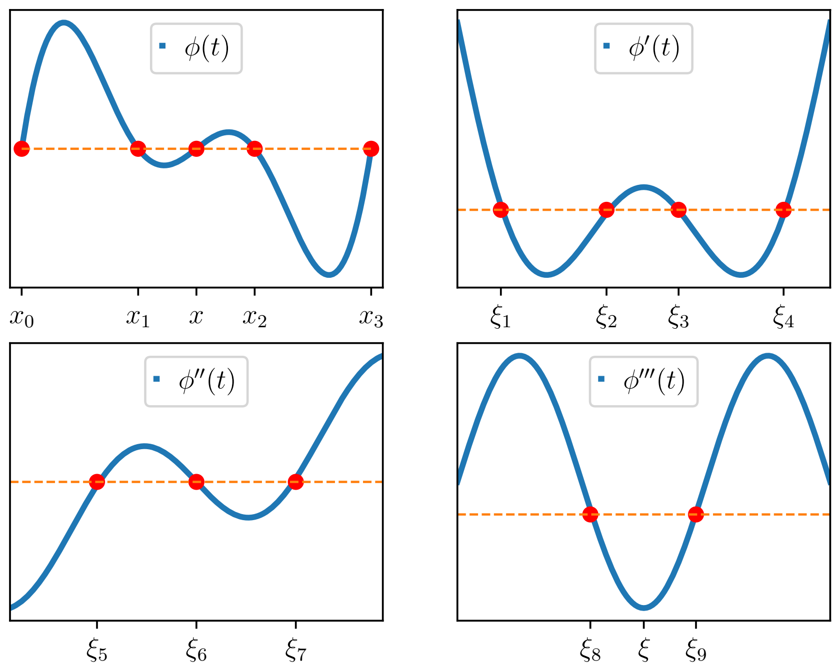

Figure 3.2: Successive application of Rolle's Theorem on $\phi(t)$ for Theorem 3.3, $n = 3$.

Example 3.4. *Let us find an approximation to* $\cos(0.8\pi)$ *using interpolation of the values* $(0,1),(0.5,0),(1,-1),(1.5,0),(2,1)$*. We first employ Newton's divided differences to get* p_4*.*

x_j	f_j				
0	1				
0.5	0	-2			
1	-1	-2	0		
1.5	0	2	4	8/3	
2	1	2	0	$-8/3$	$-8/3$

Thus,

$$p_4(x) = 1 - 2x + \frac{8}{3}x(x-0.5)(x-1) - \frac{8}{3}x(x-0.5)(x-1)(x-1.5).$$

Then, $\cos(0.8\pi) \approx p_4(0.8) = -0.8176$*. Let us find an upper bound for the error using the Cauchy remainder. Since* $f(x) = cos(\pi x)$*,* $|f^5(x)| \leq \pi^5$ *for all* x*. Therefore,*

$$\begin{aligned} |\cos(0.8\pi) - p_4(0.8)| &\leq \frac{\pi^5}{5!}|(0.8-0)(0.8-0.5)(0.8-1)(0.8-1.5)(0.8-2)| \\ &\approx 0.10. \end{aligned} \tag{3.55}$$

This is a significant overestimate of the actual error $|\cos(0.8\pi) - p_4(0.8)| \approx 0.0086$ *because we replaced* $f^{(5)}(\xi(x))$ *with a global bound of the fifth derivative. Figure 3.3 shows a plot of* f *and* p_4*. Note that the interpolation nodes are equispaced and the largest error is produced toward the end of the interpolation interval.*

We have no control on the term $f^{(n+1)}(\xi(x))$ but if we have freedom to select the interpolation nodes $x_0, \ldots, x_n$, we can choose them so that the node polynomial

$$w(x) = (x-x_0)(x-x_1)\cdots(x-x_n) \tag{3.56}$$

has the smallest possible infinity norm. In $[-1,1]$, we know the answer for we have proved in Section 2.4 that the monic Chebyshev polynomial

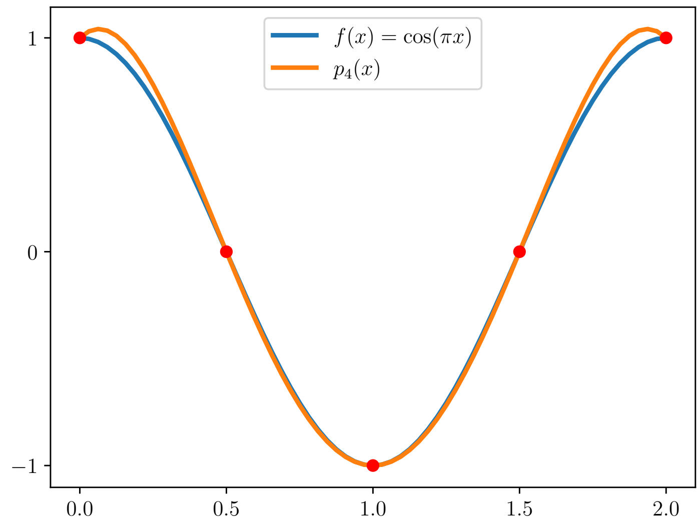

Figure 3.3: $f(x) = \cos(\pi x)$ in $[0, 2]$ and its interpolating polynomial p_4 at $x_j = j/2$, $j = 0, 1, 2, 3, 4$.

$\widetilde{T}_{n+1} = T_{n+1}/2^n$ is the monic polynomial of degree $n+1$ with smallest infinity norm in $[-1, 1]$. Hence, if the interpolation nodes are taken to be the zeros of $\widetilde{T}_{n+1}$, namely

$$x_j = \cos\left(\frac{(2j+1)}{n+1}\frac{\pi}{2}\right), \quad j = 0, 1, \ldots n. \tag{3.57}$$

$\|w\|_\infty = \max_{x\in[-1,1]} |w(x)|$ is minimized and $\|w\|_\infty = 2^{-n}$. Figure 3.4 shows a plot of w for equispaced nodes and for the nodes (3.57) for $n = 10$ in $[-1, 1]$. For equispaced nodes, w oscillates unevenly with much larger (absolute) values toward the end of the interval than around the center. In contrast, for the nodes (3.57), w equioscillates between $\pm 1/2^n$, which is a small fraction of maximum amplitude of the equispaced-node w. The following theorem summarizes this observation.

Theorem 3.4. *Let Π_n be the interpolating polynomial of degree at most n of $f \in C^{n+1}[-1, 1]$ with respect to the nodes (3.57) then*

$$\|f - \Pi_n\|_\infty \leq \frac{1}{2^n(n+1)!}\|f^{n+1}\|_\infty. \tag{3.58}$$

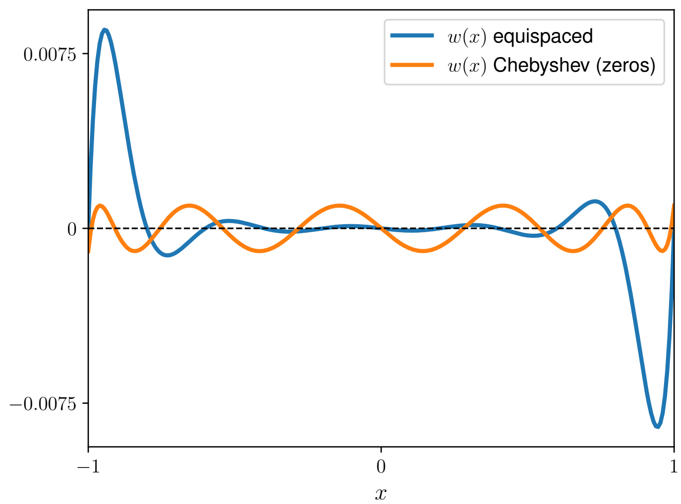

Figure 3.4: The node polynomial $w(x) = (x - x_0) \cdots (x - x_n)$, for equispaced nodes and for the zeros of T_{n+1} taken as nodes, $n = 10$.

The Chebyshev nodes,

$$x_j = \cos\left(\frac{j\pi}{n}\right), \quad j = 0, 1, \ldots, n, \tag{3.59}$$

which are the extremal points and not the zeros of the corresponding Chebyshev polynomial, do not minimize $\max_{x\in[-1,1]} |w(x)|$. However, they are nearly optimal. More precisely, since the Chebyshev nodes (3.59) are the zeros of the (monic) polynomial [see (2.85) and (3.31)]

$$\frac{1}{2^{n-1}}(1 - x^2)U_{n-1}(x) = \frac{1}{2^{n-1}} \sin\theta \sin n\theta, \quad x = \cos\theta. \tag{3.60}$$

We have that

$$\|w\|_\infty = \max_{x\in[-1,1]} \left| \frac{1}{2^{n-1}}(1 - x^2)U_{n-1}(x) \right| = \frac{1}{2^{n-1}}. \tag{3.61}$$

Thus, the Chebyshev nodes yield a $\|w\|_\infty$ of no more than a factor of two from the optimal value. Figure 3.5 compares w for equispaced nodes and for the Chebyshev nodes. For the latter, w is qualitatively very similar to that with the (3.57) nodes but, as we just proved, with an amplitude twice as large.

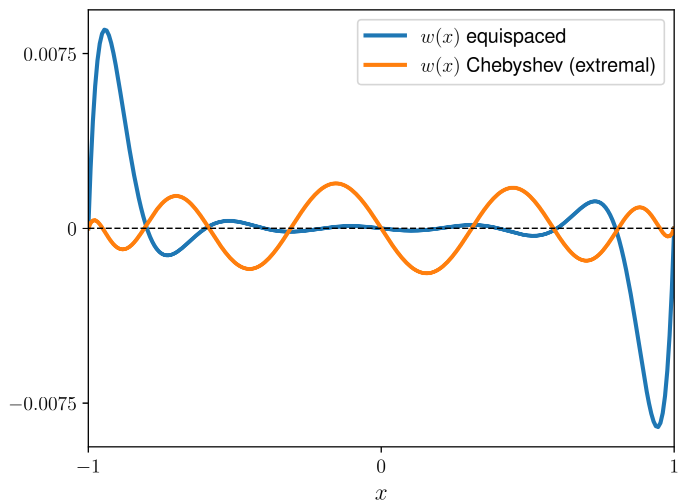

Figure 3.5: The node polynomial $w(x) = (x - x_0) \cdots (x - x_n)$, for equispaced nodes and for the Chebyshev nodes, the extremal points of T_n, $n = 10$.

3.6 Divided Differences and Derivatives

We now relate divided differences to the derivatives of f using the Cauchy remainder. Take an arbitrary point t distinct from $x_0, \ldots, x_n$. Let p_{n+1} be the interpolating polynomial of f at $x_0, \ldots, x_n, t$ and p_n that at $x_0, \ldots, x_n$. Then, Newton's formula (3.41) implies

$$p_{n+1}(x) = p_n(x) + f[x_0, \ldots, x_n, t](x - x_0)(x - x_1) \cdots (x - x_n). \tag{3.62}$$

Noting that $p_{n+1}(t) = f(t)$ we get

$$f(t) = p_n(t) + f[x_0, \ldots, x_n, t](t - x_0)(t - x_1) \cdots (t - x_n). \tag{3.63}$$

Since t was arbitrary we can set $t = x$ and obtain

$$f(x) = p_n(x) + f[x_0, \ldots, x_n, x](x - x_0)(x - x_1) \cdots (x - x_n), \tag{3.64}$$

and upon comparing with the Cauchy remainder we get

$$f[x_0, ..., x_n, x] = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}. \tag{3.65}$$

If we set $x = x_{n+1}$ and relabel $n+1$ by k we have

$$f[x_0, ..., x_k] = \frac{1}{k!} f^{(k)}(\xi), \tag{3.66}$$

where $\min\{x_0, \ldots, x_k\} < \xi < \max\{x_0, \ldots, x_k\}$.

Suppose that we now let $x_1, ..., x_k \to x_0$. Then $\xi \to x_0$ and

$$\lim_{x_1,...,x_k \to x_0} f[x_0, ..., x_k] = \frac{1}{k!} f^{(k)}(x_0). \tag{3.67}$$

We can use this relation to define a divided difference where there are *coincident* nodes. For example $f[x_0, x_1]$ when $x_0 = x_1$ by $f[x_0, x_0] = f'(x_0)$, etc. This is going to be very useful for interpolating both function and derivative values.

3.7 Hermite Interpolation

The Hermite interpolation problem is the following: given values of f and some of its derivatives at the nodes $x_0, x_1, ..., x_n$, find the polynomial of smallest degree interpolating those values. This polynomial is called the *Hermite interpolation polynomial* and can be obtained with a minor modification to Newton's representation of the interpolating polynomial.

For example, suppose we look for a polynomial p of lowest degree which satisfies the interpolation conditions:

$$\begin{aligned} p(x_0) &= f(x_0), \\ p'(x_0) &= f'(x_0), \\ p(x_1) &= f(x_1), \\ p'(x_1) &= f'(x_1). \end{aligned}$$

We can view this problem as a limiting case of polynomial interpolation of f at two pairs of coincident nodes, x_0, x_0, x_1, x_1 and we can use Newton's interpolation form to obtain p. The table of divided differences, in view of (3.67), is

x_j	f_j			
x_0	$f(x_0)$			
x_0	$f(x_0)$	$f'(x_0)$		
x_1	$f(x_1)$	$f[x_0, x_1]$	$f[x_0, x_0, x_1]$	
x_1	$f(x_1)$	$f'(x_1)$	$f[x_0, x_1, x_1]$	$f[x_0, x_0, x_1, x_1]$

(3.68)

and

$$
\begin{aligned}
p(x) = f(x_0) + f'(x_0)(x - x_0) + f[x_0, x_0, x_1](x - x_0)^2 \\
+ f[x_0, x_0, x_1, x_1](x - x_0)^2(x - x_1).
\end{aligned} \tag{3.69}
$$

Example 3.5. *Let $f(0) = 1$, $f'(0) = 0$ and $f(1) = \sqrt{2}$. Find the Hermite interpolation polynomial.*

We construct the table of divided differences as follows:

x_j	f_j		
0	1		
0	1	$f'(0) = 0$	
1	$\sqrt{2}$	$(\sqrt{2} - 1)/(1 - 0) = \sqrt{2} - 1$	$(\sqrt{2} - 1 - 0)/(1 - 0) = \sqrt{2} - 1$

and therefore

$$
p(x) = 1 + 0(x - 0) + (\sqrt{2} - 1)(x - 0)^2 = 1 + (\sqrt{2} - 1)x^2. \tag{3.70}
$$

3.8 Convergence of Polynomial Interpolation

From the Cauchy remainder formula

$$
f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi(x))(x - x_0)(x - x_1) \cdots (x - x_n)
$$

it is clear that the accuracy of the interpolating polynomial p_n of f depends on both the *regularity of f* and the *distribution of the interpolation nodes* $x_0, x_1, \ldots, x_n$.

The function

$$
f(x) = \frac{1}{1 + 25x^2} \qquad x \in [-1, 1], \tag{3.71}
$$

provides a classical example, due to Runge, that illustrates the importance of node distribution. It has an infinite number of continuous derivatives, i.e. $f \in C^\infty[-1, 1]$ (in fact f is real analytic in the whole real line, i.e. it has a convergent Taylor series to $f(x)$ for every $x \in \mathbb{R}$). Nevertheless, for the equispaced nodes (3.12) p_n does not converge uniformly to $f(x)$ as $n \to \infty$. In fact it diverges quite dramatically toward the end points of the interval as Fig. 3.6 demonstrates. In contrast, as Fig. 3.7 shows, there is fast

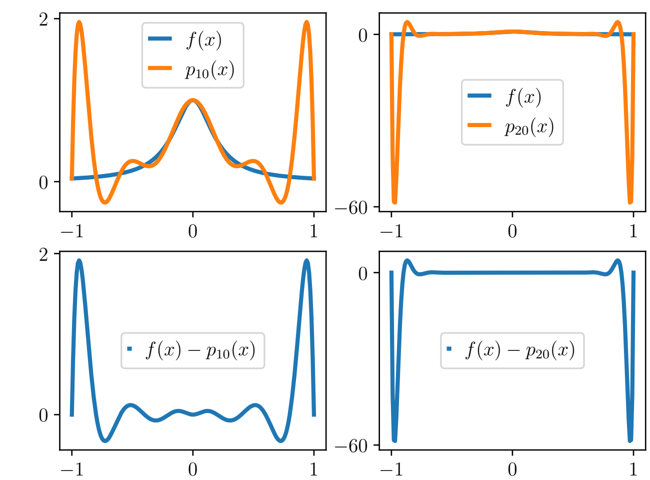

Figure 3.6: Lack of convergence of the interpolant p_n for $f(x) = 1/(1+25x^2)$ in $[-1, 1]$ using equispaced nodes. The first row shows plots of f and p_n ($n = 10, 20$) and the second row shows the corresponding error $f - p_n$.

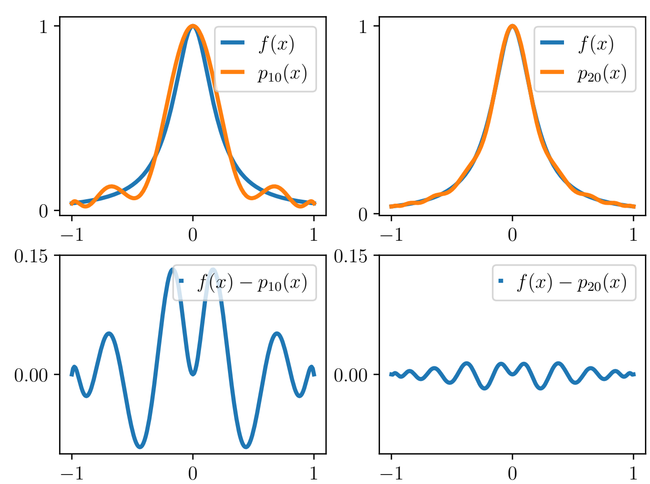

Figure 3.7: Convergence of the interpolant p_n for $f(x) = 1/(1 + 25x^2)$ in $[-1, 1]$ using Chebyshev nodes. The first row shows plots of f and p_n ($n = 10, 20$) and the second row shows the corresponding error $f - p_n$.

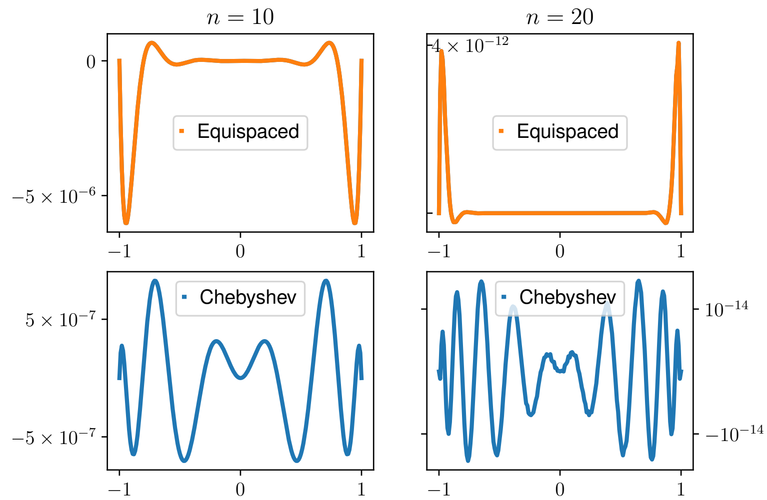

Figure 3.8: Fast convergence of the interpolant p_n for $f(x) = e^{-x^2}$ in $[-1, 1]$. Plots of the error $f - p_n$, $n = 10, 20$ for both the equispaced (first row) and the Chebyshev nodes (second row).

and uniform convergence of p_n to f when the Chebyshev nodes (3.13) are employed.

Now consider

$$f(x) = e^{-x^2}, \quad x \in [-1, 1]. \tag{3.72}$$

The interpolating polynomial p_n converges to f, even when equispaced nodes are used. In fact, the convergence is noticeably fast. Figure 3.8 shows plots of the error $f - p_n$, $n = 10, 20$, for both equispaced and Chebyshev nodes. The interpolant p_{10} has already more than 5 and 6 digits of accuracy for the equispaced and Chebyshev nodes, respectively. Note that the error when using Chebyshev nodes is significantly smaller and more equidistributed throughout the interval $[-1, 1]$ than when using equispaced nodes. For the latter, as we have seen earlier, the error is substantially larger toward the endpoints of the interval than around the center.

What is so special about $f(x) = e^{-x^2}$? The function $f(z) = e^{-z^2}$, $z \in \mathbb{C}$ is analytic in the entire complex plane[2]. Using complex variables analysis, it

[2]A function of a complex variable $f(z)$ is said to be analytic in an open set D if it has

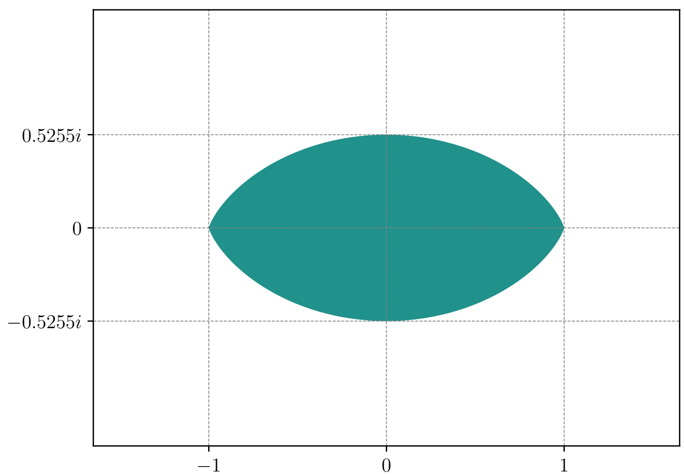

Figure 3.9: For uniform convergence of the interpolants p_n, $n = 1, 2, \ldots$ to f on $[-1, 1]$, with equi-spaced nodes, f must be analytic in the shaded, football-like region.

can be shown that if f is analytic in a sufficiently large region of the complex plane containing $[-1, 1]$ [3] then $\|f - p_n\|_\infty \to 0$. Just how large the region of analyticity needs to be? it depends on the asymptotic distribution of the nodes as $n \to \infty$. We will show next that for equispaced nodes, f must be analytic in the football-like region shown in Fig. 3.9 for p_n to converge uniformly to f in $[-1, 1]$, as $n \to \infty$. The Runge function (3.71) is not analytic in this region (it has singularities at $\pm i/5$) and hence the divergence of p_n. In contrast, for the Chebyshev nodes, it suffices that f be analytic in any region containing $[-1, 1]$, however thin this region might be, to guarantee the uniform convergence of p_n to f in $[-1, 1]$, as $n \to \infty$.

Let us consider the interpolation error, evaluated at a complex point

a derivative at every point of D. If f is analytic in D then all its derivatives exist and are analytic in D.

[3] Of course, the same arguments can be applied for a general interval $[a, b]$.

$z \in \mathbb{C}$ [4]:

$$f(z) - p_n(z) = f(z) - \sum_{j=0}^{n} l_j(z) f(x_j). \tag{3.73}$$

Employing (3.26), we can rewrite this as

$$f(z) - p_n(z) = f(z) - \sum_{j=0}^{n} \frac{\omega(z)}{(z - x_j)\omega'(x_j)} f(x_j), \tag{3.74}$$

where $\omega(z) = (z-x_0)(z-x_1)\cdots(z-x_n)$. Using the calculus of residues, the right hand side of (3.74) can be expressed as a contour integral:

$$f(z) - p_n(z) = \frac{1}{2\pi i} \oint_C \frac{\omega(z)}{\omega(\xi)} \frac{f(\xi)}{\xi - z} d\xi, \tag{3.75}$$

where C is a positively oriented closed curve that encloses $[-1, 1]$ and z but not any singularity of f. The integrand has a simple pole at $\xi = z$ with residue $f(z)$. It also has simple poles at $\xi = x_j$ for $j = 0, 1, \ldots, n$ with corresponding residues $-f(x_j)\omega(z)/[(z-x_j)\omega'(x_j)]$, which produces $-p_n(z)$. Expression (3.75) is called Hermite's formula for the interpolation remainder.

To estimate $|f(z) - p_n(z)|$ using (3.75) we need to estimate $|\omega(z)|/|\omega(\xi)|$ for $\xi \in C$ and z inside C. To this end, it is convenient to choose a contour C on which $|\omega(\xi)|$ is approximately constant for sufficiently large n. Note that

$$|\omega(\xi)| = \prod_{j=0}^{n} |\xi - x_j| = \exp\left(\sum_{j=0}^{n} \log |\xi - x_j| \right). \tag{3.76}$$

In the limit as $n \to \infty$, we can view the interpolation nodes as a continuum of a density ρ (or limiting distribution), with

$$\int_{-1}^{1} \rho(x) dx = 1, \tag{3.77}$$

so that, for sufficiently large n,

$$\text{the total number of nodes in } [\alpha, \beta] = (n+1) \int_{\alpha}^{\beta} \rho(x) dx, \tag{3.78}$$

[4] The rest of this section uses complex variables theory.

for $-1 \leq \alpha < \beta \leq 1$. Therefore, assuming the interpolation nodes have a limiting distribution ρ, we have

$$\frac{1}{n+1}\sum_{j=0}^{n} \log|\xi - x_j| \xrightarrow[n\to\infty]{} \int_{-1}^{1} \log|\xi - x|\rho(x)dx. \tag{3.79}$$

Let us define the function

$$\phi(\xi) = -\int_{-1}^{1} \log|\xi - x|\rho(x)dx. \tag{3.80}$$

Then, for sufficiently large n, $|\omega(z)|/|\omega(\xi)| \approx e^{-(n+1)[\phi(z)-\phi(\xi)]}$. The level curves of ϕ, i.e. the set of points $\xi \in \mathbb{C}$ such that $\phi(\xi) = c$, with c constant, approximate large circles for very large and negative values of c. As c is increased, the level curves shrink. Let z_0 be the singularity of f closest to the origin. Then, we can take any $\epsilon > 0$ and select C to be the level curve $\phi(\xi) = \phi(z_0)+\epsilon$ so that f is analytic on and inside C. Take z inside C. From (3.75), (3.79), and (3.80)

$$\begin{aligned} |f(z) - p_n(z)| &\leq \frac{1}{2\pi}\oint_C \frac{|\omega(z)|}{|\omega(\xi)|}\frac{|f(\xi)|}{|\xi - z|}ds \\ &\leq \text{constant } e^{-(n+1)[\phi(z)-(\phi(z_0)+\epsilon)]}. \end{aligned} \tag{3.81}$$

Therefore, it follows that $|f(z) - p_n(z)| \to 0$ as $n \to \infty$ and the convergence is exponential. Note that this holds as long as z is inside the chosen contour C. If z is outside the level curve $\phi(\xi) = \phi(z_0)$, i.e. $\phi(z) < \phi(z_0)$, then $|f(z)-p_n(z)|$ diverges exponentially. Therefore, p_n converges (uniformly) to f in $[-1, 1]$ if and only if f is analytic on and inside the smallest level curve of ϕ that contains $[-1, 1]$. More precisely, let γ be the supremum over all the values of c for which $[-1, 1]$ lies inside the level set curve $\phi(\xi) = c$. Define the region

$$D_\gamma = \{z \in \mathbb{C} : \phi(z) \geq \gamma\}. \tag{3.82}$$

Then, we have the following result.

Theorem 3.5. *The f be analytic in any region containing D_γ in its interior. Then,*

$$|f(z) - p_n(z)| \xrightarrow[n\to\infty]{} 0, \textit{uniformly, for } z \in D_\gamma. \tag{3.83}$$

For equispaced nodes, the number of nodes is the same (asymptotically) for all intervals of the same length. Therefore, ρ is a constant. The normalization condition (3.77) implies that $\rho(x) = 1/2$ for equispaced points in $[-1, 1]$. It can be shown that with $\rho(x) = 1/2$ we get

$$\phi(\xi) = 1 - \frac{1}{2}\mathrm{Re}\left\{(\xi+1)\log(\xi+1) - (\xi-1)\log(\xi-1)\right\}. \tag{3.84}$$

The curve of ϕ that bounds D_γ for equispaced nodes is the one that passes through ± 1, has value $1 - \log 2$, and is shown in Fig. 3.9. It crosses the imaginary axis at $\pm 0.5255...i$. On the hand, the level curve that passes through $\pm i/5$ crosses the real axis at about $\pm 0.7267...$. Thus, there is uniform convergence of p_n to f in the reduced interval $[-0.72, 0.72]$.

The Chebyshev points $x_j = \cos\theta_j$, $j = 0, 1, \ldots, n$, are equispaced in θ ($\theta_j = j\pi/n$) and since

$$\int_\alpha^\beta \rho(x)dx = \int_{\cos^{-1}\beta}^{\cos^{-1}\alpha} \rho(\cos\theta)\sin\theta d\theta, \tag{3.85}$$

then $\rho(\cos\theta)\sin\theta = \rho(x)\sqrt{1-x^2}$ must be constant. Using (3.77), it follows the density for Chebyshev nodes is

$$\rho(x) = \frac{1}{\pi\sqrt{1-x^2}}, \quad x \in [-1, 1]. \tag{3.86}$$

With this node distribution it can be shown that

$$\phi(\xi) = \log\frac{2}{|\xi + \sqrt{\xi^2-1}|}. \tag{3.87}$$

The level curves of ϕ in this case are the points $\xi \in \mathbb{C}$ such that $|\xi + \sqrt{\xi^2-1}| = c$, with c constant. These are ellipses with foci at ± 1 as shown in Fig. 3.10. The level curve that passes through ± 1 degenerates into the interval $[-1, 1]$.

3.9 Piecewise Polynomial Interpolation

One way to avoid the oscillatory behavior of high-order interpolation when the interpolation nodes do not cluster appropriately is to employ low order polynomials in small subintervals.

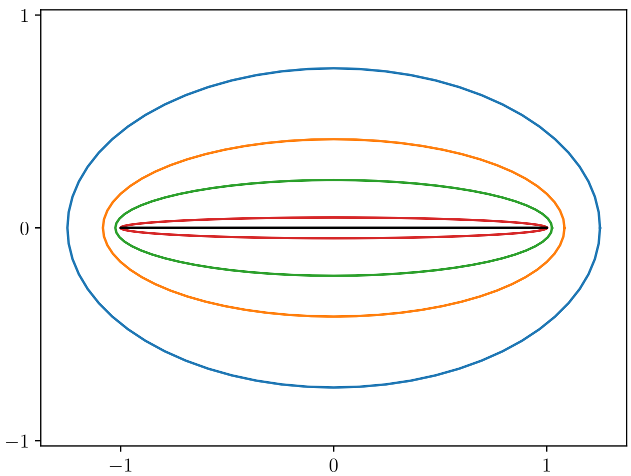

Figure 3.10: Some level curves of ϕ for the Chebyshev node distribution.

Given the nodes $a = x_0 < x_1 \ldots < x_n = b$ we can consider the subintervals $[x_0, x_1], ..., [x_{n-1}, x_n]$ and construct in each a polynomial degree at most k (for $k \geq 1$ small) that interpolates f. For $k = 1$, on each $[x_j, x_{j+1}]$, $j = 0, 1, \ldots, n-1$, we know there is a unique polynomial $s_j \in \mathbb{P}_1$ that interpolates f at x_j and x_{j+1}. Thus, there is a unique, continuous piecewise linear interpolant s of f at the given $n+1$ nodes. We simply use $\mathbb{P}_1$ interpolation for each of its pieces:

$$s_j(x) = f_j + \frac{f_{j+1} - f_j}{x_{j+1} - x_j}(x - x_j), \quad x \in [x_j, x_{j+1}], \tag{3.88}$$

for $j = 0, 1, \ldots, n-1$ and we have set $f_j = f(x_j)$. Figure 3.11 shows an illustration of this piecewise linear interpolant s.

Assuming that $f \in C^2[a, b]$, we know that

$$f(x) - s(x) = \frac{1}{2} f''(\xi(x))(x - x_j)(x - x_{j+1}), \qquad x \in [x_j, x_{j+1}], \tag{3.89}$$

where $\xi(x)$ is some point between x_j and x_{j+1}. Then,

$$\max_{x_j \leq x \leq x_{j+1}} |f(x) - p(x)| \leq \frac{1}{2} \|f''\|_\infty \max_{x_j \leq x \leq x_{j+1}} |(x - x_j)(x - x_{j+1})|, \tag{3.90}$$

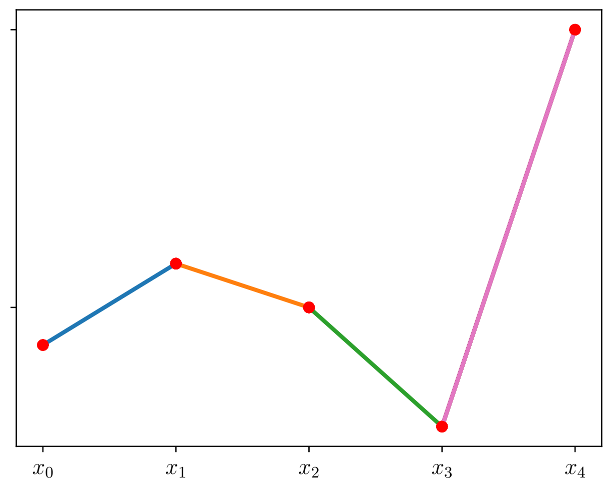

Figure 3.11: Piecewise linear interpolation.

where $\|f''\|_\infty$ is the sup norm of f'' over $[a, b]$. Now, the max at the right hand side is attained at the midpoint $(x_j + x_{j+1})/2$ and

$$\max_{x_j \leq x \leq x_{j+1}} |(x - x_j)(x - x_{j+1})| = \left(\frac{x_{j+1} - x_j}{2}\right)^2 = \frac{1}{4}h_j^2, \tag{3.91}$$

where $h_j = x_{j+1} - x_j$. Therefore

$$\max_{x_j \leq x \leq x_{j+1}} |f(x) - p(x)| \leq \frac{1}{8}\|f''\|_\infty \, h_j^2. \tag{3.92}$$

If we add more nodes, we can make h_j sufficiently small so that the error is smaller than a prescribed tolerance δ. That is, we can pick h_j such that $\frac{1}{8}\|f''\|_\infty h_j^2 \leq \delta$, which implies

$$h_j \leq \sqrt{\frac{8\delta}{\|f''\|_\infty}}. \tag{3.93}$$

This gives us an adaptive procedure to obtain a desired accuracy.

Continuous, piecewise quadratic interpolants ($k = 2$) can be obtained by adding an extra point in each subinterval, say its midpoint, so that each piece

$s_j \in \mathbb{P}_2$ is the one that interpolates f at $x_j, \frac{1}{2}(x_j + x_{j+1}), x_{j+1}$. For $k = 3$, we need to add 2 more points on each subinterval, etc. This procedure allows us to construct continuous, piecewise polynomial interpolants of f and if $f \in C^{k+1}[a, b]$ one can simply use the Cauchy remainder on each subinterval to get a bound for the error, as we did for the piecewise linear case.

Sometimes a smoother piecewise polynomial interpolant s is needed. If we want $s \in C^m[a, b]$ then on the first subinterval, $[x_0, x_1]$, we can take an arbitrary polynomial of degree at most k ($k + 1$ degrees of freedom) but in the second subinterval the corresponding polynomial has to match $m + 1$ (continuity plus m derivatives) conditions at x_1 so we only have $k-m$ degrees of freedom for it, and so on. Thus, in total we have $k+1+(n-1)(k-m) = n(k-m)+m+1$ degrees of freedom. For $m = k$ we only have $k+1$ degrees of freedom and since $s \in \mathbb{P}_k$ on each subinterval, it must be a polynomial of degree at most k in the entire interval $[a, b]$. Moreover, since polynomials are C^∞ it follows that $s \in \mathbb{P}_k$ on $[a, b]$ for $m \geq k$. So we restrict ourselves to $m < k$ and specifically focus on the case $m = k - 1$. These functions are called *splines*.

Definition 3.1. *Given a partition*

$$\Delta = \{a = x_0 < x_1 \ldots < x_n = b\} \tag{3.94}$$

of $[a, b]$, *the functions in the set*

$$\mathbb{S}^k_\Delta = \left\{ s : s \in C^{k-1}[a, b],\ s\,\Big|_{[x_j, x_{j+1}]} \in \mathbb{P}_k, j = 0, 1, \ldots, n-1 \right\} \tag{3.95}$$

are called splines of degree k *(or order* $k + 1$*). The nodes* x_j*,* $j = 0, 1, \ldots, n$*, are called knots or breakpoints.*

Note that if s and r are in $\mathbb{S}^k_\Delta$ so is $as + br$, i.e. $\mathbb{S}^k_\Delta$ is a linear space, a subspace of $C^{k-1}[a, b]$. The piecewise linear interpolant is a spline of degree 1. We are going to study next splines of degree 3.

3.10 Cubic Splines

Several applications require smoother approximations than that provided by a piece-wise linear interpolation. For example, continuity up to the second derivative is generally desired in computer graphics applications. With the C^2 requirement, we need to consider splines of degree $k \geq 3$. The case $k = 3$

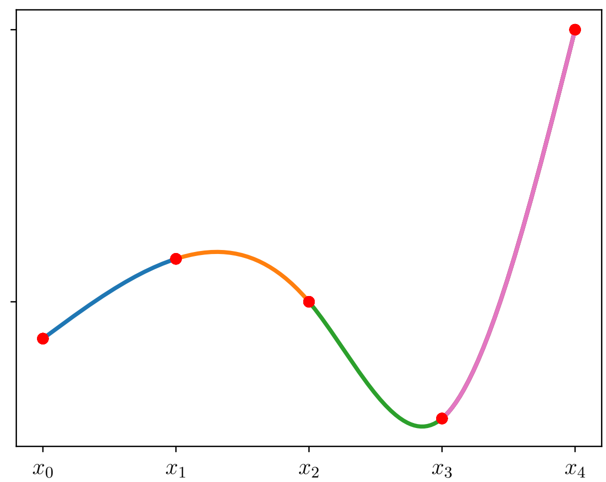

Figure 3.12: Cubic spline s interpolating 5 data points. Each color represents a cubic polynomial constructed so that s interpolates the given data, has two continuous derivatives, and $s''(x_0) = s''(x_4) = 0$.

is the most widely used and the corresponding splines are simply called *cubic splines*.

We consider here cubic splines that interpolate a set of values $f_0, f_1, \ldots, f_n$ at the nodes $a = x_0 < x_1 \ldots < x_n = b$, i.e. $s \in \mathbb{S}^3_\Delta$ with $s(x_j) = f_j$, $j = 0, 1, \ldots, n$. We call such a function a *cubic spline interpolant*. Figure 3.12 shows an example of a cubic spline interpolating 5 data points. The cubic polynomial pieces (s_j for $j = 0, 1, 2, 3$), appearing in different colors, are stitched together so that s interpolates the given data and has two continuous derivatives. The same data points have been used in both Fig. 3.11 and Fig. 3.12. Note the striking difference of the two interpolants.

As we saw in Section 3.9, there are $n+3$ degrees of freedom to determine $s \in \mathbb{S}^3_\Delta$, two more than the $n+1$ interpolation conditions. The two extra conditions could be the first or the second derivative of s at the end points ($x = a, x = b$). Note that if $s \in \mathbb{S}^3_\Delta$ then $s'' \in \mathbb{S}^1_\Delta$, i.e. the second derivative of a cubic spline is a continuous, piece-wise linear spline. Consequently, s'' is determined uniquely by its $(n+1)$ values

$$m_j = s''(x_j), \quad j = 0, 1, \ldots, n. \tag{3.96}$$

In the following construction of cubic spline interpolants we impose the $n+1$ interpolation conditions plus two extra conditions to find the unique values m_j, $j = 0, 1, \ldots, n$ that s'' must have at the nodes in order for s to be $C^2[a, b]$.

3.10.1 Natural Splines

Cubic splines with a vanishing second derivative at the first and last node, $m_0 = 0$ and $m_n = 0$, are called natural cubic splines. They are useful in graphics but not good for approximating a function f, unless f happens to also have vanishing second derivatives at x_0 and x_n.

We are now going to derive a linear system of equations for the values $m_1, m_2, \ldots, m_{n-1}$ that define the natural cubic spline interpolant. Once this system is solved we obtain the spline piece by piece.

In each subinterval $[x_j, x_{j+1}]$, s is a polynomial $s_j \in \mathbb{P}_3$, which we may represent as

$$s_j(x) = A_j(x - x_j)^3 + B_j(x - x_j)^2 + C_j(x - x_j) + D_j, \tag{3.97}$$

for $j = 0, 1, \ldots, n-1$. To simplify the formulas below we let

$$h_j = x_{j+1} - x_j. \tag{3.98}$$

The spline s interpolates the given data. Thus, for $j = 0, 1, \ldots n-1$

$$s_j(x_j) = D_j = f_j, \tag{3.99}$$

$$s_j(x_{j+1}) = A_j h_j^3 + B_j h_j^2 + C_j h_j + D_j = f_{j+1}. \tag{3.100}$$

Now $s_j'(x) = 3A_j(x - x_j)^2 + 2B_j(x - x_j) + C_j$ and $s_j''(x) = 6A_j(x - x_j) + 2B_j$. Therefore, for $j = 0, 1, \ldots n-1$

$$s_j'(x_j) = C_j, \tag{3.101}$$

$$s_j'(x_{j+1}) = 3A_j h_j^2 + 2B_j h_j + C_j, \tag{3.102}$$

and

$$s_j''(x_j) = 2B_j, \tag{3.103}$$

$$s_j''(x_{j+1}) = 6A_j h_j + 2B_j. \tag{3.104}$$

Since s'' is continuous

$$m_{j+1} = s''(x_{j+1}) = s_{j+1}''(x_{j+1}) = s_j''(x_{j+1}) \tag{3.105}$$

and we can write (3.103)-(3.104) as

$$m_j = 2B_j, \tag{3.106}$$
$$m_{j+1} = 6A_j h_j + 2B_j. \tag{3.107}$$

We now write A_j, B_j, C_j, and D_j in terms of the unknown values m_j and m_{j+1}, and the known values f_j and f_{j+1}. We have

$$\begin{aligned}
D_j &= f_j, \\
B_j &= \frac{1}{2} m_j, \\
A_j &= \frac{1}{6h_j}(m_{j+1} - m_j)
\end{aligned}$$

and substituting these values in (3.100) we get

$$C_j = \frac{1}{h_j}(f_{j+1} - f_j) - \frac{1}{6} h_j (m_{j+1} + 2m_j).$$

Let us collect all our formulas for the spline coefficients:

$$A_j = \frac{1}{6h_j}(m_{j+1} - m_j), \tag{3.108}$$
$$B_j = \frac{1}{2} m_j, \tag{3.109}$$
$$C_j = \frac{1}{h_j}(f_{j+1} - f_j) - \frac{1}{6} h_j (m_{j+1} + 2m_j), \tag{3.110}$$
$$D_j = f_j, \tag{3.111}$$

for $j = 0, 1, \ldots, n-1$. So far we have only used that s and s'' are continuous and that s interpolates the given data. We are now going to impose the continuity of the first derivative of s to determine equations for the unknown values m_j, $j = 1, 2, \ldots, n-1$. Substituting (3.108)-(3.111) in (3.102) we get

$$\begin{aligned}
s_j'(x_{j+1}) &= 3A_j h_j^2 + 2B_j h_j + C_j \\
&= 3\frac{1}{6h_j}(m_{j+1} - m_j) h_j^2 + 2\frac{1}{2} m_j h_j + \frac{1}{h_j}(f_{j+1} - f_j) \\
&\quad - \frac{1}{6} h_j (m_{j+1} + 2m_j) \\
&= \frac{1}{h_j}(f_{j+1} - f_j) + \frac{1}{6} h_j (2m_{j+1} + m_j)
\end{aligned} \tag{3.112}$$

and decreasing the index by 1

$$s'_{j-1}(x_j) = \frac{1}{h_{j-1}}(f_j - f_{j-1}) + \frac{1}{6}h_{j-1}(2m_j + m_{j-1}). \tag{3.113}$$

Continuity of the first derivative means $s'_{j-1}(x_j) = s'_j(x_j)$ for $j = 1, 2, ..., n-1$. Therefore, for $j = 1, \ldots, n-1$

$$\begin{aligned} \frac{1}{h_{j-1}}(f_j - f_{j-1}) + \frac{1}{6}h_{j-1}(2m_j + m_{j-1}) = C_j \\ = \frac{1}{h_j}(f_{j+1} - f_j) - \frac{1}{6}h_j(m_{j+1} + 2m_j) \end{aligned} \tag{3.114}$$

which can be written as

$$\begin{aligned} h_{j-1}m_{j-1} + 2(h_{j-1} + h_j)m_j + h_j m_{j+1} = \\ -\frac{6}{h_{j-1}}(f_j - f_{j-1}) + \frac{6}{h_j}(f_{j+1} - f_j), \quad j = 1, \ldots, n-1. \end{aligned} \tag{3.115}$$

This is a linear system of $n-1$ equations for the $n-1$ unknowns $m_1, m_2, \ldots, m_{n-1}$. In matrix form

$$\begin{bmatrix} a_1 & b_1 & & & & & \\ c_1 & a_2 & b_2 & & & & \\ & c_2 & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & \ddots & \ddots & b_{n-2} \\ & & & & & c_{n-2} & a_{n-1} \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ m_{n-1} \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ d_{n-1} \end{bmatrix}, \tag{3.116}$$

where

$$a_j = 2(h_{j-1} + h_j), \qquad j = 1, 2, \ldots, n-1, \tag{3.117}$$

$$b_j = h_j, \qquad j = 1, 2, \ldots, n-2, \tag{3.118}$$

$$c_j = h_j, \qquad j = 1, 2, \ldots, n-2, \tag{3.119}$$

$$d_j = -\frac{6}{h_{j-1}}(f_j - f_{j-1}) + \frac{6}{h_j}(f_{j+1} - f_j), \quad j = 1, \ldots, n-1. \tag{3.120}$$

Note that we have used $m_0 = m_n = 0$ in the first and last equation of this linear system. The matrix of the linear system (3.116) is strictly diagonally dominant, a concept we make precise in the definition below. A consequence of this property is that the matrix is nonsingular and therefore the linear system (3.116) has a unique solution. Moreover, this tridiagonal linear system can be solved efficiently with Algorithm 9.5. Once $m_1, m_2, \ldots, m_{n-1}$ are found, the spline coefficients can be computed from (3.108)-(3.111).

Definition 3.2. *An $n \times n$ matrix A with entries a_{ij}, $i, j = 1, \ldots, n$ is strictly diagonally dominant if*

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \quad \text{for } i = 1, \ldots, n. \tag{3.121}$$

Theorem 3.6. *Let A be a strictly diagonally dominant matrix. Then A is nonsingular.*

Proof. Suppose the contrary, that is there is $x \neq 0$ such that $Ax = 0$. Let k be an index such that $|x_k| = \|x\|_\infty$. Then, the k-th equation in $Ax = 0$ gives

$$a_{kk}x_k + \sum_{\substack{j=1 \\ j \neq k}}^{n} a_{kj}x_j = 0 \tag{3.122}$$

and consequently

$$|a_{kk}||x_k| \leq \sum_{\substack{j=1 \\ j \neq k}}^{n} |a_{kj}||x_j|. \tag{3.123}$$

Dividing by $|x_k|$, which by assumption in nonzero, and using that $|x_j|/|x_k| \leq 1$ for all $j = 1, \ldots, n$, we get

$$|a_{kk}| \leq \sum_{\substack{j=1 \\ j \neq k}}^{n} |a_{kj}|, \tag{3.124}$$

which contradicts the fact that A is strictly diagonally dominant. □

Example 3.6. *Find the natural cubic spline that interpolates* $(0, 0), (1, 1), (2, 0)$. *We know $m_0 = 0$ and $m_2 = 0$. We only need to find m_1 (only 1 interior node). The system (3.115) degenerates to just one equation. With $h_0 = h_1 = 1$ we have*

$$m_0 + 4m_1 + m_2 = 6[f_0 - 2f_1 + f_2] \Rightarrow m_1 = -3$$

In $[0,1]$*:*

$$\begin{aligned}
A_0 &= \frac{1}{6}(m_1 - m_0) = \left(\frac{1}{6}\right)(-3) = -\frac{1}{2},\\
B_0 &= \frac{1}{2}m_0 = 0\\
C_0 &= f_1 - f_0 - \frac{1}{6}(m_1 + 2m_0) = 1 + \frac{1}{2} = \frac{3}{2},\\
D_0 &= f_0 = 0.
\end{aligned}$$

Thus, $s_0(x) = A_0(x-0)^3 + B_0(x-0)^2 + C_0(x-0) + D_0 = -\frac{1}{2}x^3 + \frac{3}{2}x.$

In $[1,2]$*:*

$$\begin{aligned}
A_1 &= \frac{1}{6}(m_2 - m_1) = \left(\frac{1}{6}\right)(3) = \frac{1}{2},\\
B_1 &= \frac{1}{2}m_1 = -\frac{3}{2},\\
C_1 &= f_2 - f_1 - \frac{1}{6}(m_2 + 2m_1) = 0 - 1 - \frac{1}{6}(-6) = 0,\\
D_1 &= f_1 = 1.
\end{aligned}$$

and $s_1(x) = \frac{1}{2}(x-1)^3 - \frac{3}{2}(x-1)^2 + 1$*. Therefore, the natural cubic spline that interpolates the given data is*

$$s(x) = \begin{cases} -\frac{1}{2}x^3 + \frac{3}{2}x, & x \in [0,1],\\ \\ \frac{1}{2}(x-1)^3 - \frac{3}{2}(x-1)^2 + 1, & x \in [1,2]. \end{cases}$$

3.10.2 Complete Splines

If we are interested in approximating a function with a cubic spline interpolant it is generally more accurate to specify the first derivative at the endpoints instead of imposing a vanishing second derivative. A cubic spline where we specify $s'(a)$ and $s'(b)$ is called a *complete spline*.

In a complete spline the values m_0 and m_n of s'' at the endpoints become unknowns together with $m_1, m_2, \ldots, m_{n-1}$. Thus, we need to add two more equations to have a complete system for all the $n+1$ unknown values $m_0, m_1, \ldots, m_n$. Recall that

$$s_j(x) = A_j(x-x_j)^3 + B_j(x-x_j)^2 + C_j(x-x_j) + D_j$$

and so $s'_j(x) = 3A_j(x-x_j)^2 + 2B_j(x-x_j) + C_j$. Therefore

$$s'_0(x_0) = C_0 = f'_0, \tag{3.125}$$
$$s'_{n-1}(x_n) = 3A_{n-1}h_{n-1}^2 + 2B_{n-1}h_{n-1} + C_{n-1} = f'_n, \tag{3.126}$$

where $f'_0 = f'(x_0)$ and $f'_n = f'(x_n)$. Substituting C_0, A_{n-1}, B_{n-1}, and C_{n-1} from (3.108)-(3.110) we get

$$2h_0m_0 + h_0m_1 = \frac{6}{h_0}(f_1 - f_0) - 6f'_0, \tag{3.127}$$
$$h_{n-1}m_{n-1} + 2h_{n-1}m_n = -\frac{6}{h_{n-1}}(f_n - f_{n-1}) + 6f'_n. \tag{3.128}$$

If we append (3.127) and (3.127) at the top and the bottom of the system (3.115), respectively and set $h_{-1} = h_n = 0$ we obtain the following tridiagonal linear system for the values of the second derivative of the complete spline at the knots:

$$\begin{bmatrix} a_0 & b_0 & & & & \\ c_0 & a_1 & b_1 & & & \\ & c_1 & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots \\ & & & & \ddots & \ddots & b_{n-1} \\ & & & & & c_{n-1} & a_n \end{bmatrix} \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ m_n \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ d_n \end{bmatrix}, \tag{3.129}$$

where

$$a_j = 2(h_{j-1} + h_j), \qquad j = 0, 1, \ldots, n, \tag{3.130}$$
$$b_j = h_j, \qquad j = 0, 1, \ldots, n-1, \tag{3.131}$$
$$c_j = h_j, \qquad j = 0, 1, \ldots, n-1, \tag{3.132}$$
$$d_0 = \frac{6}{h_0}(f_1 - f_0) - 6f'_0, \tag{3.133}$$
$$d_j = -\frac{6}{h_{j-1}}(f_j - f_{j-1}) + \frac{6}{h_j}(f_{j+1} - f_j), \quad j = 1, \ldots, n-1, \tag{3.134}$$
$$d_n = -\frac{6}{h_{n-1}}(f_n - f_{n-1}) + 6f'_n. \tag{3.135}$$

As in the case of natural cubic splines, this linear system is also diagonally dominant (hence nonsingular) and can be solved efficiently with Algorithm 9.5.

It can be proved that if f is sufficiently smooth its complete spline interpolant s produces an error $\|f - s\|_\infty \leq Ch^4$, where $h = \max_i h_i$, whereas for the natural cubic spline interpolant the error deteriorates to $O(h^2)$ near the endpoints.

3.10.3 Minimal Bending Energy

Consider a curve given by $y = f(x)$ for $x \in [a, b]$, where $f \in C^2[a, b]$. Its curvature is

$$\kappa(x) = \frac{f''(x)}{[1 + (f'(x))^2]^{3/2}} \tag{3.136}$$

and a measure of how much the curve "curves" or bends is its bending energy

$$E_b = \int_a^b \kappa^2(x)dx. \tag{3.137}$$

For curves with small $|f'|$ compared to 1, $\kappa(x) \approx f''(x)$ and $E_b \approx \|f''\|_2^2$. We are going to show that cubic splines interpolants are C^2 functions that have minimal $\|f''\|_2$, in a sense we make more precise below. To show this we are going to use the following two results.

Lemma 3.10.1. *Let $s \in \mathbb{S}^3_\Delta$ be a cubic spline interpolant of $f \in C^2[a, b]$ at the nodes $\Delta = \{a = x_0 < x_1 \ldots < x_n = b\}$. Then, for all $g \in \mathbb{S}^1_\Delta$*

$$\int_a^b [f''(x) - s''(x)]g(x)dx = [f'(b) - s'(b)]g(b) - [f'(a) - s'(a)]g(a). \tag{3.138}$$

Proof.

$$\int_a^b [f''(x) - s''(x)]g(x)dx = \sum_{j=0}^{n-1} \int_{x_j}^{x_{j+1}} [f''(x) - s''(x)]g(x)dx. \tag{3.139}$$

We can integrate by parts on each interval:

$$\begin{aligned}\int_{x_j}^{x_{j+1}} [f''(x) - s''(x)]g(x)dx &= [f'(x) - s'(x)]g(x)\Big|_{x_j}^{x_{j+1}} \\ &\quad - \int_{x_j}^{x_{j+1}} [f'(x) - s'(x)]g'(x)dx.\end{aligned} \tag{3.140}$$

Substituting this in (3.139) the boundary terms telescope and we obtain

$$\int_a^b [f''(x) - s''(x)]g(x)dx = [f'(b) - s'(b)]g(b) - [f'(a) - s'(a)]g(a) \\ - \sum_{j=0}^{n-1} \int_{x_j}^{x_{j+1}} [f'(x) - s'(x)]g'(x)dx. \tag{3.141}$$

On each subinterval $[x_j, x_{j+1}]$, g' is constant and $f - s$ vanishes at the endpoints. Therefore, the last term is zero. □

Theorem 3.7. *Let $s \in \mathbb{S}^3_\Delta$ be the (natural or complete) cubic spline interpolant of $f \in C^2[a, b]$ at the nodes $\Delta = \{a = x_0 < x_1 \ldots < x_n = b\}$. Then,*

$$\|s''\|_2 \leq \|f''\|_2. \tag{3.142}$$

Proof.

$$\|f'' - s''\|_2^2 = \int_a^b [f''(x) - s''(x)]^2 dx = \|f''\|_2^2 + \|s''\|_2^2 - 2\int_a^b f''(x)s''(x)dx \\ = \|f''\|_2^2 - \|s''\|_2^2 - 2\int_a^b [f''(x) - s''(x)]s''(x)dx. \tag{3.143}$$

By Lemma 3.10.1 with $g = s''$ the last term vanishes for the natural spline ($s''(a) = s''(b) = 0$) and for the complete spline ($s'(a) = f'(a)$ and $s'(b) = f'(b)$) and we get the identify

$$\|f'' - s''\|_2^2 = \|f''\|_2^2 - \|s''\|_2^2 \tag{3.144}$$

from which the results follows. □

In Theorem 3.7 f could be substituted for any sufficiently smooth interpolant g of the given data.

Theorem 3.8. *Let $s \in \mathbb{S}^3_\Delta$ and $g \in C^2[a, b]$ both interpolate the values $f_0, f_1, \ldots, f_n$ at the nodes $\Delta = \{a = x_0 < x_1 \ldots < x_n = b\}$. Then,*

$$\|s''\|_2 \leq \|g''\|_2, \tag{3.145}$$

if either $s''(a) = s''(b) = 0$ (natural spline) or $s'(a) = g'(a)$ and $s'(b) = g'(b)$ (complete spline).

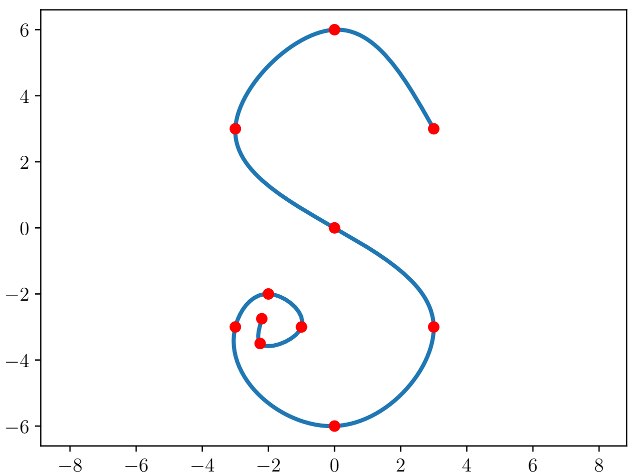

Figure 3.13: Example of a parametric spline representation to interpolate the given data points (in red).

3.10.4 Splines for Parametric Curves

In computer graphics and animation it is often required to construct smooth curves that are not necessarily the graph of a function but that have a parametric representation $x = x(t)$ and $y = y(t)$ for $t \in [a, b]$. Hence we need to determine two splines interpolating (t_j, x_j) and (t_j, y_j) $(j = 0, 1, \ldots n)$, respectively. Usually, only the position of the "control points" $(x_0, y_0), \ldots (x_n, y_n)$ is given and not the parameter values $t_0, t_1, \ldots, t_n$. In such cases, we can use the distances of consecutive control points to generate appropriate t_j's as follows:

$$t_0 = 0, \quad t_j = t_{j-1} + \sqrt{(x_j - x_{j-1})^2 + (y_j - y_{j-1})^2}, \quad j = 1, 2, \ldots n. \tag{3.146}$$

Figure 3.13 shows an example of this approach.

3.11 Trigonometric Interpolation

We consider now the important case of interpolation of a periodic array of data $(x_0, f_0), (x_1, f_1), \ldots, (x_N, f_N)$ with $f_N = f_0$, and $x_j = j(2\pi/N)$, $j = 0, 1, \ldots, N$, by a trigonometric polynomial.

Definition 3.3. *A function of the form*

$$s_n(x) = \sum_{k=-n}^{n} c_k e^{ikx}, \tag{3.147}$$

where $c_0, c_1, c_{-1}, \ldots, c_n, c_{-n}$ are complex, or equivalently of the form[5]

$$s_n(x) = \frac{1}{2}a_0 + \sum_{k=1}^{n} (a_k \cos kx + b_k \sin kx) \tag{3.148}$$

where the coefficients $a_0, a_1, b_1, \ldots, a_n, b_n$ are real is called a trigonometric polynomial of degree (at most) n.

The values f_j, $j = 0, 1, \ldots, N$, could come from a 2π-periodic function, $f(j\,2\pi/N) = f_j$, or can simply be given data. Note that the interpolation nodes are equi-spaced points in $[0, 2\pi]$. One can accommodate any other period by doing a simple scaling. Because of periodicity ($f_N = f_0$), we only have N independent data points $(x_0, f_0), \ldots, (x_{N-1}, f_{N-1})$ or $(x_1, f_1), \ldots, (x_N, f_N)$. The interpolation problem is then to find a trigonometric polynomial s_n of lowest degree such that $s_n(x_j) = f_j$, for $j = 0, 1, \ldots, N-1$. Such polynomial has $2n+1$ coefficients. If we take $n = N/2$ (assuming N even), we have $N+1$ coefficients to be determined but only N interpolation conditions. An additional condition arises by noting that the sine term of highest wavenumber, $k = N/2$, vanishes at the equi-spaced nodes, $\sin(\frac{N}{2}x_j) = \sin(j\pi) = 0$. Thus, the coefficient $b_{N/2}$ is irrelevant for interpolation and we can set it to zero. Consequently, we look for a trigonometric polynomial of the form

$$s_{N/2}(x) = \frac{1}{2}a_0 + \sum_{k=1}^{N/2-1} (a_k \cos kx + b_k \sin kx) + \frac{1}{2}a_{N/2} \cos\left(\frac{N}{2}x\right). \tag{3.149}$$

The convenience of the $1/2$ factor in the last term will be seen in the formulas we obtain below for the coefficients.

It is conceptually and computationally simpler to work with the corresponding trigonometric polynomial in complex form

$$s_{N/2}(x) = {\sum_{k=-N/2}^{N/2}}'' c_k e^{ikx}, \tag{3.150}$$

[5] Recall $2\cos kx = e^{ik} + e^{-ik}$ and $2i \sin kx = e^{ik} - e^{-ik}$

where the double prime in the summation sign means that the first and last terms ($k = -N/2$ and $k = N/2$) have a factor of $1/2$. It is also understood that $c_{-N/2} = c_{N/2}$, which is equivalent to the $b_{N/2} = 0$ condition in (3.149).

Theorem 3.9.

$$s_{N/2}(x) = \sum_{k=-N/2}^{N/2}{}'' c_k e^{ikx} \tag{3.151}$$

interpolates $(j2\pi/N, f_j), j = 0, \ldots, N-1$ *if and only if*

$$c_k = \frac{1}{N}\sum_{j=0}^{N-1} f_j e^{-ik2\pi j/N}, \quad k = -\frac{N}{2}, \ldots, \frac{N}{2}. \tag{3.152}$$

Proof. Substituting (3.152) in (3.151) we get

$$s_{N/2}(x) = \sum_{k=-N/2}^{N/2}{}'' c_k e^{ikx} = \sum_{j=0}^{N-1} f_j \frac{1}{N} \sum_{k=-N/2}^{N/2}{}'' e^{ik(x-x_j)},$$

with $x_j = j2\pi/N$ and defining the cardinal functions

$$l_j(x) = \frac{1}{N}\sum_{k=-N/2}^{N/2}{}'' e^{ik(x-x_j)} \tag{3.153}$$

we obtain

$$s_{N/2}(x) = \sum_{j=0}^{N-1} l_j(x) f_j. \tag{3.154}$$

Note that we have written $s_{N/2}$ in a form similar to the Lagrange form of polynomial interpolation. We will prove that for j and m in the range $0, \ldots, N-1$

$$l_j(x_m) = \begin{cases} 1 & \text{for } m = j, \\ 0 & \text{for } m \neq j, \end{cases} \tag{3.155}$$

and in view of (3.154), $s_{N/2}$ satisfies the interpolation conditions.

Now,

$$l_j(x_m) = \frac{1}{N} \sum_{k=-N/2}^{N/2}{}'' e^{ik(m-j)2\pi/N} \tag{3.156}$$

and $e^{i(\pm N/2)(m-j)2\pi/N} = e^{\pm i(m-j)\pi} = (-1)^{(m-j)}$ so we can combine the first and the last term and remove the double prime from the sum:

$$\begin{aligned}
l_j(x_m) &= \frac{1}{N} \sum_{k=-N/2}^{N/2-1} e^{ik(m-j)2\pi/N} \\
&= \frac{1}{N} \sum_{k=-N/2}^{N/2-1} e^{i(k+N/2)(m-j)2\pi/N} e^{-i(N/2)(m-j)2\pi/N} \\
&= e^{-i(m-j)\pi} \frac{1}{N} \sum_{k=0}^{N-1} e^{ik(m-j)2\pi/N}.
\end{aligned}$$

Recall that (see Section 1.3)

$$\frac{1}{N} \sum_{k=0}^{N-1} e^{-ik(j-m)2\pi/N} = \begin{cases} 1 & \text{if } \left(\frac{j-m}{N}\right) \in \mathbb{Z}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.157}$$

Then, (3.155) follows and

$$s_{N/2}(x_m) = f_m, \quad m = 0, 1, \ldots N-1. \tag{3.158}$$

Now suppose $s_{N/2}$ interpolates $(j2\pi/N, f_j), j = 0, \ldots, N-1$. Then, the c_k coefficients of $s_{N/2}$ satisfy

$$\sum_{k=-N/2}^{N/2}{}'' c_k e^{ik2\pi j/N} = f_j, \quad j = 0, 1, \ldots, N-1. \tag{3.159}$$

Since $c_{-N/2} = c_{N/2}$, we can write (3.159) equivalently as the linear system

$$\sum_{k=-N/2}^{N/2-1} c_k e^{ik2\pi j/N} = f_j, \quad j = 0, 1, \ldots, N-1. \tag{3.160}$$

From the discrete orthogonality of the complex exponential (3.157), it follows that the matrix of coefficients of (3.160) has orthogonal columns and hence it is nonsingular. Therefore, (3.160) has a unique solution and thus the c_k coefficients must be those given by (3.152). □

Using the relations $c_0 = \frac{1}{2}a_0$, $c_k = \frac{1}{2}(a_k - ib_k)$, $c_{-k} = \bar{c}_k$, we find that

$$s_{N/2}(x) = \frac{1}{2}a_0 + \sum_{k=1}^{N/2-1}(a_k \cos kx + b_k \sin kx) + \frac{1}{2}a_{N/2}\cos\left(\frac{N}{2}x\right)$$

interpolates $(j2\pi/N, f_j), j = 0, \ldots, N-1$ if and only if

$$a_k = \frac{2}{N}\sum_{j=0}^{N-1} f_j \cos kx_j, \quad k = 0, 1, \ldots, N/2, \tag{3.161}$$

$$b_k = \frac{2}{N}\sum_{j=0}^{N-1} f_j \sin kx_j, \quad k = 1, \ldots, N/2-1. \tag{3.162}$$

A smooth periodic function f can be approximated accurately by its interpolating trigonometric polynomial of low to moderate degree. Figure 3.14(a) shows the approximation of $f(x) = \sin x\, e^{\cos x}$ on $[0, 2\pi]$ by s_4 $(N = 8)$. The graphs of f and $s_{N/2}$ are almost indistinguishable. In fact, the interpolating trigonometric polynomial $s_{N/2}$ converges uniformly to f exponentially fast as Fig. 3.14(b) demonstrates (note that the vertical axis uses a logarithmic scale).

Note also that derivatives of $s_{N/2}$ can be easily computed

$$s_{N/2}^{(p)}(x) = \sum_{k=-N/2}^{N/2}{}'' (ik)^p c_k e^{ikx}. \tag{3.163}$$

The Fourier coefficients of the p-th derivative of $s_{N/2}$ can thus be readily obtained from the DFT of f (the c_k's) and $s_{N/2}^{(p)}$ yields an accurate approximation of $f^{(p)}$ if this is smooth. We discuss the implementations details of this approach in Section 6.4.

Let us go back to the complex, interpolating trigonometric polynomial (3.150). Its coefficients c_k are periodic of period N,

$$c_{k+N} = \frac{1}{N}\sum_{j=0}^{N-1} f_j e^{-i(k+N)x_j} = \frac{1}{N}\sum_{j=0}^{N-1} f_j e^{-ikx_j} e^{-ij2\pi} = c_k. \tag{3.164}$$

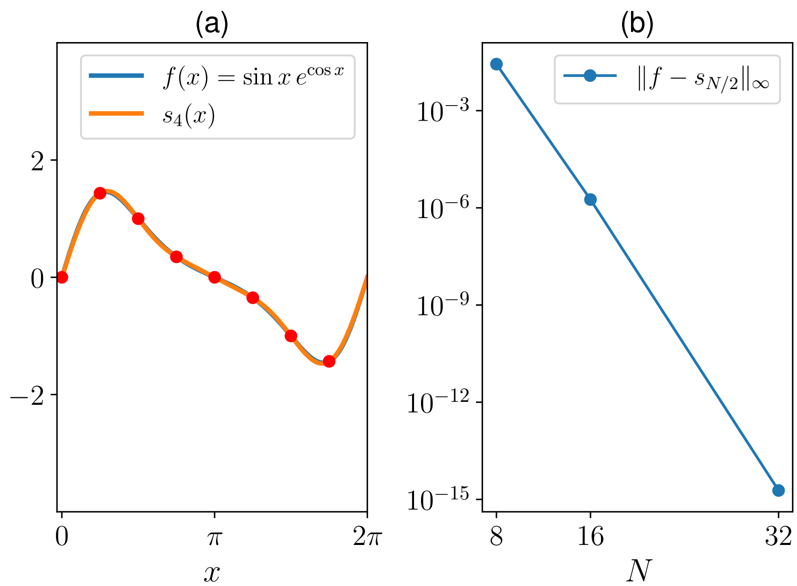

Figure 3.14: (a) $f(x) = \sin x\ e^{\cos x}$ and its interpolating trigonometric polynomial $s_4(x)$ and (b) the maximum error $\|f - s_{N/2}\|_\infty$ for $N = 8, 16, 32$.

Now, from (3.160) we have

$$
\begin{aligned}
f_j = \sum_{k=-N/2}^{N/2-1} c_k e^{ikx_j} &= \sum_{k=-N/2}^{-1} c_k e^{ikx_j} + \sum_{k=0}^{N/2-1} c_k e^{ikx_j} \\
&= \sum_{k=N/2}^{N-1} c_k e^{ikx_j} + \sum_{k=0}^{N/2-1} c_k e^{ikx_j} \\
&= \sum_{k=0}^{N-1} c_k e^{ikx_j},
\end{aligned} \tag{3.165}
$$

where we have used that $c_{k+N} = c_k$ to shift the sum from $-N/2$ to -1 to the sum from $N/2$ to $N-1$. Combining this with the formula for the c_k's we get the discrete Fourier transform (DFT) pair

$$
c_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j e^{-ikx_j}, \quad k = 0, \ldots, N-1, \tag{3.166}
$$

$$
f_j = \sum_{k=0}^{N-1} c_k e^{ikx_j}, \qquad j = 0, \ldots, N-1. \tag{3.167}
$$

The set of coefficients (3.166) is known as the DFT of the periodic array $f_0, f_1, \ldots, f_{N-1}$ and (3.167) is called the inverse DFT. It is important to note that the DFT coefficients for $k = N/2, \ldots, N-1$ correspond to those for $k = -N/2, \ldots, -1$ of the interpolating trigonometric polynomial $s_{N/2}$.

3.12 The Fast Fourier Transform

The direct evaluation of the DFT or the inverse DFT is computationally expensive, it requires $O(N^2)$ operations. However, there is a remarkable algorithm which achieves this in merely $O(N \log_2 N)$ operations. This algorithm is known as the Fast Fourier Transform.

We now look at the main ideas of this widely used algorithm.

Let us define $d_k = Nc_k$ for $k = 0, 1, \ldots, N-1$, and $\omega_N = e^{-i2\pi/N}$. Then we can rewrite the DFT (3.166) as

$$
d_k = \sum_{j=0}^{N-1} f_j \omega_N^{kj}, \quad k = 0, 1, \ldots, N-1. \tag{3.168}
$$

Let $N = 2n$. If we split the even-numbered and the odd-numbered points we have

$$d_k = \sum_{j=0}^{n-1} f_{2j}\omega_N^{2jk} + \sum_{j=0}^{n-1} f_{2j+1}\omega_N^{(2j+1)k} \tag{3.169}$$

But

$$\omega_N^{2jk} = e^{-i2jk\frac{2\pi}{N}} = e^{-ijk\frac{2\pi}{\frac{N}{2}}} = e^{-ijk\frac{2\pi}{n}} = \omega_n^{kj}, \tag{3.170}$$

$$\omega_N^{(2j+1)k} = e^{-i(2j+1)k\frac{2\pi}{N}} = e^{-ik\frac{2\pi}{N}}e^{-i2jk\frac{2\pi}{N}} = \omega_N^k\omega_n^{kj}. \tag{3.171}$$

Thus, denoting $f_j^e = f_{2j}$ and $f_j^o = f_{2j+1}$, we get

$$d_k = \sum_{j=0}^{n-1} f_j^e\omega_n^{jk} + \omega_N^k \sum_{j=0}^{n-1} f_j^o\omega_n^{jk} \tag{3.172}$$

We have reduced the problem to two DFT's of size $n = \frac{N}{2}$ plus N multiplications (and N sums). The numbers ω_N^k, $k = 0, 1, \ldots, N-1$ depend only on N so they can be precomputed once and stored for other DFT's of the same size N.

If $N = 2^p$, for p positive integer, we can repeat the process to reduce each of the DFT's of size n to a pair of DFT's of size $n/2$ plus n multiplications (and n additions), etc. We can do this p times so that we end up with 1-point DFT's, which require no multiplications!

Let us count the number of operations in the FFT algorithm. For simplicity, let is count only the number of multiplications (the numbers of additions is of the same order). Let m_N be the number of multiplications to compute the DFT for a periodic array of size N and assume that $N = 2^p$. Then

$$\begin{aligned}
m_N &= 2m_{\frac{N}{2}} + N \\
&= 2m_{2^{p-1}} + 2^p \\
&= 2(2m_{2^{p-2}} + 2^{p-1}) + 2^p \\
&= 2^2 m_{2^{p-2}} + 2 \cdot 2^p \\
&= \cdots \\
&= 2^p m_{2^0} + p \cdot 2^p = p \cdot 2^p \\
&= N\log_2 N,
\end{aligned}$$

where we have used that $m_{2^0} = m_1 = 0$ (no multiplication is needed for DFT of 1 point). To illustrate the savings, if $N = 2^{20}$, with the FFT we can obtain the DFT (or the inverse DFT) in order 20×2^{20} operations, whereas the direct methods requires order 2^{40}, i.e. a factor of $\frac{1}{20}2^{20} \approx 52429$ more operations. The FFT can also be implemented efficiently when N is the product of small primes.

3.13 The Chebyshev Interpolant and the DCT

We take now a closer look at polynomial interpolation of a function f in $[-1, 1]$[6] at the Chebyshev nodes

$$x_j = \cos\left(\frac{j\pi}{n}\right), \quad j = 0, 1, \ldots, n. \tag{3.173}$$

The unique interpolating polynomial $p_n \in \mathbb{P}_n$ of f at the $n + 1$ Chebyshev nodes, which we will call the *Chebyshev interpolant*, can be evaluated efficiently using its barycentric representation (Section 3.3). However, there is another representation of p_n that is also computationally efficient and useful for obtaining fast converging methods for integration and differentiation. This alternative representation is based on an expansion of Chebyshev polynomials and the DCT, the discrete cosine transform.

Since $p_n \in \mathbb{P}_n$, there are unique coefficients $c_0, c_1, \ldots, c_n$ such that

$$p_n(x) = \frac{1}{2}c_0 + \sum_{k=1}^{n-1} c_k T_k(x) + \frac{1}{2}c_n T_n(x) := {\sum_{k=0}^{n}}'' c_k T_k(x). \tag{3.174}$$

The $1/2$ factor for $k = 0, n$ is introduced for convenience to have one formula for all the c_k's, as we will see below. Under the change of variable $x = \cos\theta$, for $\theta \in [0, \pi]$ we get

$$p_n(\cos\theta) = \frac{1}{2}c_0 + \sum_{k=1}^{n-1} c_k \cos k\theta + \frac{1}{2}c_n \cos n\theta. \tag{3.175}$$

Let $\Pi_n(\theta) = p_n(\cos\theta)$ and $F(\theta) = f(\cos\theta)$. By extending F evenly over $[\pi, 2\pi]$ and using Theorem 3.9, we conclude that $\Pi_n(\theta)$ interpolates $F(\theta) =$

[6]For a function defined in an interval $[a, b]$ the change of variables $t = \frac{1}{2}(1-x)a + \frac{1}{2}(1+x)b$ could be used.

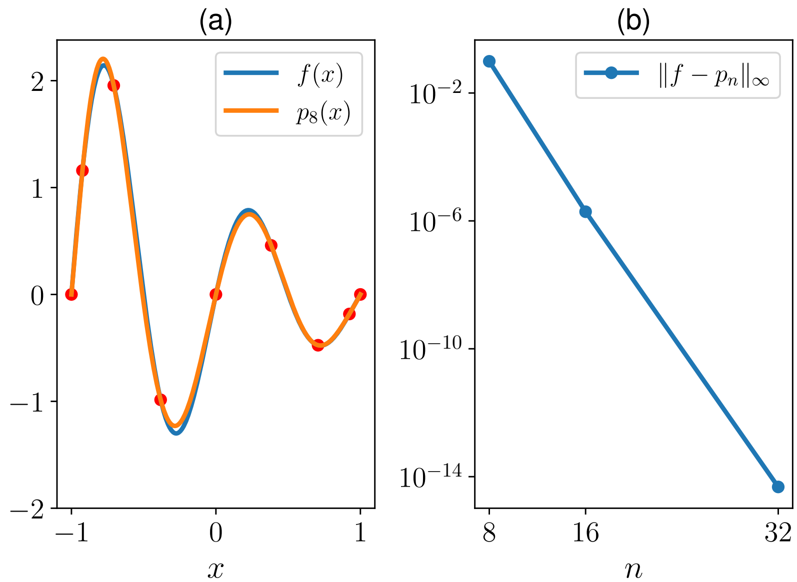

Figure 3.15: (a) $f(x) = \sin(2\pi x)\, e^{-x}$ and its Chebychev interpolant $p_8(x)$ and (b) the maximum error $\|f - p_n\|_\infty$ for $n = 8, 16, 32$.

$f(\cos\theta)$ at the equally spaced points $\theta_j = j\pi/n,\ j = 0, 1, ...n$ if and only if

$$c_k = \frac{2}{n} \sum_{j=0}^{n}{}'' F(\theta_j) \cos k\theta_j, \qquad k = 0, 1, .., n. \tag{3.176}$$

These are the (type I) Discrete Cosine Transform (DCT) coefficients of F and we can compute them efficiently in $O(n \log_2 n)$ operations with the fast DCT, an FFT-based algorithm which exploits that F is even and real. [7] Figure 3.15(a) presents a plot of $f(x) = \sin(2\pi x)e^{-x}$ on $[-1, 1]$ and its Chebyshev interpolant p_8, whose coefficients c_k were obtained with the fast DCT. The two graphs almost overlap. Figure. 3.15(a) shows the fast, uniform convergence of the Chebychev interpolant. With just $n = 32$, near machine precision is obtained.

One application of Chebyshev interpolation and its connection with the DCT is the Clenshaw-Curtis quadrature, which we consider in Section 7.4.

[7] Using the full FFT requires extending F evenly to $[\pi, 2\pi]$, doubling the size of the arrays, and is thus computationally less efficient than the fast DCT.

3.14 Bibliographic Notes

Section 3.1 The simple proof of existence and uniqueness of the interpolating polynomial using (3.1) appears in the book by Davis [Dav75].

Section 3.2. Rivlin [Riv81] provides a derivation of the bound for the Lebesgue constant $\Lambda > \frac{2}{\pi^2} \log n - 1$. There is a sharper estimate $\Lambda > \frac{2}{\pi} \log n - c$ for some positive constant c due to Erdös [Erd64]. Davis [Dav75] has a deeper discussion of the issue of convergence given a triangular system of nodes. He points to the independent discovery by Faber and Bernstein in 1914 that given any triangular system in advance, it is possible to construct a continuous function for which the interpolating polynomial does not converge uniformly to this function.

Section 3.3. Berrut and Trefethen [BT04] provide an excellent review of barycentric interpolation, including a discussion of numerical stability and historical notes. They also show that in most cases this should be the method of choice for repeated evaluation of the interpolating polynomial. For historical reasons explained in [BT04], barycentric interpolation has rarely appeared in numerical analysis textbooks. Among the rare exceptions are the books by Schwarz [Sch89], Greenbaum and Chartier [GC12], and Gautschi [Gau11]. Our presentation here follows the latter. Our derivation of the barycentric weights for the Chebyshev nodes follows that of Salzer [Sal72].

Section 3.4. Divided differences receive considerable attention as an interpolation topic in most classical, numerical analysis textbooks (see for example [CdB72, Hil13, RR01, IK94]). Here, we keep our presentation to a minimum to devote more space to barycentric interpolation (which is more efficient for the evaluation of the interpolating polynomial) and to other interpolation topics not extensively treated in most traditional textbooks. The emphasis of this section is to establish the connection of divided differences with the derivatives of f and later to Hermite interpolation.

Section 3.5. The elegant proof of Theorem 3.3 has been attributed to Cauchy (see for example [Gau11]). The interpolation error in the form (3.64) was derived by Cauchy in 1840 [Cau40]. The minimization of the node polynomial $w(x) = (x - x_0) \cdots (x - x_n)$ by the zeros of T_{n+1} is covered in many textbooks (e.g. [Dav75, Hil13, Sch89, Gau11]). However, the more practical bound (3.61) for the Chebyshev nodes (the extremal points of T_n) is more rarely found. The derivation here follows that of Salzer [Sal72].

Section 3.6. Gautschi [Gau11] makes the observation that (3.64) is a tautology because $f[x_0, \ldots, x_n, x]$ involves itself the value $f(x)$ so it really reduces to a trivial identity. However, the connection of divided differences with the derivatives of f obtained from (3.64) and the Cauchy remainder has important consequences and applications; one of them is Hermite interpolation.

Section 3.7. Hermite interpolation is treated more extensively in [SB02, KC02]. Here, we make use of the notion of coincident nodes (see e.g. [Dav75]) and the connection of divided differences with derivatives to link Hermite interpolation with Newton's interpolation form.

Section 3.8. Runge [Run01] presented his famous example $f(x) = 1/(1 + x^2)$ in the interval $[-5, 5]$. Here, we have rescaled it for the interval $[-1, 1]$. Runge employs Hermite formula [Her77] for the interpolation error for the analysis of interpolation with equispaced nodes. The convergence theorem for polynomial interpolation and its proof have been adapted from [Kry12, For96].

Section 3.9 and *Section 3.10*. The canonical reference for splines is de Boor's monograph [dB78]. This interpolation subject is also excellently treated in the numerical analysis textbooks by Kincaid and Cheney [KC02], Schwarz [Sch89], and Gautschi [Gau11], whose presentations inspired these two sections. The use of (3.146) for obtaining the parameter values t_j in splines for parametric, smooth curves is proposed in [Sch89].

Section 3.11. Trigonometric interpolation appears in most modern numerical analysis textbooks, e.g. [Sch89, KC02, SB02, Sau12]. It is a central topic of spectral methods.

Section 3.12. The FFT algorithm was proposed by Cooley and Tukey [CT65] in 1965. It is now understood [HJB85] that this famous algorithm was discovered much earlier by Gauss, around 1805. The sorting out of the coefficients (not described in this text) using binary representation of the indices is provided in [CT65]. Sauer's book [Sau12] has an excellent section on the FFT and signal processing and a chapter on the DCT and compression.

Section 3.13. Despite its usefulness, Chebyshev interpolation is rarely found in introductory numerical analysis textbooks. One exception is the book by Schwarz [Sch89].

Chapter 4

Least Squares

In this chapter we study the best approximation in the L^2 or the 2-norm, which is called the *least squares approximation*. We consider both approximation of continuous functions (using the L^2 norm) and discrete (data) sets (using the Euclidean, 2-norm). The theory is the same for both settings except that integrals are replaced by sums in the latter. Throughout this chapter $\|\cdot\|$ is the (weighted) L^2 or the 2-norm, unless otherwise noted.

4.1 Least Squares for Functions

Definition 4.1. *A set of functions $\{\phi_0, ..., \phi_n\}$ defined on an interval $[a, b]$ is said to be linearly independent if*

$$c_0\phi_0(x) + c_1\phi_1(x) + \ldots c_n\phi_n(x) = 0, \quad \text{for all } x \in [a, b], \tag{4.1}$$

then $c_0 = c_1 = \ldots = c_n = 0$. Otherwise, it is said to be linearly dependent.

Example 4.1. *The set of functions $\{\phi_0, ..., \phi_n\}$, where ϕ_k is a polynomial of degree exactly k for $k = 0, 1, \ldots, n$ is linearly independent on any interval $[a, b]$. For $c_0\phi_0 + c_1\phi_1 + \ldots c_n\phi_n$ is a polynomial of degree at most n and hence $c_0\phi_0(x) + c_1\phi_1(x) + \ldots c_n\phi_n(x) = 0$ for all x in a given interval $[a, b]$ implies $c_0 = c_1 = \ldots = c_n = 0$. In particular, the set $\{1, x, \ldots, x^n\}$ is linearly independent.*

We are going to use the weighted L^2 norm. This is given in terms of the inner product

$$\langle f, g\rangle = \int_a^b f(x)\overline{g(x)}w(x)dx, \tag{4.2}$$

where $w(x) \geq 0$ for all $x \in (a, b)$ [1] and the overline denotes the complex conjugate. We have

$$\|f\| = \sqrt{\langle f, f\rangle}. \tag{4.3}$$

Definition 4.2. *Two functions f and g are orthogonal, with respect to the inner product $\langle\cdot,\cdot\rangle$, if $\langle f, g\rangle = 0$.*

Theorem 4.1. *Pythagorean Theorem. If f and g are orthogonal, then*

$$\|f + g\|^2 = \|f\|^2 + \|g\|^2. \tag{4.4}$$

Proof.

$$\begin{aligned} \|f + g\|^2 &= \langle f + g, f + g\rangle \\ &= \langle f, f\rangle + \langle f, g\rangle + \langle g, f\rangle + \langle g, g\rangle \\ &= \langle f, f\rangle + \langle g, g\rangle = \|f\|^2 + \|g\|^2. \end{aligned} \tag{4.5}$$

□

Given a continuous function f and a set of linearly independent, continuous functions $\{\phi_0, ..., \phi_n\}$ both defined on $[a, b]$, the least squares problem is to find the best approximation to f in the L^2 norm by functions in

$$W = \text{Span}\{\phi_0, ..., \phi_n\}. \tag{4.6}$$

Since W is finite-dimensional and the L^2 norm is strictly convex, we know (Theorem 2.2 and Theorem 2.3) there is a unique best approximation $f^* \in W$ to f. That is, there is a unique $f^* \in W$ such that

$$\|f - f^*\| \leq \|f - g\|, \quad \forall g \in W. \tag{4.7}$$

This best approximation f^* is called the *least squares approximation* to f (by functions in W) because it minimizes the squared error $\|f - g\|^2$ over $g \in W$. It has a geometric interpretation: the error $f - f^*$ is orthogonal to W:

$$\langle f - f^*, g\rangle = 0, \quad \forall g \in W, \tag{4.8}$$

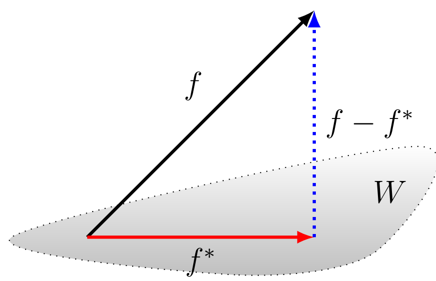

Figure 4.1: Geometric interpretation of the least squares approximation f^* to f by functions in W. The error $f - f^*$ is orthogonal to W

as Fig. 4.1 illustrates. That is, f^* is the *orthogonal projection* of f onto W. Since all functions in W are linear combinations of $\phi_0, \phi_1, \ldots, \phi_n$, this geometric characterization of the least squares approximation is equivalent to $\langle f - f^*, \phi_j \rangle = 0$ for $j = 0, 1, \ldots, n$ and writing $f^* = c_0\phi_0 + \ldots c_n\phi_n$ we obtain the *normal equations*

$$\sum_{k=0}^{n} \langle \phi_k, \phi_j \rangle c_k = \langle f, \phi_j \rangle, \quad j = 0, 1, \ldots, n. \tag{4.9}$$

We will show that this linear system of equations for $c_0, c_1, \ldots, c_n$ has a unique solution but first let's state and prove the geometric characterization of f^*.

Theorem 4.2. *The least squares approximation to f by functions in W is characterized by the geometric property (4.8).*

Proof. By uniqueness of the least squares approximation (Theorem 2.2 and Theorem 2.3)) we only need to show that if $f^* \in W$ satisfies the geometric property then it is a least squares approximation to f.

Suppose $f - f^*$ is orthogonal to W and let $g \in W$. Then, $f^* - g$ is also in W and hence orthogonal to $f - f^*$. Therefore,

$$\|f - g\|^2 = \|f - f^* + f^* - g\|^2 = \|f - f^*\|^2 + \|f^* - g\|^2, \tag{4.10}$$

where we have used the Pythagorean theorem in the last equality. From (4.10) it follows that $\|f - g\| \geq \|f - f^*\|$ for all $g \in W$. $\square$

---

[1]More precisely, we will assume $w(x) \geq 0$, $\displaystyle\int_a^b w(x)dx > 0$, and $\displaystyle\int_a^b x^k w(x)dx < +\infty$ for $k = 0, 1, \ldots$. We call such a w an admissible weight function.

We now prove that if the set $\{\phi_0, ..., \phi_n\}$ is linearly independent then there is a unique solution $c_0^*, \ldots, c_n^*$ of the normal equations (4.9), so that $f^* = c_0^*\phi_0 + \ldots c_n^*\phi_n$. Equivalently, we will show that the homogeneous system

$$\sum_{k=0}^{n} \langle \phi_k, \phi_j \rangle c_k = 0, \quad j = 0, 1, \ldots, n, \tag{4.11}$$

has only the trivial solution. Indeed,

$$\begin{aligned} \left\| \sum_{k=0}^{n} c_k \phi_k \right\|^2 &= \left\langle \sum_{k=0}^{n} c_k \phi_k, \sum_{j=0}^{n} c_j \phi_j \right\rangle \\ &= \sum_{k=0}^{n} \sum_{j=0}^{n} \langle \phi_k, \phi_j \rangle c_k \bar{c}_j \\ &= \sum_{j=0}^{n} \left(\sum_{k=0}^{n} \langle \phi_k, \phi_j \rangle c_k \right) \bar{c}_j = \sum_{j=0}^{n} 0 \, \bar{c}_j = 0. \end{aligned} \tag{4.12}$$

Therefore $\displaystyle\sum_{k=0}^{n} c_k \phi_k(x) = 0$ for all $x \in [a, b]$. By the linear independence of the set $\{\phi_0, \phi_1, \ldots, \phi_n\}$ it follows that $c_0 = c_1 = \ldots = c_n = 0$.

Orthogonality plays a central role in the least squares problem. But it is also important to keep in mind the minimization character of the solution. Indeed, if f^* is the best L^2-approximation of f in W then for any fixed $g \in W$ $J(\epsilon) = \|f - f^* + \epsilon g\|^2$ has a minimum at $\epsilon = 0$. But

$$J(\epsilon) = \|f - f^*\|^2 + \epsilon \langle f - f^*, g \rangle + \epsilon \langle g, f - f^* \rangle + \epsilon^2 \|g\|^2. \tag{4.13}$$

This is a parabola opening upwards. Hence the minimum is at its critical point. Since $J'(\epsilon) = 2\text{Re}\langle f - f^*, g \rangle + 2\epsilon \|g\|^2$ and we know the minimum is attained at $\epsilon = 0$ it follows that $\text{Re}\langle f - f^*, g \rangle = 0$. Repeating the argument for $-ig$ it follows that $\text{Im}\langle f - f^*, g \rangle = 0$ and consequently $\langle f - f^*, g \rangle = 0$.

Definition 4.3. *$\{\phi_0, ..., \phi_n\}$ is an orthogonal set if $\langle \phi_j, \phi_k \rangle = 0$ for all $j \neq k$ $(j, k = 0, 1, \ldots, n)$. If in addition $\|\phi_k\| = 1$ for $k = 0, 1, \ldots, n$, $\{\phi_0, ..., \phi_n\}$ is called an orthonormal set.*

If $\{\phi_0, ..., \phi_n\}$ is an orthogonal set of functions, then the normal equations (4.9) simplify to

$$\langle \phi_k, \phi_k \rangle c_k = \langle f, \phi_k \rangle, \quad k = 0, 1, \ldots, n, \tag{4.14}$$

which can be solved immediately to give

$$c_k = \frac{1}{\|\phi_k\|^2} \langle f, \phi_k \rangle, \quad k = 0, 1, \ldots, n. \tag{4.15}$$

These c_k's are called *(generalized) Fourier coefficients.* They are a generalization of the familiar Fourier coefficients, obtained from the set of trigonometric functions $\{1, \cos x, \sin x, \ldots, \cos nx, \sin nx\}$ or equivalently from the set $\{1, e^{ix}, e^{-ix}, \ldots, e^{inx}, e^{-inx}\}$, as we will see next. Note that the Fourier coefficients (4.15) are independent of n. If we have computed c_k, $k = 0, 1, \ldots, n$ and would like to increase n we just need to compute the new coefficients $k > n$ and reuse the previously computed ones.

4.1.1 Trigonometric Polynomial Approximation

The set $\{1, e^{ix}, e^{-ix}, \ldots, e^{inx}, e^{-inx}\}$ is an orthogonal set with the inner product (4.2) and $w \equiv 1$, on any interval of length 2π, say $[0, 2\pi]$, for

$$\langle e^{ijx}, e^{ikx} \rangle = \int_0^{2\pi} e^{i(j-k)x} dx = 0, \quad \text{for } j \neq k. \tag{4.16}$$

Thus, the least squares approximation to a function f (defined on $[0, 2\pi]$ and squared integrable), i.e. the best approximation in the L^2 norm, by a trigonometric polynomial of degree at most n (see Definition 3.3) is given by the truncated Fourier series of f:

$$f^*(x) = \sum_{k=-n}^{n} c_k e^{ikx}, \tag{4.17}$$

$$c_k = \frac{1}{2\pi} \langle f, e^{ikx} \rangle = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-ikx} dx, \quad k = 0, \pm 1, \ldots, \pm n. \tag{4.18}$$

or equivalently

$$f^*(x) = \frac{1}{2}a_0 + \sum_{k=1}^{n} \left(a_k \cos kx + b_k \sin kx\right), \tag{4.19}$$

$$a_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos kx dx, \quad k = 0, 1, \ldots, n, \tag{4.20}$$

$$b_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin kx dx, \quad k = 1, \ldots, n. \tag{4.21}$$

That is, the solution of the normal equations in this case are the (traditional) Fourier coefficients of f. Assuming f is a smooth, 2π-periodic function (with a uniformly convergent Fourier series),

$$f(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx}, \tag{4.22}$$

the squared error is given by

$$\|f - f^*\|^2 = \left\langle \sum_{|k|>n} c_k e^{ikx}, \sum_{|j|>n} c_j e^{ijx} \right\rangle = 2\pi \sum_{|k|>n} |c_k|^2. \tag{4.23}$$

If f is 2π-periodic and $f \in C^m[0, 2\pi]$ for $m \geq 1$, its Fourier coefficients decay like $|c_k| \leq A_m |k|^{-m}$ for some constant A_m [cf. (1.63)]. Then,

$$\|f - f^*\|^2 \leq 4\pi A_m^2 \sum_{k=n+1}^{\infty} \frac{1}{k^{2m}} \tag{4.24}$$

and if we use the bound

$$\sum_{k=1}^{\infty} \frac{1}{k^{2m}} < \int_{n+1}^{\infty} \frac{dx}{x^{2m}} = \frac{1}{(2m-1)(n+1)^{2m-1}} \tag{4.25}$$

we obtain

$$\|f - f^*\| \leq \frac{C_m}{(n+1)^{m-1/2}}, \tag{4.26}$$

for some constant C_m.

In practice, we approximate the Fourier coefficients (4.18) with the composite trapezoidal rule at $N = 2n$ equi-spaced-points

$$c_k \approx \tilde{c}_k = \frac{1}{N} \sum_{j=0}^{N-1} f(j2\pi/N) e^{-ikj2\pi/N}. \tag{4.27}$$

Now, substituting (4.22) with $x = j2\pi/N$

$$\begin{aligned} \tilde{c}_k &= \frac{1}{N} \sum_{j=0}^{N-1} \left(\sum_{l=-\infty}^{\infty} c_l e^{ilj2\pi/N} \right) e^{-ikj2\pi/n} \\ &= \sum_{l=-\infty}^{\infty} c_l \left(\frac{1}{N} \sum_{j=0}^{N-1} e^{i(l-k)j2\pi/N} \right) \end{aligned} \tag{4.28}$$

and using the discrete orthogonality of the complex exponential (3.157) we get

$$\tilde{c}_k = c_k + c_{k-N} + c_{k+N} + c_{k-2N} + c_{k+2N} + \ldots. \tag{4.29}$$

For computational efficiency we take $N = 2n$ and obtain the discrete Fourier coefficients $\tilde{c}_k$ for $k = -N/2, \ldots, N/2 - 1$ with the FFT, i.e. in practice we use the Fourier interpolant

$$s_{N/2}(x) = \sum_{k=-N/2}^{N/2}{}'' \, \tilde{c}_k e^{ikx} \tag{4.30}$$

instead of f^*. From (4.29), the error $|\tilde{c}_k - c_k|$ depends on the decay of the Fourier coefficients $c_{k \pm lN}$, $l = 1, 2, \ldots$, for $|k \pm lN| \geq N/2$ (given that $|k| \leq N/2$). In particular, if f is periodic and $f \in C^m[0, 2\pi]$ we can proceed as we did for c_0 in Section 1.3 to show that $|\tilde{c}_k - c_k| = O(N^{-m})$ for $k = -N/2, \ldots, N/2 - 1$. Thus, the additional error of using the Fourier interpolant instead of f^* is asymptotically the same order as the error of the least squares approximation.

4.1.2 Algebraic Polynomial Approximation

Let us consider now the least squares approximation of f by algebraic polynomials of degree at most n. If we choose $\{1, x, \ldots, x^n\}$ as a basis for $\mathbb{P}_n$

and $w \equiv 1$, the least squares approximation can be written as $f^*(x) = c_0 + c_1 x + \ldots + c_n x^n$, where the coefficients c_k, $k = 0, 1, \ldots, n$ are the solution of the normal equations (4.9). Thus, in principle we just need to solve this $(n+1) \times (n+1)$ linear system of equations. There are however two problems with this approach:

1. It is difficult to solve this linear system numerically for even moderate n because the matrix of coefficients is very sensitive to small perturbations and this sensitivity increases rapidly with n. For example, if we take the interval $[0, 1]$ the matrix of coefficients in the normal equations system is

$$\begin{bmatrix} \frac{1}{1} & \frac{1}{2} & \cdots & \frac{1}{n+1} \\ \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n+2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n+1} \end{bmatrix}. \tag{4.31}$$

   Numerical solutions in double precision (about 16 digits of accuracy) of a linear system with this matrix (known as the Hilbert matrix, of size $n + 1$) will lose all accuracy for $n \geq 11$.

2. If we want to increase the degree of the approximating polynomial we need to start all over again and solve a larger set of normal equations. That is, we cannot reuse the $c_0, c_1, \ldots, c_n$ we already found.

Fortunately, we can overcome these two problems with orthogonalization.

4.1.2.1 Gram-Schmidt Orthogonalization

Given a set of linearly independent functions $\{\phi_0, \ldots, \phi_n\}$ we can produce an orthogonal set $\{\psi_0, \ldots, \psi_n\}$ by doing the Gram-Schmidt procedure:

$$
\begin{aligned}
&\psi_0 = \phi_0, \\
&\psi_1 = \phi_1 - r_{01}\psi_0, \\
&\qquad \langle \psi_1, \psi_0 \rangle = 0 \Rightarrow r_{01} = \frac{\langle \psi_0, \phi_1 \rangle}{\langle \psi_0, \psi_0 \rangle} \\
&\psi_2 = \phi_2 - r_{02}\psi_0 - r_{12}\psi_1, \\
&\qquad \langle \psi_2, \psi_0 \rangle = 0 \Rightarrow r_{02} = \frac{\langle \psi_0, \phi_2 \rangle}{\langle \psi_0, \psi_0 \rangle} \\
&\qquad \langle \psi_2, \psi_1 \rangle = 0 \Rightarrow r_{12} = \frac{\langle \psi_1, \phi_2 \rangle}{\langle \psi_1, \psi_1 \rangle}, \\
&\text{etc.}
\end{aligned}
$$

We can write this procedure recursively as

$$
\begin{aligned}
&\psi_0 = \phi_0, \\
&\text{For } k = 1, \ldots, n \\
&\psi_k = \phi_k - \sum_{j=0}^{k-1} r_{jk}\psi_j, \quad r_{jk} = \frac{\langle \psi_j, \phi_k, \rangle}{\langle \psi_j, \psi_j \rangle}.
\end{aligned}
\tag{4.32}
$$

4.1.2.2 Orthogonal Polynomials

Let us take the set $\{1, x, \ldots, x^n\}$ on an interval $[a, b]$. We can use the Gram-Schmidt process to obtain an orthogonal set $\{\psi_0, ..., \psi_n\}$ of polynomials with respect to the inner product (4.2). Each ψ_k, $k = 0, \ldots, n$, is a polynomial of degree k, determined up to a multiplicative constant (orthogonality is not changed). Suppose we select ψ_k, $k = 0, 1, \ldots, n$ to be monic, i.e. the coefficient of x^k is 1. Then, $\psi_{k+1} - x\psi_k$ is a polynomial of degree at most k and we can write

$$
\psi_{k+1} - x\psi_k = \sum_{j=0}^{k} c_j\psi_j, \tag{4.33}
$$

for some coefficients c_j, $j = 0, \ldots, k$. Taking the inner product of (4.33) with ψ_m for $m = 0, \ldots, k-2$ we get

$$-\langle \psi_k, x\psi_m \rangle = c_m \langle \psi_m, \psi_m \rangle, \quad m = 0, \ldots, k-2.$$

But the left hand side is zero because $x\psi_m \in \mathbb{P}_{k-1}$ and hence it is orthogonal to ψ_k. Therefore, $c_j = 0$ for $j = 0, \ldots, k-2$. Setting $\alpha_k = -c_k$ and $\beta_k = -c_{k-1}$ (4.33) simplifies to

$$\psi_{k+1} - x\psi_k = -\alpha_k \psi_k - \beta_k \psi_{k-1}. \tag{4.34}$$

Taking the inner product of this expression with ψ_k and using orthogonality we get

$$-\langle x\psi_k, \psi_k \rangle = -\alpha_k \langle \psi_k, \psi_k \rangle$$

and therefore

$$\alpha_k = \frac{\langle x\psi_k, \psi_k \rangle}{\langle \psi_k, \psi_k \rangle}.$$

Similarly, taking the inner product of (4.34) with ψ_{k-1} we obtain

$$-\langle x\psi_k, \psi_{k-1} \rangle = -\beta_k \langle \psi_{k-1}, \psi_{k-1} \rangle$$

but $\langle x\psi_k, \psi_{k-1} \rangle = \langle \psi_k, x\psi_{k-1} \rangle$ and $x\psi_{k-1} = \psi_k + p_{k-1}$, where $p_{k-1} \in \mathbb{P}_{k-1}$. Then,

$$\langle \psi_k, x\psi_{k-1} \rangle = \langle \psi_k, \psi_k \rangle + \langle \psi_k, p_{k-1} \rangle = \langle \psi_k, \psi_k \rangle,$$

where we have used orthogonality in the last equation. Therefore,

$$\beta_k = \frac{\langle \psi_k, \psi_k \rangle}{\langle \psi_{k-1}, \psi_{k-1} \rangle}.$$

Collecting the results we obtain a three-term recursion formula

$$\psi_0(x) = 1, \tag{4.35}$$

$$\psi_1(x) = x - \alpha_0, \quad \alpha_0 = \frac{\langle x\psi_0, \psi_0 \rangle}{\langle \psi_0, \psi_0 \rangle} \tag{4.36}$$

and for $k = 1, \ldots n$

$$\alpha_k = \frac{\langle x\psi_k, \psi_k \rangle}{\langle \psi_k, \psi_k \rangle}, \quad \beta_k = \frac{\langle \psi_k, \psi_k \rangle}{\langle \psi_{k-1}, \psi_{k-1} \rangle}, \tag{4.37}$$

$$\psi_{k+1}(x) = (x - \alpha_k)\psi_k(x) - \beta_k \psi_{k-1}(x). \tag{4.38}$$

If the interval is symmetric with respect to the origin, $[-a, a]$, and the weight function is even, $w(-x) = w(x)$, the orthogonal polynomials have parity, i.e. $\psi_k(x) = (-1)^k \psi_k(-x)$. This follows from the simple change of variables $y = -x$. Define $\widetilde{\psi}_j(x) = (-1)^j \psi_j(-x)$. Then, for $j \neq k$

$$
\begin{aligned}
\langle \widetilde{\psi}_j, \widetilde{\psi}_k \rangle &= \int_{-a}^{a} \widetilde{\psi}_j(x) \widetilde{\psi}_k(x) w(x) dx \\
&= (-1)^{j+k} \int_{-a}^{a} \psi_j(-x) \psi_k(-x) w(x) dx \\
&= (-1)^{j+k} \int_{-a}^{a} \psi_j(y) \psi_k(y) w(y) dy = (-1)^{j+k} \langle \psi_j, \psi_k \rangle = 0.
\end{aligned}
\tag{4.39}
$$

Since the orthogonal polynomials are defined up to a multiplicative constant and we have fixed that by choosing them to be monic, we conclude that $\widetilde{\psi}_k = \psi_k$, i.e. $\psi_k(x) = (-1)^k \psi_k(-x)$, for $k = 0, 1, \ldots, n$.

Example 4.2. *Let $[a, b] = [-1, 1]$ and $w(x) \equiv 1$. The corresponding orthogonal polynomials are known as the* **Legendre polynomials** *and are used in a variety of numerical methods. Because of the interval and weight function symmetry ψ_k^2 is even and $x\psi_k^2 w$ is odd for all k. Consequently, $\alpha_k = 0$ for all k.*

We have $\psi_0(x) = 1$ and $\psi_1(x) = x$. We can now use the three-term recursion (4.38) to obtain

$$\beta_1 = \frac{\int_{-1}^{1} x^2 dx}{\int_{-1}^{1} dx} = 1/3$$

and $\psi_2(x) = x^2 - \frac{1}{3}$. Now for $k = 2$ we get

$$\beta_2 = \frac{\int_{-1}^{1} (x^2 - \frac{1}{3})^2 dx}{\int_{-1}^{1} x^2 dx} = 4/15$$

and $\psi_3(x) = x(x^2 - \frac{1}{3}) - \frac{4}{15}x = x^3 - \frac{3}{5}x$. We now collect Legendre polynomials *we have found:*

$$\begin{aligned}\psi_0(x) &= 1,\\ \psi_1(x) &= x,\\ \psi_2(x) &= x^2 - \frac{1}{3},\\ \psi_3(x) &= x^3 - \frac{3}{5}x.\end{aligned}$$

Example 4.3. *The* **Hermite polynomials** *are the orthogonal polynomials in $(-\infty, \infty)$ with the weight function*[2] $w(x) = e^{-x^2/2}$. *Again, due to symmetry $\alpha_k = 0$, $\forall k \in \mathbb{N}$. Let us find the first few Hermite polynomials. We have $\psi_0(x) \equiv 1$, $\psi_1(x) = x$. Now,*

$$\beta_1 = \frac{\displaystyle\int_{-\infty}^{\infty} x^2 e^{-x^2/2} dx}{\displaystyle\int_{-\infty}^{\infty} e^{-x^2/2} dx} = \frac{\sqrt{2\pi}}{\sqrt{2\pi}} = 1, \tag{4.40}$$

and so $\psi_2(x) = x^2 - 1$.

$$\beta_2 = \frac{\displaystyle\int_{-\infty}^{\infty} (x^2-1)^2 e^{-x^2/2} dx}{\displaystyle\int_{-\infty}^{\infty} x^2 e^{-x^2/2} dx} = \frac{2\sqrt{2\pi}}{\sqrt{2\pi}} = 2, \tag{4.41}$$

and $\psi_3(x) = x(x^2-1) - 2x = x^3 - 3x$. Thus, the first 4 Hermite polynomials are

$$\begin{aligned}\psi_0(x) &= 1,\\ \psi_1(x) &= x,\\ \psi_2(x) &= x^2 - 1,\\ \psi_3(x) &= x^3 - 3x.\end{aligned}$$

Example 4.4. Chebyshev polynomials

We introduced in Section 2.4 the Chebyshev polynomials. As we have seen, they have remarkable properties. We now add one more important property: orthogonality.

---

[2]There is an alternative definition with the weight $w(x) = e^{-x^2}$

The Chebyshev polynomials are orthogonal with respect to the weight function $w(x) = 1/\sqrt{1-x^2}$. *Indeed, recall* $T_k(x) = \cos k\theta$, $(x = \cos\theta, \theta \in [0,\pi])$. *Then,*

$$\langle T_j, T_k \rangle = \int_{-1}^{1} T_j(x) T_k(x) \frac{dx}{\sqrt{1-x^2}} = \int_0^{\pi} \cos j\theta \cos k\theta d\theta \tag{4.42}$$

and since $2\cos j\theta \cos k\theta = \cos(j+k)\theta + \cos(j-k)\theta$, *we get for* $j \neq k$

$$\langle T_j, T_k \rangle = \frac{1}{2}\left[\frac{1}{j+k}\sin(j+k)\theta + \frac{1}{j-k}\sin(j-k)\theta\right]\Bigg|_0^{\pi} = 0. \tag{4.43}$$

Moreover, using $2\cos^2 k\theta = 1 + \cos 2k\theta$ *we obtain* $\langle T_k, T_k \rangle = \pi/2$ *for* $k > 0$ *and* $\langle T_0, T_0 \rangle = \pi$. *Therefore,*

$$\langle T_j, T_k \rangle = \begin{cases} 0 & \text{for } j \neq k, \\ \frac{\pi}{2} & \text{for } j = k > 0, \\ \pi & \text{for } j = k = 0. \end{cases} \tag{4.44}$$

Finding α_k and β_k in the three-term recursion formula (4.38) is in general a tedious process and is limited to our ability to evaluate the corresponding integrals. Fortunately, the recursion coefficients are known for several orthogonal polynomials (e.g. Legendre, Hermite, Chebyshev, Laguerre). Moreover, in the discrete case, when the integrals are replaced by sums, α_k and β_k can be evaluated directly with a simple computer code. Lastly, as we will see in Section 7.3.2, the three-term recursion formula can be used to cast the problem of finding the zeros of orthogonal polynomials into an eigenvalue problem more appropriate for computation.

Theorem 4.3. *The zeros of orthogonal polynomials are real, simple, and they all lie in* (a, b).

Proof. Indeed, $\psi_k(x)$ is orthogonal to $\psi_0(x) = 1$ for each $k \geq 1$, thus

$$\int_a^b \psi_k(x) w(x) dx = 0 \tag{4.45}$$

i.e. ψ_k has to change sign in $[a, b]$ so it has a zero, say $x_1 \in (a, b)$. Suppose x_1 is not a simple zero, then $q(x) = \psi_k(x)/(x - x_1)^2$ is a polynomial of degree $k - 2$ and so

$$0 = \langle \psi_k, q \rangle = \int_a^b \frac{\psi_k^2(x)}{(x - x_1)^2} w(x) dx > 0,$$

which is of course impossible. Assume that $\psi_k(x)$ has only l zeros in (a, b), $x_1, \ldots, x_l$. Then $\psi_k(x)(x - x_1) \cdots (x - x_l) = q_{k-l}(x)(x - x_1)^2 \cdots (x - x_l)^2$, where $q_{k-l}(x)$ is a polynomial of degree $k - l$ which does not change sign in $[a, b]$. Then

$$\langle \psi_k, (x - x_1) \cdots (x - x_l) \rangle = \int_a^b q_{k-j}(x)(x - x_1)^2 \cdots (x - x_l)^2 w(x) dx \neq 0$$

but $\langle \psi_k, (x - x_1) \cdots (x - x_l) \rangle = 0$ for $l < k$. Therefore $l = k$. □

4.1.3 Convergence of Least Squares by Orthogonal Polynomials

The three-term recursion formula allows to generate sets of orthogonal polynomials $\{\psi_0, \psi_1, \ldots, \psi_n\}$ for any $n \in \mathbb{N}$. A natural question is if the least squares approximation improves with increasing n.

Given $f \in C[a, b]$, let us denote by s_n the least squares approximation to f by the linear span of the first $n+1$ orthogonal polynomials $\{\psi_0, \psi_1, \ldots, \psi_n\}$, i.e.

$$s_n = \sum_{k=0}^{n} \frac{\langle f, \psi_k \rangle}{\|\phi_k\|^2} \psi_k. \tag{4.46}$$

Since s_n is the best approximation in the L^2 norm

$$\|f - s_n\| \leq \|f - p_n^*\|, \tag{4.47}$$

where p_n^* is the best *uniform* (i.e. sup norm) approximation to f in $\mathbb{P}_n$. Now, for any $g \in C[a, b]$

$$\|g\|^2 = \langle g, g \rangle = \int_a^b |g(x)|^2 w(x) dx \leq \|g\|_\infty^2 \int_a^b w(x) dx, \tag{4.48}$$

and thus $\|g\| \leq C\|g\|_\infty$. Together with (4.47) this implies

$$\|f - s_n\| \leq C\|f - p_n^*\|_\infty. \tag{4.49}$$

By Weierstrass approximation theorem $\|f - p_n^*\|_\infty \to 0$ as $n \to \infty$. Therefore $\|f - s_n\| \to 0$ as $n \to \infty$. Note that this does not imply $\|f - s_n\|_\infty \to 0$ as $n \to \infty$. In fact, it is generally not true for continuous functions.

Formally, to each $f \in C[a,b]$ we can assign an orthogonal polynomial expansion

$$f \sim \sum_{k=0}^{\infty} \frac{\langle f, \psi_k \rangle}{\|\phi_k\|^2} \psi_k. \tag{4.50}$$

The partial sums of this expansion are precisely the least squares approximations of f.

4.1.4 Chebyshev Expansions

The set of Chebyshev polynomials $\{T_0, T_1, \ldots, T_n\}$ is orthogonal with respect to the inner product

$$\langle f, g \rangle = \int_{-1}^{1} f(x) g(x) \frac{1}{\sqrt{1-x^2}} dx. \tag{4.51}$$

Given $f \in C[-1,1]$, the least squares approximation s_n, in the norm defined by the inner product (4.51), by polynomials of degree at most n is given by

$$s_n(x) = {\sum_{k=0}^{n}}' c_k T_k(x), \quad x \in [-1,1], \tag{4.52}$$

where

$$c_k = \frac{2}{\pi} \langle f, T_k \rangle = \frac{2}{\pi} \int_{-1}^{1} f(x) T_k(x) \frac{1}{\sqrt{1-x^2}} dx, \tag{4.53}$$

for $k = 0, 1, \ldots, n$, and the prime in the summation means the $k = 0$ term has a factor of $1/2$, i.e. $s_n = \frac{1}{2} c_0 + c_1 T_1 + \ldots c_n T_n$.

It can be shown that if f is Lipschitz, then $\|f - s_n\|_\infty \to 0$ as $n \to \infty$ and we can write

$$f(x) = {\sum_{k=0}^{\infty}}' c_k T_k(x), \quad x \in [-1,1], \tag{4.54}$$

where $c_k = \frac{2}{\pi} \langle f, T_k \rangle, k = 0, 1, \ldots$. The right hand side of (4.54) is called the *Chebyshev expansion* of f.

Assuming f is smooth and using the orthogonality of the Chebyshev polynomials we have

$$\|f - s_n\|^2 = \left\langle \sum_{k=n+1}^{\infty} c_k T_k, \sum_{k=n+1}^{\infty} c_k T_k \right\rangle = \frac{\pi}{2} \sum_{k=n+1}^{\infty} |c_k|^2. \tag{4.55}$$

Thus, the least squares error depends on the rate of decay of the Chebyshev coefficients c_k for $k \geq n+1$.

There is a clear parallel with Fourier series. With the change of variables $x = \cos\theta$, (4.53) becomes

$$c_k = \frac{2}{\pi} \int_0^{\pi} f(\cos\theta) \cos k\theta d\theta. \tag{4.56}$$

If f is smooth so is $F(\theta) = f(\cos\theta)$ as a function of θ. Moreover, the odd derivatives of F vanish at $\theta = 0$ and $\theta = \pi$ so that two successive integrations by parts of (4.56) give

$$\int_0^{\pi} F(\theta) \cos k\theta d\theta = -\frac{1}{k} \int_0^{\pi} F'(\theta) \sin\theta d\theta = -\frac{1}{k^2} \int_0^{\pi} F''(\theta) \cos\theta d\theta. \tag{4.57}$$

Thus, if $f \in C^m[-1,1]$ we can perform m integrations by parts to conclude that $|c_k| \leq A_m/k^m$ $(k > 0)$ for some constant A_m. Finally, by (4.24)-(4.26) we obtain

$$\|f - s_n\| \leq C_m (n+1)^{-m+1/2}, \tag{4.58}$$

for some constant C_m.

Often in applications, the Chebyshev interpolant is used instead of the least squares approximation. The coefficients (4.56) are approximated with the composite trapezoidal rule (6.34) at equi-spaced points in θ and computed efficiently with the fast DCT as pointed out in Section 3.13. The error made by this approximation depends again on the high wavenumber decay of the Chebyshev coefficients. Indeed

$$\begin{aligned}
\tilde{c}_k &= \frac{2}{n} {\sum_{j=0}^{n}}'' f(\cos\theta_j) \cos k\theta_j \\
&= \frac{2}{n} {\sum_{j=0}^{n}}'' \left({\sum_{l=0}^{\infty}}' c_l \cos l\theta_j \right) \cos k\theta_j \\
&= {\sum_{l=0}^{\infty}}' c_l \left(\frac{2}{n} {\sum_{j=0}^{n}}'' \cos k\theta_j \cos l\theta_j \right),
\end{aligned} \tag{4.59}$$

where $\theta_j = j\pi/n$ and we employed in the second equality the Chebyshev expansion of f at $x = \cos\theta_j$. Now,

$$\begin{aligned}\sum_{j=0}^{n}{}'' \cos k\theta_j \cos l\theta_j &= \frac{1}{2}\sum_{j=0}^{2n-1} \cos k\theta_j \cos l\theta_j \\ &= \frac{1}{4}\sum_{j=0}^{2n-1} \left[\cos(k+l)\theta_j + \cos(k-l)\theta_j\right].\end{aligned} \tag{4.60}$$

Then, by the discrete orthogonality of the complex exponential (3.157) we obtain the discrete orthogonality of the Chebyshev polynomials:

$$\sum_{j=0}^{n}{}'' \cos k\theta_j \cos l\theta_j = \begin{cases} n/2 & \text{if either } \frac{k+l}{2n} \in \mathbb{Z} \text{ or } \frac{k-l}{2n} \in \mathbb{Z}, \\ n & \text{if both } \frac{k+l}{2n} \in \mathbb{Z} \text{ and } \frac{k-l}{2n} \in \mathbb{Z}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.61}$$

Using this in (4.59) it follows that

$$\tilde{c}_k = c_k + c_{2n-k} + c_{2n+k} + c_{4n-k} + c_{4n+k} + \ldots \tag{4.62}$$

for $k = 0, 1, \ldots, n$. Thus, a bound for the error $|\tilde{c}_k - c_k|$ can be obtained from the asymptotic decay of the Chebyshev coefficients, just as in the Fourier case.

4.1.5 Decay of Chebyshev Coefficients for Analytic Functions

If we extend $F(\theta) = f(\cos\theta)$ evenly to $[\pi, 2\pi]$, $F(\theta) = F(2\pi - \theta)$, $\theta \in [\pi, 2\pi]$, we get

$$c_k = \frac{1}{\pi}\int_0^{2\pi} f(\cos\theta)\cos k\theta d\theta. \tag{4.63}$$

In other words, the Chebyshev expansion of $f(x)$ is the (cosine) Fourier expansion of $f(\cos\theta)$.

To estimate the rate of decay of the Chebyshev coefficients we are going to go to the complex plane. Letting $z = e^{i\theta}$, $\cos\theta = \frac{1}{2}(z + 1/z)$, we turn (4.63) into

$$c_k = \frac{1}{2\pi i}\oint_{|z|=1} f\left(\frac{z+1/z}{2}\right)\left(z^k + 1/z^k\right)\frac{dz}{z}. \tag{4.64}$$

The transformation

$$w(z) = \frac{1}{2}\left(z + \frac{1}{z}\right) \tag{4.65}$$

maps the unit circle $|z| = 1$ into $[-1, 1]$, twice. On the other hand, for a circle $|z| = \rho$ with $\rho \neq 1$ we have

$$w(\rho e^{i\theta}) = \frac{1}{2}\left(\rho + \frac{1}{\rho}\right)\cos\theta + i\frac{1}{2}\left(\rho - \frac{1}{\rho}\right)\sin\theta. \tag{4.66}$$

Writing $w = u + iv$ we get

$$\frac{u^2}{\left[\frac{1}{2}\left(\rho + \rho^{-1}\right)\right]^2} + \frac{v^2}{\left[\frac{1}{2}\left(\rho - \rho^{-1}\right)\right]^2} = 1, \tag{4.67}$$

which is the equation of an ellipse $\mathcal{E}_\rho$ with major and minor semi-axes $\frac{1}{2}\left(\rho + \rho^{-1}\right)$ and $\frac{1}{2}\left(\rho - \rho^{-1}\right)$, respectively and foci at $(\pm 1, 0)$. By symmetry, (4.65) maps the circle $|z| = 1/\rho$ also into the ellipse $\mathcal{E}_\rho$.

Theorem 4.4. *If f is analytic on and inside the ellipse $\mathcal{E}_\rho$, for some $\rho > 1$, then*

$$|c_k| \leq \frac{C}{\rho^k}. \tag{4.68}$$

Proof. From (4.64),

$$\begin{aligned} |c_k| \leq & \left| \frac{1}{2\pi i} \oint_{|z|=1/\rho} f\left(\frac{z + 1/z}{2}\right) z^{k-1} dz \right| \\ & + \left| \frac{1}{2\pi i} \oint_{|z|=\rho} f\left(\frac{z + 1/z}{2}\right) z^{-k-1} dz \right|, \end{aligned} \tag{4.69}$$

where we have used contour deformation (a consequence of Cauchy's theorem) to change the integration paths. Each term on the right hand side of (4.69) is bounded by $M\rho^{-k}$, where $M = \max_{z \in \mathcal{E}_\rho} |f(z)|$. □

4.1.6 Splines

We have used splines for interpolation but we could also use them to approximate, in the least squares sense, a continuous function on an interval $[a, b]$.

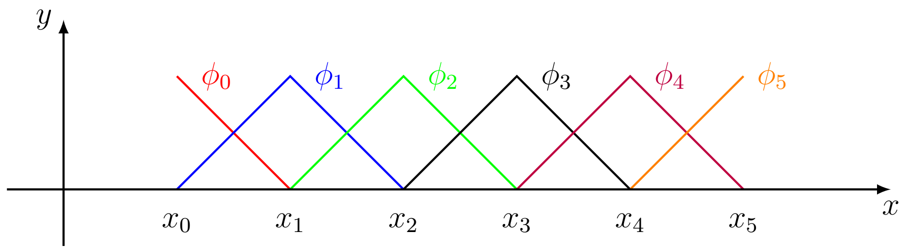

Figure 4.2: Basis "hat" functions ($n = 5$, equi-spaced nodes) for $\mathbb{S}^1_\Delta$.

As an illustration we look now at the approximation by splines of degree 1, $\mathbb{S}^1_\Delta$, i.e. continuous, piecewise linear functions. Recall that, given a partition $\Delta = \{a = x_0 < x_1 \ldots < x_n = b\}$, the set $\mathbb{S}^k_\Delta$ of splines of degree k (see Definition 3.1) is a subspace of $C^{k-1}[a, b]$ of dimension $n + k$.

Set $x_{-1} = x_0$ and $x_{n+1} = x_n$. The following set of "hat" functions

$$\phi_j(x) = \begin{cases} \dfrac{x - x_{j-1}}{x_j - x_{j-1}}, & \text{for } x \in [x_{j-1}, x_j], \\ \dfrac{x_{j+1} - x}{x_{j+1} - x_j}, & \text{for } x \in [x_j, x_{j+1}], \\ 0, & \text{otherwise}, \end{cases} \qquad j = 0, 1, \ldots, n \tag{4.70}$$

is a convenient basis for $\mathbb{S}^1_\Delta$. Figure 4.2 plots these functions for an equi-spaced partition with $n = 5$. Note that ϕ_0 and ϕ_n are only half "hat" functions. The first and the second parts of their definition (4.70), respectively, should be disregarded. Clearly, $\phi_j \in \mathbb{S}^1_\Delta$ for all j. $\{\phi_0, \phi_1, \ldots, \phi_n\}$ is not an orthogonal set but each function is nonzero only in a small region (small support) and $\phi_j(x_i) = \delta_{ij}$, for $i, j = 0, 1, \ldots, n$.

Let us prove that $\{\phi_0, \phi_1, \ldots, \phi_n\}$ is indeed a basis of $\mathbb{S}^1_\Delta$.

1) It is linearly independent, for if

$$\sum_{j=0}^{n} c_j \phi_j(x) = 0, \quad \forall x \in [a, b], \tag{4.71}$$

taking $x = x_j$ and using $\phi_j(x_i) = \delta_{ij}$, it follows that $c_j = 0$ for $j = 0, 1, \ldots, n$.

2) It spans $\mathbb{S}^1_\Delta$, since any $s \in \mathbb{S}^1_\Delta$ can be represented as

$$s(x) = \sum_{j=0}^{n} s(x_j)\phi_j(x). \tag{4.72}$$

The equality follows because the right hand side has the same values as s at x_i for $i = 0, 1, \ldots, n$ and since they are both in $\mathbb{S}^1_\Delta$ they must be equal.

As we know, we can represent the least squares approximation $s^* \in \mathbb{S}^1_\Delta$ to f as $s^* = c_0^*\phi_0 + \ldots + c_n^*\phi_n$, where the c_k^*, $k = 0, \ldots, n$, are the unique solution of the normal equations

$$\sum_{k=0}^{n} \langle \phi_k, \phi_j \rangle c_k = \langle f, \phi_j \rangle, \quad j = 0, 1, \ldots, n.$$

Now, $\langle \phi_k, \phi_j \rangle = 0$ if ϕ_k and ϕ_j do not overlap, i.e. $|k - j| > 1$ and by direct integration we get

$$\langle \phi_j, \phi_j \rangle = \int_a^b \phi_j^2(x)dx = \int_{x_{j-1}}^{x_{j+1}} \phi_j^2(x)dx = \frac{1}{3}(h_{j-1} + h_j), \tag{4.73}$$

$$\langle \phi_{j-1}, \phi_j \rangle = \int_a^b \phi_{j-1}(x)\phi_j(x)dx = \int_{x_{j-1}}^{x_j} \phi_{j-1}(x)\phi_j(x)dx = \frac{1}{6}h_{j-1}, \tag{4.74}$$

$$\langle \phi_{j+1}, \phi_j \rangle = \int_a^b \phi_{j+1}(x)\phi_j(x)dx = \int_{x_j}^{x_{j+1}} \phi_{j+1}(x)\phi_j(x)dx = \frac{1}{6}h_j, \tag{4.75}$$

where $h_j = x_{j+1} - x_j$. Hence, we obtain the tridiagonal linear system (note $h_{-1} = h_n = 0$)

$$\frac{1}{6}h_j c_{j-1} + \frac{1}{3}(h_{j-1} + h_j)c_j + \frac{1}{6}h_{j-1}c_{j+1} = \langle f, \phi_j \rangle, \quad j = 0, 1, \ldots, n. \tag{4.76}$$

This system is diagonally dominant and the solution can be found efficiently with Algorithm 9.5. There is one caveat though, in general the right hand side, $\langle f, \phi_j \rangle, \quad j = 0, 1, \ldots, n$ needs to be approximated numerically.

We close this section with one observation. The second derivative of the (complete or natural) cubic spline interpolant $s_I \in \mathbb{S}^3_\Delta$ of f is the L^2-best approximation to f'' in $\mathbb{S}^1_\Delta$. That is,

$$\|f'' - s_I''\| \le \|f'' - s\|, \quad \forall s \in \mathbb{S}^1_\Delta. \tag{4.77}$$

This follows immediately from Lemma 3.10.1 by taking $g = s_I''$.

4.2 Discrete Least Squares Approximation

Suppose that we are given a data set $(x_0, f_0), (x_1, f_1), \cdots, (x_N, f_N)$ and we would like to find the best 2-norm approximation f^* of these data in

$$W = \text{span}\{\phi_0, \phi_1, \ldots, \phi_n\},$$

where $\{\phi_0, \phi_1, \ldots, \phi_n\}$ is a set of linearly independent functions defined on an interval containing $x_0, \ldots, x_N$ and $N >> n$. The problem is same as the least squares problem for function approximation, except that now we measure the error using the 2-norm:

$$\|f - g\|^2 = \sum_{j=0}^{N} |f_j - g_j|^2, \quad g \in W, \tag{4.78}$$

where $g_j = g(x_j)$. The inner product is now the usual dot product

$$\langle f, g\rangle = \sum_{j=0}^{N} f_j \bar{g}_j. \tag{4.79}$$

In more generality we could use a weighted 2-norm,

$$\|f\|^2 = \sum_{j=0}^{N} |f_j|^2 w_j, \tag{4.80}$$

where $w_j > 0$, $j = 0 \ldots, N$ are given weights, but here we only consider the case $w_j = 1$ for all j.

The solution of the discrete least square problem is again characterized by the orthogonality of the error and we can write the least squares approximation $f^* \in W$ explicitly when the set of functions $\{\phi_0, \phi_1, \ldots, \phi_n\}$ is orthogonal with respect to the inner product (4.79).

$W = \mathbb{P}_n$ is often used for data fitting, particularly for small n. It is worth noting that when $N = n$ the solution to the discrete least squares problem in $\mathbb{P}_n$ is the interpolating polynomial p_n of the data, for

$$\|f - p_n\|^2 = \sum_{j=0}^{n} |f_j - p_n(x_j)|^2 = 0. \tag{4.81}$$

The case $W = \mathbb{P}_1$ is also known as linear regression. Taking $\phi_0(x) \equiv 1$, $\phi_1(x) = x$ the normal equations

$$\sum_{k=0}^{1} \langle \phi_k, \phi_j\rangle c_k = \langle f, \phi_j\rangle, \quad j = 0, 1,$$

become

$$\left(\sum_{j=0}^{N} 1\right) c_0 + \left(\sum_{j=0}^{N} x_j\right) c_1 = \sum_{j=0}^{N} f_j, \tag{4.82}$$

$$\left(\sum_{j=0}^{N} x_j\right) c_0 + \left(\sum_{j=0}^{N} x_j^2\right) c_1 = \sum_{j=0}^{N} x_j f_j. \tag{4.83}$$

This 2×2 linear system can be easily solved to obtain c_0 and c_1 and the least square approximation is $f^*(x) = c_0 + c_1 x$. For larger n, it is more appropriate to employ an orthogonal basis for $\mathbb{P}_n$. This can be obtained using the three-term recursion formula (4.38), which in this discrete setting is easy to implement because the coefficients α_k and β_k are just simple sums instead of integrals.

Example 4.5. *Suppose we are given the data set*

$$\{(0, 1.1), (1, 3.2), (2, 5.1), (3, 6.9)\}$$

and we would like to fit it to a line (in the least squares sense). Performing the sums, the normal equations of (4.82)-(4.83) become

$$4c_0 + 6c_1 = 16.3, \tag{4.84}$$

$$6c_0 + 14c_1 = 34.1. \tag{4.85}$$

Solving this 2×2 *linear system we get* $c_0 = 1.18$ *and* $c_1 = 1.93$. *Thus, the least squares approximation is*

$$p_1^*(x) = 1.18 + 1.93x$$

and the square of the error is

$$\sum_{j=0}^{3} [f_j - (1.18 + 1.93x_j)]^2 = 0.023.$$

Figure 4.3 shows the data and its least squares fit, $p_1^*(x) = 1.18 + 1.93x$.

Example 4.6. *Fitting to an exponential* $y = ae^{bx}$. *In this case the approximating function is not a linear combination of given (linearly independent) functions. Thus, the problem of finding the parameters* a *and* b *that minimize*

$$\sum_{j=0}^{N} [f_j - ae^{bx_j}]^2$$

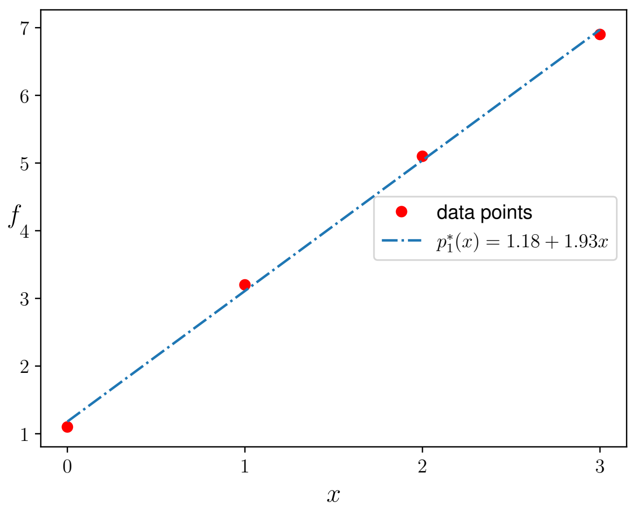

Figure 4.3: The data set $\{(0, 1.1), (1, 3.2), (2, 5.1), (3, 6.9)\}$ and its least squares fitting by a linear polynomial.

is nonlinear. However, we can turn it into a linear problem by taking the natural logarithm of $y = ae^{bx}$*, i.e.* $\ln y = \ln a + bx$*. Thus, tabulating* $(x_j, \ln f_j)$ *we obtain the normal equations*

$$\left(\sum_{j=0}^{N} 1\right) \ln a + \left(\sum_{j=0}^{N} x_j\right) b = \sum_{j=0}^{N} \ln f_j, \tag{4.86}$$

$$\left(\sum_{j=0}^{N} x_j\right) \ln a + \left(\sum_{j=0}^{N} x_j^2\right) b = \sum_{j=0}^{N} x_j \ln f_j, \tag{4.87}$$

and solve this linear system for $\ln a$ *and* b*. Then,* $a = e^{\ln a}$ *and* $b = b$*.*

If b *is given and we only need to determine* a *then the problem is linear as we are looking a function of the form* $a\phi_0$*, where* $\phi_0(x) = e^{bx}$*. We only have one normal equation to solve*

$$\left(\sum_{j=0}^{N} e^{2bx_j}\right) a = \sum_{j=0}^{N} f_j e^{bx_j}, \tag{4.88}$$

from which we obtain

$$a = \frac{\sum_{j=0}^{N} f_j e^{bx_j}}{\sum_{j=0}^{N} e^{2bx_j}}. \tag{4.89}$$

Example 4.7. *Discrete orthogonal polynomials. Let us construct the first few orthogonal polynomials with respect to the discrete inner product and* $x_j = (j+1)/10, j = 0, 1, \ldots, 9$. *We have* $\psi_0(x) = 1$ *and* $\psi_1(x) = x - \alpha_0$, *where*

$$\alpha_0 = \frac{\langle x\psi_0, \psi_0\rangle}{\langle \psi_0, \psi_0\rangle} = \frac{\sum_{j=0}^{9} x_j}{\sum_{j=0}^{9} 1} = 0.55.$$

and hence $\psi_1(x) = x - 0.55$. *Now,*

$$\psi_2(x) = (x - \alpha_1)\psi_1(x) - \beta_1\psi_0(x), \tag{4.90}$$

$$\alpha_1 = \frac{\langle x\psi_1, \psi_1\rangle}{\langle \psi_1, \psi_1\rangle} = \frac{\sum_{j=0}^{9} x_j(x_j - 0.55)^2}{\sum_{j=0}^{9}(x_j - 0.55)^2} = 0.55, \tag{4.91}$$

$$\beta_1 = \frac{\langle \psi_1, \psi_1\rangle}{\langle \psi_0, \psi_0\rangle} = 0.0825. \tag{4.92}$$

Therefore, $\psi_2(x) = (x - 0.55)^2 - 0.0825$. *We can now use these orthogonal polynomials to find the least squares approximation* p_2^* *by polynomial of degree at most two of a set of data* $(x_0, f_0), (x_1, f_1), \cdots, (x_9, f_9)$. *Let us take* $f_j = x_j^2 + 2x_j + 3$, *for* $j = 0, 1, \ldots, 9$. *Clearly, the least squares approximation should be* $p_2^*(x) = x^2 + 2x + 3$. *Let us confirm this by using the orthogonal* *polynomials ψ_0, ψ_1 and ψ_2. The coefficients are given by*

$$c_0 = \frac{\langle f, \psi_0 \rangle}{\langle \psi_0, \psi_0 \rangle} = 4.485, \tag{4.93}$$

$$c_1 = \frac{\langle f, \psi_1 \rangle}{\langle \psi_1, \psi_1 \rangle} = 3.1, \tag{4.94}$$

$$c_2 = \frac{\langle f, \psi_2 \rangle}{\langle \psi_2, \psi_2 \rangle} = 1, \tag{4.95}$$

which gives, $p_2^*(x) = (x-0.55)^2 - 0.0825 + (3.1)(x-0.55) + 4.485 = x^2 + 2x + 3$.

4.3 High-dimensional Data Fitting

In many applications each data point contains many variables. For example, a value for each pixel in an image, or clinical measurements of a patient, etc. We can put all these variables in a vector $x \in \mathbb{R}^d$ for $d >> 1$. Associated with x there is a scalar quantity f that can be measured or computed so that our data set consists of the points (x_j, f_j), with $x_j \in \mathbb{R}^d$ and $f_j \in \mathbb{R}$, for $j = 1, \ldots, N$.

A central problem in machine learning is that of predicting f from a given large, high-dimensional dataset; this is called *supervised learning*. The simplest approach is to postulate a linear relation

$$f(x) = a_0 + a^T x \tag{4.96}$$

and determine the *bias coefficient* $a_0 \in \mathbb{R}$ and the vector $a \in \mathbb{R}^d$ as a least squares solution, i.e. such that they minimize

$$\sum_{j=1}^{N} [f_j - (a_0 + a^T x_j)]^2.$$

We have already talked about the case $d = 1$. Here we are interested in $d >> 1$.

If we append an extra component, equal to 1, to each data vector x_j so that now $x_j = [1, x_{j1}, \ldots, x_{jd}]^T$, for $j = 1, \ldots, N$, we can write (4.96) as

$$f(x) = a^T x \tag{4.97}$$

and the dimension d is increased by one, $d \leftarrow d+1$, and relabeled again d. We are seeking a vector $a \in \mathbb{R}^d$ that minimizes

$$J(a) = \sum_{j=1}^{N} [f_j - a^T x_j]^2. \tag{4.98}$$

Putting the data x_j, $j = 1, \ldots, N$ as rows of an $N \times d$ matrix X and the f_j, $j = 1, \ldots, N$ as the components of a (column) vector f, i.e.

$$X = \begin{bmatrix} \quad & x_1 & \quad \\ & \vdots & \\ & x_N & \end{bmatrix} \quad \text{and} \quad f = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix} \tag{4.99}$$

we can write (4.98) as

$$J(a) = \langle f - Xa, f - Xa \rangle = \|f - Xa\|^2, \tag{4.100}$$

where $\langle \cdot, \cdot \rangle$ is the standard inner product in $\mathbb{R}^N$. Thus, we are looking for the least squares approximation f^* to f by functions in

$$W = \text{span}\{\text{columns of } X\}.$$

We can find this from its geometric characterization:

$$\langle f - f^*, w \rangle = 0, \quad \forall w \in W. \tag{4.101}$$

Since $f - f^*$ is orthogonal to W if it is orthogonal to each column of X, i.e. $X^T(f - f^*) = 0$, writing $f^* = Xa^*$ it follows that a^* should be a solution of the linear system

$$X^T X a = X^T f. \tag{4.102}$$

These are the normal equations of this least squares problem. If the columns of X are linearly independent, i.e. if for every $a \neq 0$ we have that $Xa \neq 0$, then the $d \times d$ matrix $X^T X$ is positive definite and hence nonsingular. Thus, in this case, there is a unique solution to $\min_{a \in \mathbb{R}^d} \|f - Xa\|^2$ given by

$$a^* = (X^T X)^{-1} X^T f. \tag{4.103}$$

The $d \times N$ matrix

$$X^\dagger = (X^T X)^{-1} X^T \tag{4.104}$$

is called the *pseudoinverse* of the $N \times d$ matrix X. Note that if X were square and nonsingular $X^\dagger$ would coincide with the inverse X^{-1}.

Orthogonality is again central for the computation of a least squares approximation. Rather than working with the normal equations, whose matrix $X^T X$ may be very sensitive to perturbations in the data such as noise, we employ an orthogonal basis for the approximating subspace W to find a solution. While in principle this can be done by applying the Gram-Schmidt process (cf. Section 4.1.2.1) to the columns of X, this is a numerically unstable procedure. A more efficient method using a sequence of orthogonal transformations, known as *Householder reflections* (see Section 11.2), is usually preferred. Once this orthonormalization process is completed we get $X = QR$ [see (8.30)], where

$$Q = \begin{bmatrix} & * & \cdots & * \\ & \vdots & \cdots & \vdots \\ \tilde{Q} & & & \\ & \vdots & & \vdots \\ & * & \cdots & * \end{bmatrix}, \quad R = \begin{bmatrix} & \tilde{R} & \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix}. \tag{4.105}$$

Here Q is an $N \times N$ orthogonal matrix (i.e. $Q^T Q = QQ^T = I$). The $N \times d$ block $\tilde{Q}$ consists of columns that form an orthonormal basis for the column space of X and $\tilde{R}$ is a $d \times d$ upper triangular matrix.

Using this QR factorization of the matrix X we have

$$\|f - Xa\|^2 = \|f - QRa\|^2 = \|Q^T(f - QRa)\|^2 = \|Q^T f - Ra\|^2. \tag{4.106}$$

Therefore, a solution to $\min_{a \in \mathbb{R}^d} \|f - Xa\|^2$ is obtained by solving the system $Ra = Q^T f$. Because of the zero block in R, the problem reduces to solving the $d \times d$ upper triangular system

$$\tilde{R}a = \tilde{Q}^T f. \tag{4.107}$$

If the matrix X is full rank there is a unique solution to (4.107). Note however that the last $N - d$ equations in $Ra = Q^T f$ may be satisfied or not (that depends on f) but we have no control on them.

4.4 Bibliographic Notes

Section 4.1 . The main objective of this section is to emphasize that this particular case of best approximation has a useful geometric interpretation and that orthogonality plays a central role both in the theory and in actual computations. We separate the continuous and discrete cases. Gautschi [Gau11] presents a unified approach. Good references for orthogonal polynomials are the classical book by Szegö [Sze39] and the more modern monograph by Gautschi [Gau04]. The convergence of the least squares approximation is treated more extensively in [Che82]. The estimate of the Chebyshev coefficients for analytic functions is from Rivlin's monograph [Riv20]. Finally, the least squares approximation by splines is a popular technique in data analytics (see, for example, *smoothing splines* in [HTF09]).

Section 4.2 . Our presentation was influenced by the data fitting section in Conte and de Boor's classical book [CdB72], which also has a fortran code for the generation of discrete orthogonal polynomials.

Section 4.3 . This section was drawn from Section 4.8 of [SB02]. The discussion of the QR factorization has been postponed to the linear algebra part of this text as this matrix factorization also plays an important role in numerical methods for eigenvalue problems.

Chapter 5

Computer Arithmetic

Up to now, we have tacitly assumed that all the needed numerical computations were to be done with exact arithmetic. In reality, a computer approximates numbers using only a finite number of digits. Thus, all numerical computations executed in a computer inevitably involve this additional, number approximation. In this chapter, we will discuss briefly the basics of computer number representation and computer arithmetic, focusing on one of their most important aspects, which is the potential cancellation of digits of accuracy.

5.1 Floating Point Numbers

Floating point numbers are based on scientific notation in binary (base 2). For example,

$$\begin{aligned}(1.0101)_2 \times 2^2 &= (1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4}) \times 2^2 \\ &= (1 + \frac{1}{4} + \frac{1}{16}) \times 4 = 5.25_{10}.\end{aligned}$$

We can write any non-zero real number x in normalized, binary, scientific notation as

$$x = \pm S \times 2^E, \quad 1 \leq S < 2, \tag{5.1}$$

where S is called the *significant* or *mantissa* and E is the *exponent*. In general, S is an infinite expansion of the form

$$S = (1.b_1 b_2 \cdots)_2. \tag{5.2}$$

In a computer, a real number is represented in scientific notation but using a finite number of binary digits (bits). We call these numbers **floating point numbers**. In single precision (SP), floating point numbers are stored in 32-bit words whereas in double precision (DP), used in most scientific computing applications, a 64-bit word is employed: 1 bit is used for the sign, 52 bits for S, and 11 bits for E. This memory limits produce a large but *finite set of floating point numbers* that can be represented in a computer. Moreover, the floating points numbers are not uniformly distributed!

The maximum exponent possible in DP would be $2^{11} - 1 = 2047$ but this is shifted to allow an approximately even representation of small and large numbers. So we actually have $E_{\min} = -1022$, $E_{\max} = 1023$. Consequently, the smallest and the largest DP floating point numbers which can be represented in DP are

$$N_{\min} = \min_{x \in DP} |x| = 2^{-1022} \approx 2.2 \times 10^{-308}, \tag{5.3}$$

$$N_{\max} = \max_{x \in DP} |x| = (1.1.....1)_2 \cdot 2^{1023} = (2 - 2^{-52}) \cdot 2^{1023} \approx 1.8 \times 10^{308}. \tag{5.4}$$

If in the course of a computation a number is produced which is bigger than $N_{\max}$ we get an *overflow* error and the computation would halt. If the number is less than $N_{\min}$ (in absolute value) then an *underflow* error occurs.

5.2 Rounding and Machine Precision

To represent a real number x as a floating point number, rounding has to be performed to retain only the numbers of binary bits allowed in the significant.

Let $x \in \mathbb{R}$ and its binary expansion be $x = \pm(1.b_1 b_2 \cdots)_2 \times 2^E$. One way to approximate x to a floating number with d bits in the significant is to truncate or *chop* discarding all the bits after b_d, i.e.

$$x^* = \text{chop}(x) = \pm(1.b_1 b_2 \cdots b_d)_2 \times 2^E. \tag{5.5}$$

Recall that in DP $d = 52$.

A better way to approximate a real number with a floating point number is to do rounding up or down (to the nearest floating point number), just as we do when we round in base 10. In binary, rounding is simpler because b_{d+1} can only be 0 (we round down) or 1 (we round up). We can write this type of rounding in terms of the chopping described above as

$$x^* = \text{round}(x) = \text{chop}(x + 2^{-(d+1)} \times 2^E). \tag{5.6}$$

Definition 5.1. *Given an approximation* x^* *to* x, *the* **absolute error** *is defined by* $|x - x^*|$ *and the* **relative error** *by* $|\frac{x-x^*}{x}|$, *for* $x \neq 0$.

The relative error is generally more meaningful than the absolute error to measure a given approximation. The relative error in chopping and in rounding (called a **round-off error**) are

$$\left|\frac{x - \text{chop}(x)}{x}\right| \leq \frac{2^{-d}2^E}{(1.b_1b_2\cdots)2^E} \leq 2^{-d}, \tag{5.7}$$

$$\left|\frac{x - \text{round}(x)}{x}\right| \leq \frac{1}{2}2^{-d}. \tag{5.8}$$

The number 2^{-d} is called **machine precision** or epsilon (eps). In DP, eps $= 2^{-52} \approx 2.22 \times 10^{-16}$. The smallest DP number greater than 1 is 1+eps. As we will see below, it is more convenient to write (5.8) as

$$\text{round}(x) = x(1+\delta), \quad |\delta| \leq \text{eps}. \tag{5.9}$$

5.3 Correctly Rounded Arithmetic

Computers today follow the IEEE standard for floating point representation and arithmetic. This standard requires a consistent floating point representation of numbers across computers and *correctly rounded arithmetic*.

In correctly rounded arithmetic, *the computer operations of addition, subtraction, multiplication, and division are the correctly rounded value of the exact result.* For example, if x and y are floating point numbers and $\oplus$ is the machine addition, then

$$x \oplus y = \text{round}(x+y) = (x+y)(1+\delta_+), \quad |\delta_+| \leq \text{eps}, \tag{5.10}$$

and similarly for computer subtraction, multiplication, and division.

One important interpretation of (5.10) is as follows. Assuming $x + y \neq 0$, write

$$\delta_+ = \frac{1}{x+y}[\delta_x + \delta_y].$$

Then,

$$x \oplus y = (x+y)\left[1 + \frac{1}{x+y}(\delta_x + \delta_y)\right] = (x+\delta_x) + (y+\delta_y). \tag{5.11}$$

The computer addition $\oplus$ is giving the *exact result but for a sightly perturbed data*. This interpretation is the basis for **backward error analysis**, which is used to study how round-off errors propagate in a numerical algorithm.

5.4 Propagation of Errors and Cancellation of Digits

Let round(x) and round(y) denote the floating point approximation of x and y, respectively, and assume that their product is computed exactly, i.e

$$\text{round}(x)\cdot\text{round}(y) = x(1+\delta_x)\cdot y(1+\delta_y) = x\cdot y(1+\delta_x+\delta_y+\delta_x\delta_y) \approx x\cdot y(1+\delta_x+\delta_y),$$

where $|\delta_x|$,$|\delta_y| \leq \text{eps}$. Therefore, for the relative error we get

$$\left|\frac{x\cdot y - \text{round}(x)\cdot\text{round}(y)}{x\cdot y}\right| \approx |\delta_x + \delta_y|, \tag{5.12}$$

which is acceptable.

Let us now consider addition (or subtraction):

$$\begin{aligned}\text{round}(x) + \text{round}(y) &= x(1+\delta_x) + y(1+\delta_y) = x + y + x\delta_x + y\delta_y \\ &= (x+y)\left(1 + \frac{x}{x+y}\delta_x + \frac{y}{x+y}\delta_y\right),\end{aligned}$$

where we have assume $x + y \neq 0$. The relative error is

$$\left|\frac{x+y-(\text{round}(x)+\text{round}(y))}{x+y}\right| = \left|\frac{x}{x+y}\delta_x + \frac{y}{x+y}\delta_y\right|. \tag{5.13}$$

If x and y have the same sign then $\frac{x}{x+y}, \frac{y}{x+y}$ are both positive and bounded by 1. Therefore the relative error is less than $|\delta_x + \delta_y|$, which is fine. But if x and y have different sign and are close in magnitude, then the error could be largely amplified because $|\frac{x}{x+y}|, |\frac{y}{x+y}|$ can be very large.

Example 5.1. *Suppose we have 10 bits of precision and*

$$\begin{aligned}x &= (1.01011100**)_2 \times 2^E, \\ y &= (1.01011000**)_2 \times 2^E,\end{aligned}$$

where the $*$ stands for inaccurate bits that, for example, were generated in previous floating point computations. Then, in this 10 bit precision arithmetic

$$z = x - y = (1.00 * * * * * * * *)_2 \times 2^{E-6}. \tag{5.14}$$

We end up with only 2 bits of accuracy in z. Any further computations using z will result in an accuracy of 2 bits or lower!

Example 5.2. *Sometimes we can rewrite the difference of two very close numbers to avoid digit cancellation. For example, suppose we would like to compute*

$$y = \sqrt{1+x} - 1$$

for $x > 0$ and very small. Clearly, we will have loss of digits if we proceed directly. However, if we rewrite y as

$$y = (\sqrt{1+x} + 1)\frac{\sqrt{1+x} - 1}{\sqrt{1+x} + 1} = \frac{x}{\sqrt{1+x} + 1}$$

then the computation can be performed at nearly machine precision level.

5.5 Bibliographic Notes

Sections 5.1-5.4. This is usually the first chapter in most numerical analysis textbooks. We have decided to present it only now and to keep it to a minimum because often students find the topics of floating point arithmetic and round-off errors tedious. But more importantly, many students get the discouraging misconception that numerical analysis is a subject just about round-off errors when they are first introduced to it through this particular topic. The analysis of round-off errors is an important but small part of the broad field of numerical analysis. The book of Wilkinson [Wil94] provides many examples of backward (and forward) error analysis in the context of computations involving polynomials and for numerical linear algebra.

The main sources for this short chapter on computer arithmetic were the excellent monograph by Overton [Ove01] and Chapter 1 of Gautschi's book [Gau11].

Chapter 6

Numerical Differentiation

In this chapter we look at the approximation of the derivative(s) of a function f at a point, given a few values of f at neighboring points, via interpolation. The resulting formulas are called finite difference formulas. We also look briefly at spectral derivative approximations using the fast Fourier transform.

6.1 Finite Differences

Suppose f is a differentiable function and we'd like to approximate $f'(x_0)$ given the value of f at x_0 and at neighboring points $x_1, x_2, \ldots, x_n$. We could approximate f by its interpolating polynomial p_n at those points and use

$$f'(x_0) \approx p_n'(x_0). \tag{6.1}$$

There are several other possibilities. For example, we can approximate $f'(x_0)$ by the derivative of the cubic spline interpolant of f evaluated at x_0, or by the derivative of the least squares Chebyshev expansion of f, etc.

We are going to focus here on simple, finite difference formulas obtained by differentiating low order interpolating polynomials.

Assuming $x, x_0, \ldots, x_n \in [a, b]$ and $f \in C^{n+1}[a, b]$, we have

$$f(x) = p_n(x) + \frac{1}{(n+1)!} f^{(n+1)}(\xi(x))\, \omega_n(x), \tag{6.2}$$

for some $\xi(x) \in (a, b)$ and

$$\omega_n(x) = (x - x_0)(x - x_1) \cdots (x - x_n). \tag{6.3}$$

Thus,

$$f'(x_0) = p_n'(x_0) + \frac{1}{(n+1)!}\left[\frac{d}{dx}f^{(n+1)}(\xi(x))\omega_n(x) + f^{(n+1)}(\xi(x))\,\omega_n'(x)\right]\Bigg|_{x=x_0}.$$

But $\omega_n(x_0) = 0$ and $\omega_n'(x_0) = (x_0 - x_1)\cdots(x_0 - x_n)$, thus

$$f'(x_0) = p_n'(x_0) + \frac{1}{(n+1)!}f^{(n+1)}(\xi_0)(x_0 - x_1)\cdots(x_0 - x_n), \tag{6.4}$$

where ξ_0 is between $\min\{x_0, x_1, \ldots, x_n\}$ and $\max\{x_0, x_1, \ldots, x_n\}$.

Example 6.1. *Take $n = 1$ and $x_1 = x_0 + h$ ($h > 0$). In Newton's form*

$$p_1(x) = f(x_0) + \frac{f(x_0+h) - f(x_0)}{h}(x - x_0), \tag{6.5}$$

and $p_1'(x_0) = \frac{1}{h}[f(x_0+h) - f(x_0)]$. We obtain the forward difference formula for approximating $f'(x_0)$

$$D_h^+ f(x_0) := \frac{f(x_0+h) - f(x_0)}{h}. \tag{6.6}$$

From (6.4) the error in this approximation is

$$f'(x_0) - D_h^+ f(x_0) = \frac{1}{2!}f''(\xi_0)(x_0 - x_1) = -\frac{1}{2}f''(\xi_0)h. \tag{6.7}$$

Example 6.2. *Take again $n = 1$ but now $x_1 = x_0 - h$. Then $p_1'(x_0) = \frac{1}{h}[f(x_0) - f(x_0 - h)]$ and we get the backward difference formula for approximating $f'(x_0)$*

$$D_h^- f(x_0) := \frac{f(x_0) - f(x_0-h)}{h}. \tag{6.8}$$

Its error is

$$f'(x_0) - D_h^- f(x_0) = \frac{1}{2}f''(\xi_0)h. \tag{6.9}$$

Example 6.3. *Let n=2 and $x_1 = x_0 - h$, $x_2 = x_0 + h$. Then, p_2 in Newton's form is*

$$p_2(x) = f[x_1] + f[x_1, x_0](x - x_1) + f[x_1, x_0, x_2](x - x_1)(x - x_0).$$

Let us obtain the divided difference table:

$$
\begin{array}{cccc}
x_0 - h & f(x_0 - h) & & \\
 & & \frac{f(x_0)-f(x_0-h)}{h} & \\
x_0 & f(x_0) & & \frac{f(x_0+h)-2f(x_0)+f(x_0-h)}{2h^2} \\
 & & \frac{f(x_0+h)-f(x_0)}{h} & \\
x_0 + h & f(x_0 + h) & &
\end{array}
$$

Therefore,

$$p_2'(x_0) = \frac{f(x_0) - f(x_0 - h)}{h} + \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{2h^2} h$$

and thus

$$p_2'(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h}. \tag{6.10}$$

This defines the centered difference formula to approximate $f'(x_0)$

$$D_h^0 f(x_0) := \frac{f(x_0 + h) - f(x_0 - h)}{2h}. \tag{6.11}$$

Its error is

$$f'(x_0) - D_h^0 f(x_0) = \frac{1}{3!} f'''(\xi_0)(x_0 - x_1)(x_0 - x_2) = -\frac{1}{6} f'''(\xi_0) h^2. \tag{6.12}$$

Example 6.4. *Let $n = 2$ and $x_1 = x_0 + h$, $x_2 = x_0 + 2h$. The table of divided differences is*

$$
\begin{array}{cccc}
x_0 & f(x_0) & & \\
 & & \frac{f(x_0+h)-f(x_0)}{h} & \\
x_0 + h & f(x_0 + h) & & \frac{f(x_0+2h)-2f(x_0+h)+f(x_0)}{2h^2} \\
 & & \frac{f(x_0+2h)-f(x_0+h)}{h} & \\
x_0 + 2h & f(x_0 + 2h). & &
\end{array}
$$

Therefore,

$$p_2'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} + \frac{f(x_0 + 2h) - 2f(x_0 + h) + f(x_0)}{2h^2}(-h)$$

and simplifying

$$p_2'(x_0) = \frac{-f(x_0+2h) + 4f(x_0+h) - 3f(x_0)}{2h}. \tag{6.13}$$

If we use this sided difference to approximate $f'(x_0)$, the error is

$$f'(x_0) - p_2'(x_0) = \frac{1}{3!} f'''(\xi_0)(x_0 - x_1)(x_0 - x_2) = \frac{1}{3} h^2 f'''(\xi_0), \tag{6.14}$$

which is twice as large as that of the centered finite difference formula.

Example 6.5. *Tables 6.1 and 6.2 show the approximations of the derivative for $f(x) = e^{-x}$ at $x_0 = 0$, obtained with the forward and the centered finite differences, respectively. The rate of convergence is evidenced in the last column, the decrease factor. The error decreases by approximately a factor of $1/2$ when h is halved for the forward difference (linear rate of convergence) and by approximately a factor of $1/4$ for the centered difference (second order of convergence).*

Table 6.1: Approximation of $f'(0)$ for $f(x) = e^{-x}$ using the forward finite difference. The decrease factor is $\text{error}(\frac{h}{2})/\text{error}(h)$.

h	$D_h^+ f(0)$	$\lvert D_h^+ f(0) - f'(0)\rvert$	Decrease factor
0.20	-0.90634623	0.09365377	
0.10	-0.95162582	0.04837418	0.51652147
0.05	-0.97541151	0.02458849	0.50829781
0.025	-0.98760352	0.01239648	0.50415789

Table 6.2: Approximation of $f'(0)$ for $f(x) = e^{-x}$ using the centered finite difference. The decrease factor is $\text{error}(\frac{h}{2})/\text{error}(h)$.

h	$D_h^0 f(0)$	$\lvert D_h^0 f(0) - f'(0)\rvert$	Decrease factor
0.20	-1.00668001	0.00668001	
0.10	-1.0016675	0.00166750	0.24962530
0.05	-1.00041672	0.00041672	0.24990627
0.025	-1.00010417	0.00010417	0.24997656

6.2 The Effect of Round-Off Errors

In numerical differentiation we take differences of values, which for small h, could be very close to each other. As we know, this leads to loss of accuracy because of finite precision, floating point arithmetic. For example, consider the centered difference formula (6.11). For simplicity, let us suppose that h has an exact floating point representation and that we make no rounding error when doing the division by h. That is, suppose that the only source of round-off error is in the computation of the difference

$$f(x_0 + h) - f(x_0 - h).$$

Then, $f(x_0+h)$ and $f(x_0-h)$ are replaced by $f(x_0+h)(1+\delta_+)$ and $f(x_0 - h)(1+\delta_-)$, respectively with $|\delta_+| \leq \text{eps}$ and $|\delta_-| \leq \text{eps}$ (recall eps is the machine precision) and we have

$$\frac{f(x_0+h)(1+\delta_+) - f(x_0-h)(1+\delta_-)}{2h} = \frac{f(x_0+h) - f(x_0-h)}{2h} + r_h,$$

where

$$r_h = \frac{f(x_0+h)\delta_+ - f(x_0-h)\delta_-}{2h}.$$

Clearly,

$$|r_h| \leq (|f(x_0+h)| + |f(x_0-h)|)\,\frac{\text{eps}}{2h} \approx |f(x_0)|\frac{\text{eps}}{h}.$$

The *approximation error or truncation error* for the centered finite difference approximation is $-\frac{1}{6}f'''(\xi_0)h^2$. Thus, the total error can be approximately bounded by $\frac{1}{6}h^2\|f'''\|_\infty + |f(x_0)|\frac{\text{eps}}{h}$. Differentiating this error bound with respect to h and setting the derivative to zero, we find that it has a minimum at the value

$$h^* = \left(3\frac{|f(x_0)|}{\|f'''\|_\infty}\,\text{eps}\right)^{\frac{1}{3}}. \tag{6.15}$$

Consequently, the total error can at most be decreased to $O(\text{eps}^{\frac{2}{3}})$, i.e. we do not get machine precision. Figure 6.1 shows the behavior of the round-off and discretization errors as a function of h for the centered finite difference. When these two errors become comparable, around the point h^*, decreasing h

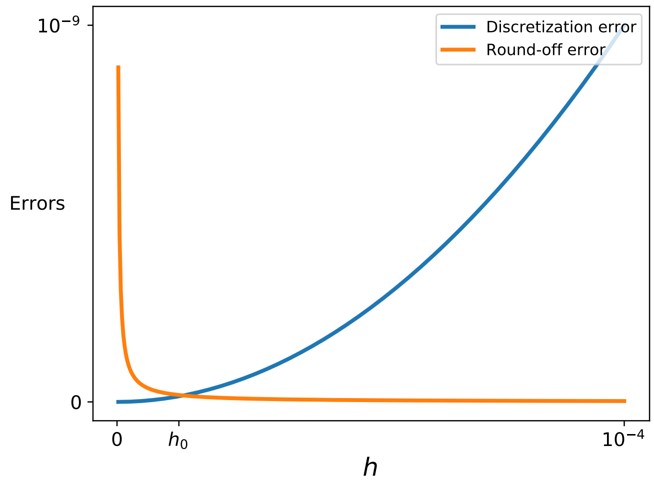

Figure 6.1: Behavior of the round-off and discretization errors for the centered finite difference. The smallest total error is achieved for a value h^* around the point where the two errors become comparable.

further does not decrease the total error as roundoff errors start to dominate.

The dominant effect of roundoff errors in finite differences when h is very small is exacerbated in finite differences for higher order derivatives. If f can be extended to an analytic function in a region of the complex plane, a more accurate approximation of the derivatives of f can be obtained by employing Cauchy's integral formula for the n-th derivative of f:

$$f^{(n)}(z_0) = \frac{n!}{2\pi i} \int_C \frac{f(z)}{(z - z_0)^{n+1}} dz, \tag{6.16}$$

where C is a simple closed contour around z_0 and f is analytic on and inside C. Parametrizing C as a circle of radius r we get

$$f^{(n)}(z_0) = \frac{n!}{2\pi r^n} \int_0^{2\pi} f(z_0 + re^{i\theta})e^{-in\theta} d\theta. \tag{6.17}$$

The integrand is periodic and smooth so it can be approximated with spectral accuracy with the composite trapezoidal rule.

Table 6.3: Approximation of $f'(0)$, $f''(0)$, and $f'''(0)$ for $f(x) = e^{-x}$ using the discrete Cauchy's integral formula (6.19) with $r = 1$ and $N = 4, 8, 16, 32$.

N	$f'(0)$ approx.	$f''(0)$ approx.	$f'''(0)$ approx.
4	-1.008336089225849	1.002778328947104	-1.0011906265077144
8	-1.0000027557319253	1.000000551146385	-1.000000150312653
16	-1.0000000000000027	1.0000000000000004	-1.0000000000000029
32	-1.0000000000000000	1.0000000000000002	-1.0000000000000009

Example 6.6. *We are going to use* (6.17) *to approximate the first and second derivatives ($n = 1, 2$) of $f(x) = e^{-x}$ at* 0*. First, because f is real-valued we have*

$$
\begin{aligned}
f^{(n)}(0) &= \frac{n!}{2\pi r^n} \int_0^{2\pi} Re\left\{ f(re^{i\theta}) e^{-in\theta} \right\} d\theta \\
&= \frac{n!}{2\pi r^n} \int_0^{2\pi} e^{-r\cos\theta} \cos(n\theta + r\sin\theta) d\theta.
\end{aligned}
\tag{6.18}
$$

We now approximate the integral with the composite trapezoidal rule using N equi-spaced points $\theta_j = 2\pi j/N$, $j = 0, 1, \ldots, N-1$:

$$
f^{(n)}(0) \approx \frac{n!}{Nr^n} \sum_{j=0}^{N-1} e^{-r\cos\theta_j} \cos(n\theta_j + r\sin\theta_j) \tag{6.19}
$$

Table 6.3 shows the fast convergence of the approximations to $f'(0)$, $f''(0)$, and $f'''(0)$ and demonstrates that it is possible to achieve machine precision ($O(10^{-16})$ in DP) accuracy with a modest N even for higher derivatives of f. However, it is important to keep in mind the underlying assumptions for the use of (6.16)*: f is an analytic function in a region containing a disk centered at z_0 and we have access to N equi-spaced values of f on boundary of that disk.*

6.3 Richardson's Extrapolation

Another approach to obtain finite difference formulas to approximate derivatives is through Taylor expansions. For example,

$$f(x_0+h) = f(x_0) + f'(x_0)h + \frac{1}{2}f''(x_0)h^2 + \frac{1}{3!}f^{(3)}(x_0)h^3 + \dots \tag{6.20}$$

$$f(x_0-h) = f(x_0) - f'(x_0)h + \frac{1}{2}f''(x_0)h^2 - \frac{1}{3!}f^{(3)}(x_0)h^3 + \dots \tag{6.21}$$

Then, subtracting (6.21) from (6.20), we have $f(x_0+h) - f(x_0-h) = 2f'(x_0)h + \frac{2}{3!}f'''(x_0)h^3 + \cdots$ and therefore

$$\frac{f(x_0+h) - f(x_0-h)}{2h} = f'(x_0) + c_2h^2 + c_4h^4 + \cdots, \tag{6.22}$$

where the c_n, for n even, are constants.

Similarly, if we add (6.20) and (6.21) we obtain

$$f(x_0+h) + f(x_0-h) = 2f(x_0) + f''(x_0)h^2 + \frac{1}{12}f^{(4)}(x_0)h^4 + \cdots \tag{6.23}$$

and consequently

$$f''(x_0) = \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{h^2} - \frac{1}{12}f^{(4)}(x_0)h^2 + \cdots \tag{6.24}$$

The finite difference

$$D_h^2 f(x_0) = \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{h^2} \tag{6.25}$$

is thus a second order approximation of $f''(x_0)$. Moreover

$$D_h^2 f(x_0) = f''(x_0) + \tilde{c}_2h^2 + \tilde{c}_4h^4 + \dots \tag{6.26}$$

for some constants $\tilde{c}_2, \tilde{c}_4, \dots$ and h sufficiently small.

From (6.22) we know that asymptotically

$$D_h^0 f(x_0) = f'(x_0) + c_2h^2 + c_4h^4 + \cdots \tag{6.27}$$

so we could apply Richardson extrapolation once to obtain a fourth order approximation of $f'(x_0)$. Evaluating (6.27) at $h/2$ we get

$$D_{h/2}^0 f(x_0) = f'(x_0) + \frac{1}{4}c_2h^2 + \frac{1}{16}c_4h^4 + \cdots \tag{6.28}$$

and multiplying this equation by 4, subtracting (6.27) to the result and dividing by 3 we obtain

$$D_h^{ext} f(x_0) := \frac{4D_{h/2}^0 f(x_0) - D_h^0 f(x_0)}{3} = f'(x_0) + Ch^4 + \cdots , \tag{6.29}$$

where C is a constant. The new method $D_h^{ext} f(x_0)$ has order of convergence 4 for about twice the amount of work needed for $D_h^0 f(x_0)$. Table 6.4 shows the approximation $D_h^{ext} f(0)$ of $f'(0)$, again for $f(x) = e^{-x}$, and its error for $h = 0.2/2^j$, $j = 0, 1, 2, 3$. The error decreases by a factor of approximately $0.0625 = 2^{-4}$ when halving h, confirming that the method is $O(h^4)$ accurate. Note that with $h = 0.025$, $D_h^{ext} f(0)$ has about 8 digits of accuracy whereas $D_h^0 f(0)$ has only 3 (Table 6.2).

Round-off errors are still $O(\text{eps}/h)$ and the minimum total error will occur when $O(h^4)$ is $O(\text{eps}/h)$, i.e. when $h = O(\text{eps}^{1/5})$. Thus, for $D_h^{ext} f(x_0)$ the minimum total error that can be achieved is $O(\text{eps}^{4/5})$.

Table 6.4: The Richardson extrapolation approximation $D_h^{ext} f(x_0)$ (6.29) of $f'(0)$ for $f(x) = e^{-x}$. The decrease factor is $\text{error}(\frac{h}{2})/\text{error}(h)$.

h	$D_h^{ext} f(0)$	$\lvert D_h^{ext} f(0) - f'(0)\rvert$	Decrease factor
0.20	-0.999996662696098	0.0000033373039020	
0.10	-0.9999997916046542	0.0000002083953458	0.062444222
0.05	-0.9999999869781995	0.0000000130218005	0.062486042
0.025	-0.9999999991861838	0.0000000008138162	0.062496444

6.4 Fast Spectral Differentiation

In many applications, such as finding approximate solutions of differential equations, we need to obtain accurate approximations of derivatives not just at one point but at all the interpolation points. This type of approximations could get computationally expensive if computed directly and for a large number of points. Fortunately, we can do this task both efficiently and accuracy using the fast Fourier transform in two important cases, when the approximation uses the trigonometric interpolant and when it uses the Chebychev interpolant.

6.4.1 Fourier Spectral Differentiation

Let us suppose that we have values f_j, $j = 0, 1, \ldots, N-1$ of a 2π-periodic function (or simply a periodic array of values) corresponding to the equivalent-spaced nodes $x_j = 2\pi j/N$, $j = 0, 1, \ldots, N-1$. Consider the trigonometric interpolant of these values

$$s_{N/2}(x) = \sum_{k=-N/2}^{N/2}{}'' c_k e^{ikx}. \tag{6.30}$$

Since

$$s'_{N/2}(x) = \sum_{k=-N/2}^{N/2}{}'' ikc_k e^{ikx}, \tag{6.31}$$

we can compute the coefficients ikc_k of the trigonometric polynomial $s'_{N/2}$ from the discrete Fourier coefficients c_k of f, which in turn can be evaluated efficiently with the FFT, and then transform back (take the inverse DFT) to get the array $s'_{N/2}(x_j)$, $j = 0, 1, \ldots, N-1$. However, there are two important details: the coefficients c_k for $k = -N/2, \ldots, -1$ of the interpolant correspond to c_k, $k = N/2, \ldots, N-1$ of the DFT (see) and $c_{N/2} = c_{-N/2}$. The latter implies that $c_{N/2}$ is real and that the highest wave number term, called the *Nyquist mode*, in $s_{N/2}$ is $c_{N/2} \cos\left(\frac{N}{2}x\right)$, whose derivative vanishes at the nodes $x_j = j2\pi/N$, $j = 0, 1, \ldots, N-1$. Consequently, we need to set to zero the $k = N/2$ coefficient of $s'_{N/2}$. Thus, to obtain an approximation of the derivative at the equi-spaced points we proceed as follows:

1. Compute the DFT of f (i.e. the coefficients c_k, $k = 0, 1, \ldots, N-1$) with the FFT.

2. Define the array of coefficients

$$i[0, c_1, \ldots, (N/2-1)c_{N/2-1}, 0, (-N/2+1)c_{N/2+1}, \ldots, -c_{N-1}] \tag{6.32}$$

3. Perform the inverse DFT (inverse FFT) of (6.32) to get the array corresponding to $s'_{N/2}(x_j)$, $j = 0, 1, \ldots, N-1$.

We call this approach Fourier spectral differentiation or Fourier spectral approximation of the derivative.

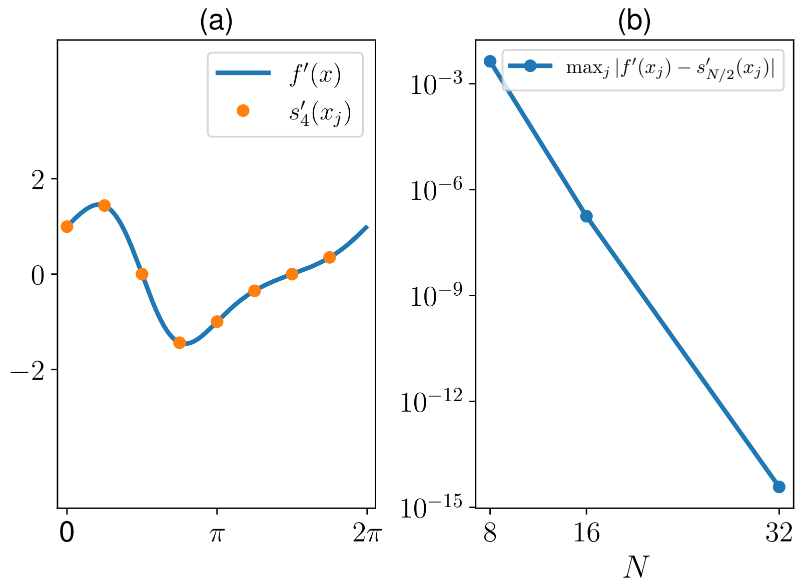

Figure 6.2: Fourier spectral approximation of the derivative of $f(x) = e^{\sin x}$ at $x_j = 2\pi j/N$, $j = 0, 1, \ldots, N-1$. (a) f' and its Fourier approximation $s'_4(x_j)$ and (b) the maximum error $\max\limits_j |f'(x_j) - s'_{N/2}(x_j)|$ for $N = 8, 16, 32$.

Example 6.7. *Figure 6.2 shows the Fourier spectral approximation of the derivative of* $f(x) = e^{\sin x}$ *at the points* $x_j = 2\pi j/N$, $j = 0, 1, \ldots, N-1$ *for* $N = 8$ *and the maximum error for* $N = 8, 16, 32$. *As it is evident from Fig. 6.2(b), the spectral approximation converges exponentially to* f' *(note the logarithmic scale on the vertical axis).*

Approximations to higher derivatives $f^{(p)}$ can be computed similarly by using $(ik)^p c_k$ as the discrete Fourier coefficients of $s^{(p)}_{N/2}$. Again, for odd derivatives (p odd) the Nyquist mode, $k = N/2$, needs to be set to zero.

6.4.2 Chebyshev Spectral Differentiation

Recall (see Section 3.13) that setting $x = \cos\theta$, the Chebyshev interpolant of f can be written as

$$p_n(\cos\theta) = {\sum_{k=0}^{n}}'' c_k \cos k\theta, \tag{6.33}$$

where

$$c_k = \frac{2}{n} {\sum_{j=0}^{n}}'' f\left(\cos(j\pi/n)\right)\cos(kj\pi/n), \qquad k = 0, 1, .., n. \tag{6.34}$$

We would like to approximate the derivative of f at the Chebyshev nodes $x_j = \cos(j\pi/n)$, $j = 0, 1, \ldots, n$ using p_n'. Denoting $\Pi_n(\theta) = p_n(\cos\theta)$ and applying the change rule, we have

$$\Pi_n'(\theta) = p_n'(x)\frac{dx}{d\theta} = -\sin\theta \; p_n'(x) = -\sqrt{1-x^2}\; p_n'(x). \tag{6.35}$$

Thus,

$$p_n'(x_j) = -\frac{\Pi_n'(j\pi/n)}{\sqrt{1-x_j^2}}, \quad \text{for } j = 1, \ldots, n-1. \tag{6.36}$$

Moreover, from (6.33),

$$\Pi_n'(j\pi/n) = -\sum_{k=1}^{n-1} kc_k \sin(kj\pi/n), \quad \text{for } j = 1, \ldots, n-1. \tag{6.37}$$

The right hand side can be identified (up to a normalization factor) as the inverse (type I) discrete sine transform (DST) of the coefficients $-kc_k$, $k = 1, \ldots, n-1$, which, like the DCT, can be computed in $O(n \log n)$ operations. Therefore, the procedure for Chebyshev spectral differentiation can be summarized as follows:

1. Compute the coefficients c_k, $k = 0, 1, \ldots, n$ with the fast DCT.

2. Evaluate (6.37) using the fast (inverse) DST.

3. Evaluate p_n' at the interior points using (6.36).

4. Compute $p_n'(\pm 1)$ using the formulas

$$p_n'(1) = \sum_{k=0}^{n}{}'' \, k^2 c_k, \tag{6.38}$$

$$p_n'(-1) = \sum_{k=0}^{n}{}'' (-1)^{k+1} k^2 c_k, \tag{6.39}$$

which follow by applying L'Hôpital's rule to $-\Pi_n'(\theta)/\sin(\theta)$.

Example 6.8. *Consider the smooth function $f(x) = e^{-x}\sin 2\pi x$ in $[-1,1]$. Figure 6.3(a) shows a plot of the derivative of f and the Chebyshev spectral approximation, computed via the fast DCT and DST, at the Chebychev nodes $x_j = \cos(\pi j/n)$, $j = 0, 1, \ldots, n$. The maximum value of the derivative in $[-1,1]$, $\|f'\|_\infty$, is attained at $x = -1$ and it is large (about 17) so it is more appropriate to consider the relative error in the approximation. This is shown in Fig. 6.3(b) for $n = 8, 16, 32$ (note the logarithmic scale on the vertical axis). There is a clear fast convergence and with $n = 32$ it is possible to achieve a relative error of $O(10^{-14})$.*

6.5 Bibliographic Notes

Section 6.1 . The presentation of finite differences was limited to equi-spaced points but formulas for arbitrary node distributions could also be obtained. Fornberg [For96] presents an algorithm for finding finite difference formulas for arbitrarily spaced grids.

Section 6.2 . The possibility of using Cauchy's integral formula to approximate the derivative is mentioned in Gautschi's book [Gau11]. Our discussion of the loss of accuracy of finite differences for small h follows Section 3.1 of that book.

Section 6.3 . As mentioned in the bibliographic notes in Chapter 1, Richardson's extrapolation is named after L. F. Richardson, who employed the procedure in 1911 [Ric11] for finite differences. Brenzinski [Bre10] and Gustafsson [Gus18] point out that the procedure actually goes back to the mid 1600's and the work of Christiaan Huygens [Huy09] to find approximations of π. Richardson's extrapolation is a very useful and general technique in

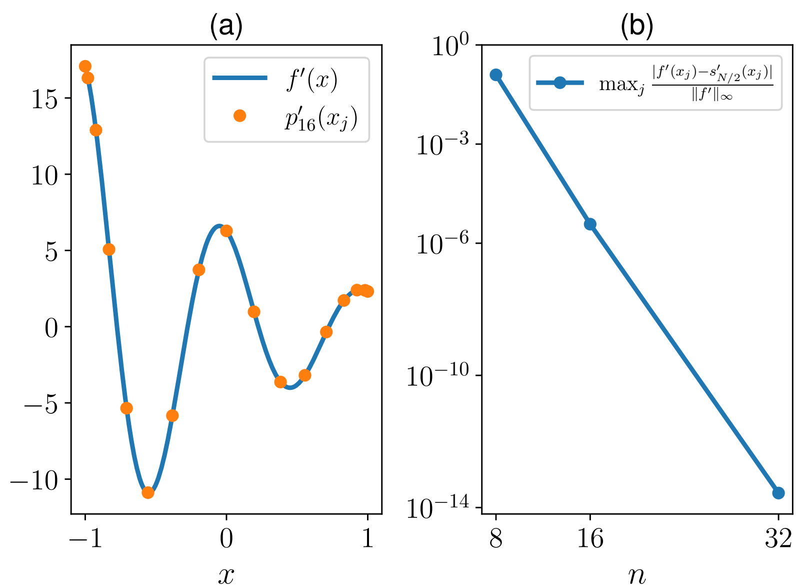

Figure 6.3: Chebychev spectral approximation of the derivative of $f(x) = e^{-x}\sin 2\pi x$ at $x_j = \cos(\pi j/n)$, $j = 0, 1, \ldots, n$. (a) f' and $p'_{16}(x_j)$ and (b) the maximum relative error $\max\limits_j |f'(x_j) - s'_{N/2}(x_j)|/\|f'\|_\infty$ for $n = 8, 16, 32$.

numerical analysis.

Section 6.4 . Spectral differentiation is central in the construction of spectral methods for differential equations. It is covered in the more specialized texts on the subject. In particular, spectral differentiation via the FFT is discussed in detail in [Tre00] and in [For96] Appendix F.

Chapter 7

Numerical Integration

We revisit now the problem of numerical integration that we used as an example to introduce some principles of numerical analysis in Chapter 1.

The problem in question is to find accurate and efficient approximations of

$$\int_a^b f(x)dx.$$

Numerical formulas to approximate a definite integral are called *quadratures* or *quadrature rules* and, as we saw in Chapter 1, they can be elementary (simple) or composite.

We shall assume henceforth, unless otherwise noted, that the integrand is sufficiently smooth.

7.1 Elementary Simpson's Rule

The elementary trapezoidal rule quadrature is derived by replacing the integrand f by its linear interpolating polynomial p_1 at a and b. That is,

$$f(x) = p_1(x) + \frac{1}{2}f''(\xi)(x-a)(x-b), \tag{7.1}$$

where $\xi \in (a,b)$ and so

$$\begin{aligned} \int_a^b f(x)dx &= \int_a^b p_1(x)dx + \frac{1}{2}\int_a^b f''(\xi)(x-a)(x-b)dx \\ &= \frac{1}{2}(b-a)[f(a)+f(b)] - \frac{1}{12}f''(\eta)(b-a)^3. \end{aligned} \tag{7.2}$$

Thus, the approximation

$$\int_a^b f(x)dx \approx \frac{1}{2}(b-a)[f(a)+f(b)] \tag{7.3}$$

has an error given by $-\frac{1}{12}f''(\eta)(b-a)^3$.

We can add an intermediate point, say the midpoint, and replace f by its quadratic interpolating polynomial p_2 with respect to the nodes a, $(a+b)/2$ and b.

For simplicity let's take $[a,b]=[-1,1]$. For a general interval $[a,b]$, we can use the transformation

$$x = \frac{1}{2}(a+b) + \frac{1}{2}(b-a)t. \tag{7.4}$$

Then, let p_2 be the interpolating polynomial of f at $-1,0,1$. The corresponding divided difference table is:

$$\begin{array}{cccc}
-1 & f(-1) & & \\
 & & f(0)-f(-1) & \\
0 & f(0) & & \frac{f(1)-2f(0)+f(-1)}{2}. \\
 & & f(1)-f(0) & \\
1 & f(1) & &
\end{array}$$

Thus,

$$\begin{aligned} p_2(x) = {} & f(-1) + [f(0)-f(-1)]\,(x+1) \\ & + \frac{1}{2}\,[f(1)-2f(0)+f(-1)]\,(x+1)x. \end{aligned} \tag{7.5}$$

Now, employing the interpolation formula with the remainder expressed in terms of a divided difference (3.64) we have

$$\begin{aligned} f(x) &= p_2(x) + f[-1,0,1,x](x+1)x(x-1) \\ &= p_2(x) + f[-1,0,1,x]x(x^2-1). \end{aligned} \tag{7.6}$$

Therefore,

$$\int_{-1}^{1} f(x)dx = \int_{-1}^{1} p_2(x)dx + \int_{-1}^{1} f[-1,0,1,x]x(x^2-1)dx. \tag{7.7}$$

We can easily evaluate the first integral on the right hand side to obtain the (elementary) Simpson's rule:

$$\int_{-1}^{1} p_2(x)dx = \frac{1}{3}[f(-1) + 4f(0) + f(1)]. \tag{7.8}$$

The error by approximating the integral of f with this quadrature is

$$E[f] = \int_{-1}^{1} f[-1, 0, 1, x]x(x^2 - 1)dx. \tag{7.9}$$

Note that $x(x^2 - 1)$ changes sign in $[-1, 1]$, so we cannot use the mean value theorem for integrals to estimate the error as we did in the trapezoidal rule quadrature. However, $x(x^2 - 1)$ is an odd function in $[-1, 1]$ and thus if $f[-1, 0, 1, x]$ were constant, as it is the case for polynomial of degree 3 or less, the error would be zero. In other words, the quadrature (7.8) is exact if $f \in \mathbb{P}_3$. We can take advantage of this fact by introducing another node, x_4, and relating $f[-1, 0, 1, x]$ to the constant divided difference $f[-1, 0, 1, x_4]$ and to the fourth order divided difference $f[-1, 0, 1, x_4, x]$:

$$f[-1, 0, 1, x] = f[-1, 0, 1, x_4] + f[-1, 0, 1, x_4, x](x - x_4). \tag{7.10}$$

This identity is just an application of Theorem 3.2. Substituting (7.10) into (7.9) we get

$$E[f] = \int_{-1}^{1} f[-1, 0, 1, x_4, x]x(x^2 - 1)(x - x_4)dx. \tag{7.11}$$

We choose now $x_4 = 0$ so that $x(x^2 - 1)(x - x_4)$ does not change sign in $[-1, 1]$ and we obtain

$$E[f] = \int_{-1}^{1} f[-1, 0, 0, 1, x]x^2(x^2 - 1)dx, \tag{7.12}$$

where we also used that $f[-1, 0, 0, 1, x] = f[-1, 0, 1, 0, x]$. If $f \in C^4[-1, 1]$, there is $\xi(x) \in (-1, 1)$ such that [(3.66)]

$$f[-1, 0, 0, 1, x] = \frac{f^{(4)}(\xi(x))}{4!}, \tag{7.13}$$

and consequently, by the mean value theorem for integrals, there is $\eta \in (-1,1)$ such that

$$E[f] = \frac{f^{(4)}(\eta)}{4!} \int_{-1}^{1} x^2(x^2-1)dx = -\frac{4}{15}\frac{f^{(4)}(\eta)}{4!} = -\frac{1}{90}f^{(4)}(\eta). \tag{7.14}$$

Summarizing, Simpson's quadrature rule for the interval $[-1,1]$ is

$$\int_{-1}^{1} f(x)dx = \frac{1}{3}[f(-1)+4f(0)+f(1)] - \frac{1}{90}f^{(4)}(\eta). \tag{7.15}$$

Note again that this quadrature gives the exact value of the integral when f is a polynomial of degree 3 or less (the error is proportional to the fourth derivative), even though we used a polynomial of degree at most 2 to approximate the integrand. This extra gain is due to the symmetry of the quadrature around 0. In fact, we could have derived Simpson's quadrature by using the Hermite (third order) interpolating polynomial of f at $-1, 0, 0, 1$.

For a general interval $[a,b]$ we use the change of variables (7.4)

$$\int_a^b f(x)dx = \frac{1}{2}(b-a)\int_{-1}^{1} F(t)dt,$$

where

$$F(t) = f\left(\frac{1}{2}(a+b) + \frac{1}{2}(b-a)t\right), \tag{7.16}$$

and noting that $F^{(k)}(t) = (\frac{b-a}{2})^k f^{(k)}(x)$ we obtain the (elementary) Simpson's rule on the interval $[a,b]$:

$$\begin{aligned}\int_a^b f(x)dx = \frac{1}{6}(b-a)\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] \\ - \frac{1}{90}f^{(4)}(\eta)\left(\frac{b-a}{2}\right)^5.\end{aligned} \tag{7.17}$$

7.2 Interpolatory Quadratures

The elementary trapezoidal and Simpson's rules are examples of *interpolatory quadratures*. This class of quadratures is obtained by selecting a set of nodes

$x_0, x_1, \ldots, x_n$ in the interval of integration and by approximating the integral by that of the interpolating polynomial p_n of the integrand at these nodes. By construction, such interpolatory quadrature is exact for polynomials of degree up to n, at least. We just saw that Simpson's rule is exact for polynomial up to degree 3 and we used an interpolating polynomial of degree at most 2 in its construction. The "degree gain" is due to the symmetric choice of the interpolation nodes. This leads us to two important questions:

1. For a given n, how do we choose the nodes $x_0, x_1, \ldots, x_n$ so that the corresponding interpolation quadrature is exact for polynomials of the highest degree possible?

2. What is that maximal degree?

Because orthogonal polynomials (Section 4.1.2.2) play a central role in the answer to these questions, we will consider the more general problem of approximating the integral

$$I[f] = \int_a^b f(x)w(x)dx, \tag{7.18}$$

where w is an admissible weight function[1], $w \equiv 1$ being a particular case. The interval of integration $[a, b]$ can be either finite or infinite (e.g. $[0, +\infty]$, $[-\infty, +\infty]$).

Definition 7.1. *We say that a quadrature $Q[f]$ to approximate $I[f]$ has degree of precision k if it is exact for all $p \in \mathbb{P}_k$ but not exact for polynomials of degree $k + 1$. Equivalently, a quadrature $Q[f]$ has degree of precision k if $I[x^m] = Q[x^m]$, for $m = 0, 1, \ldots, k$ but $I[x^{k+1}] \neq Q[x^{k+1}]$.*

Example 7.1. *The trapezoidal rule quadrature has degree of precision 1 while Simpson's quadrature has degree of precision 3.*

For a given set of nodes $x_0, x_1, \ldots, x_n$ in $[a, b]$, let p_n be the interpolating polynomial of f at these nodes. In Lagrange form, we can write p_n as (see Section 3.1)

$$p_n(x) = \sum_{j=0}^{n} f(x_j)l_j(x), \tag{7.19}$$

[1] $w \geq 0$, $\int_a^b w(x)dx > 0$, and $\int_a^b x^k w(x)dx < +\infty$ for $k = 0, 1, \ldots$

where

$$l_j(x) = \prod_{\substack{k=0 \\ k\neq j}}^{n} \frac{(x-x_k)}{(x_j-x_k)}, \qquad \text{for } j = 0, 1, ..., n. \tag{7.20}$$

are the polynomial cardinal functions. The corresponding interpolatory quadrature $Q_n[f]$ to approximate $I[f]$ is then given by

$$\begin{aligned} Q_n[f] &= \sum_{j=0}^{n} A_j f(x_j), \\ A_j &= \int_a^b l_j(x)w(x)dx, \text{ for } j = 0, 1, \ldots, n. \end{aligned} \tag{7.21}$$

Theorem 7.1. *Degree of precision of the interpolatory quadrature (7.21) is less than* $2n+2$

Proof. Suppose the degree of precision k of (7.21) is greater or equal than $2n+2$. Take $f(x) = (x-x_0)^2(x-x_1)^2\cdots(x-x_n)^2$. This is polynomial of degree exactly $2n+2$. Then.

$$\int_a^b f(x)w(x)dx = \sum_{j=0}^{n} A_j f(x_j) = 0. \tag{7.22}$$

On the other hand

$$\int_a^b f(x)w(x)dx = \int_a^b (x-x_0)^2\cdots(x-x_n)^2 w(x)dx > 0 \tag{7.23}$$

which is a contradiction. Therefore $k < 2n+2$. □

7.3 Gaussian Quadratures

We will now show that there is a choice of nodes $x_0, x_1, ..., x_n$ that yields the maximal degree of precision, $2n+1$, for an interpolatory quadrature. The corresponding quadratures are called Gaussian quadratures. To define them, we recall that ψ_k is the k-th orthogonal polynomial with respect to the inner product

$$< f, g >= \int_a^b f(x)g(x)w(x)dx, \tag{7.24}$$

if $<\psi_k, q>= 0$ for all polynomials q of degree less than k. Recall also that the zeros of orthogonal polynomials are real, simple, and contained in $[a, b]$ (Theorem 4.3).

Definition 7.2. *Let ψ_{n+1} be the $(n+1)$st orthogonal polynomial and let $x_0, x_1, ..., x_n$ be its $n+1$ zeros. Then, the interpolatory quadrature (7.21) with the nodes so chosen is called a Gaussian quadrature.*

Theorem 7.2. *The interpolatory quadrature (7.21) has maximal degree of precision $k = 2n+1$ if and only if it is a Gaussian quadrature.*

Proof. Let us suppose that the quadrature is Gaussian and let f be a polynomial of degree $\leq 2n+1$. Then, we can write

$$f(x) = q(x)\psi_{n+1}(x) + r(x), \tag{7.25}$$

where q and r are polynomials of degree $\leq n$. Now

$$\int_a^b f(x)w(x)dx = \int_a^b q(x)\psi_{n+1}(x)w(x)dx + \int_a^b r(x)w(x)dx \tag{7.26}$$

The first integral on the right hand side is zero because of orthogonality. For the second integral the quadrature is exact (it is interpolatory). Therefore,

$$\int_a^b f(x)w(x)dx = \sum_{j=0}^{n} A_j r(x_j). \tag{7.27}$$

Moreover, $r(x_j) = f(x_j) - q(x_j)\psi_{n+1}(x_j) = f(x_j)$ for all $j = 0, 1, \ldots, n$. Thus,

$$\int_a^b f(x)w(x)dx = \sum_{j=0}^{n} A_j f(x_j) \tag{7.28}$$

and consequently, the Gaussian quadrature has degree of precision $k = 2n+1$. Now suppose that the interpolatory quadrature (7.21) has maximal degree of precision $2n+1$. Take $f(x) = p(x)(x-x_0)(x-x_1)\cdots(x-x_n)$ where p is a polynomial of degree $\leq n$. Then, f is a polynomial of degree $\leq 2n+1$ and

$$\int_a^b f(x)w(x)dx = \int_a^b p(x)(x-x_0)\cdots(x-x_n)w(x)dx = \sum_{j=0}^{n} A_j f(x_j) = 0.$$

Therefore, the polynomial $(x-x_0)(x-x_1)\cdots(x-x_n)$ of degree $n+1$ is orthogonal to all polynomials of degree $\leq n$. Thus, it is a multiple of ψ_{n+1}.

$\square$

Example 7.2. *Consider the interval* $[-1,1]$ *and the weight function* $w \equiv 1$. *The orthogonal polynomials are the Legendre polynomials* $1, x, x^2 - \frac{1}{3}, x^3 - \frac{3}{5}x$, *etc. Take* $n = 1$. *The roots of* ψ_2 *are* $x_0 = -\sqrt{\frac{1}{3}}$ *and* $x_1 = \sqrt{\frac{1}{3}}$. *Therefore, the corresponding Gaussian quadrature is*

$$\int_{-1}^{1} f(x)dx \approx A_0 f\left(-\sqrt{\frac{1}{3}}\right) + A_1 f\left(\sqrt{\frac{1}{3}}\right), \tag{7.29}$$

where

$$A_0 = \int_{-1}^{1} l_0(x)dx, \tag{7.30}$$

$$A_1 = \int_{-1}^{1} l_1(x)dx. \tag{7.31}$$

We can evaluate these integrals directly or employ the **method of undetermined coefficients** *to find* A_0 *and* A_1. *The latter is generally easier and we illustrate it now. Using that the quadrature is exact for* 1 *and* x *we have*

$$2 = \int_{-1}^{1} 1\,dx = A_0 + A_1, \tag{7.32}$$

$$0 = \int_{-1}^{1} x\,dx = -A_0\sqrt{\frac{1}{3}} + A_1\sqrt{\frac{1}{3}}. \tag{7.33}$$

Solving this 2×2 *linear system we get* $A_0 = A_1 = 1$. *So the Gaussian quadrature for* $n = 1$ *in* $[-1,1]$ *is*

$$Q_1[f] = f\left(-\sqrt{\frac{1}{3}}\right) + f\left(\sqrt{\frac{1}{3}}\right). \tag{7.34}$$

Let us compare this quadrature to the elementary trapezoidal rule. Take $f(x) = x^2$. *The trapezoidal rule,* $T[f]$, *gives*

$$T[x^2] = \frac{2}{2}\left[f(-1) + f(1)\right] = 2, \tag{7.35}$$

whereas the Gaussian quadrature $Q_1[f]$ *yields the exact result:*

$$Q_1[x^2] = \left(-\sqrt{\frac{1}{3}}\right)^2 + \left(\sqrt{\frac{1}{3}}\right)^2 = \frac{2}{3}. \tag{7.36}$$

Example 7.3. *Consider the interval* $[-1, 1]$ *and* $w(x) = (1 - x^2)^{-1/2}$. *As we know (see 2.4),* $\psi_{n+1} = T_{n+1}$, *the Chebyshev polynomial of degree* $n + 1$. *Its zeros are*

$$x_j = \cos\left(\frac{2j + 1}{2(n + 1)}\pi\right), \quad \text{for } j = 0, \dots, n.$$

For $n = 1$, *we have*

$$\cos\left(\frac{\pi}{4}\right) = \sqrt{\frac{1}{2}}, \quad \cos\left(\frac{3\pi}{4}\right) = -\sqrt{\frac{1}{2}}. \tag{7.37}$$

We can use again the method of undetermined coefficients to find A_0 *and* A_1*:*

$$\pi = \int_{-1}^{1} 1 \frac{1}{\sqrt{1 - x^2}} dx = A_0 + A_1, \tag{7.38}$$

$$0 = \int_{-1}^{1} x \frac{1}{\sqrt{1 - x^2}} dx = -A_0 \sqrt{\frac{1}{2}} + A_1 \sqrt{\frac{1}{2}}, \tag{7.39}$$

which gives $A_0 = A_1 = \frac{\pi}{2}$. *Thus, the corresponding Gaussian quadrature to approximate*

$$\int_{-1}^{1} f(x) \frac{1}{\sqrt{1 - x^2}} dx$$

is

$$Q_1[f] = \frac{\pi}{2}\left[f\left(-\sqrt{\frac{1}{2}}\right) + f\left(\sqrt{\frac{1}{2}}\right)\right]. \tag{7.40}$$

7.3.1 Convergence of Gaussian Quadratures

Let f be a continuous function on a closed interval $[a, b]$ and consider the interpolation quadrature (7.21). Can we guarantee that the error converges to zero as $n \to \infty$, i.e.,

$$\int_a^b f(x)w(x)dx - \sum_{j=0}^{n} A_j f(x_j) \to 0, \text{ as } n \to \infty \text{ ?}$$

The answer is no; recall that the convergence of the interpolating polynomial to f depends on the smoothness of f and the distribution of the interpolating nodes. However, if the interpolatory quadrature is Gaussian the answer is yes. This is a consequence of the following special properties of the weights $A_0, A_1, \dots, A_n$ in the Gaussian quadrature and Weierstrass approximation theorem.

Theorem 7.3. *For a Gaussian quadrature all the weights are positive and sum up to* $\|w\|_1$*, i.e.,*

(a) $A_j > 0$ *for all* $j = 0, 1, \dots, n$.

(b) $\displaystyle\sum_{j=0}^{n} A_j = \int_a^b w(x)dx.$

Proof. (a) Let $p_k = l_k^2$ for $k = 0, 1, \dots, n$. These are polynomials of degree $2n$ and $p_k(x_j) = \delta_{kj}$. Thus,

$$0 < \int_a^b l_k^2(x)w(x)dx = \sum_{j=0}^{n} A_j l_k^2(x_j) = A_k \tag{7.41}$$

for $k = 0, 1, \dots, n$.
(b) Take $f(x) \equiv 1$ then

$$\int_a^b w(x)dx = \sum_{j=0}^{n} A_j. \tag{7.42}$$

as the quadrature is exact for polynomials of degree zero. □

We can now use these special properties of the Gaussian quadrature to prove its convergence for all continuous functions f on a closed bounded interval $[a, b]$.

Theorem 7.4. *Let*

$$Q_n[f] = \sum_{j=0}^{n} A_j f(x_j) \tag{7.43}$$

be the Gaussian quadrature. Then,

$$E_n[f] := \int_a^b f(x)w(x)dx - Q_n[f] \to 0, \ \text{as } n \to \infty\ . \tag{7.44}$$

Proof. Let p_{2n+1}^* be the best uniform approximation to f (i.e. the best approximation in the norm $\|f\|_\infty = \max_{x\in[a,b]} |f(x)|$) by polynomials of degree $\leq 2n + 1$. Then,

$$E_n[f - p_{2n+1}^*] = E_n[f] - E_n[p_{2n+1}^*] = E_n[f] \tag{7.45}$$

and therefore

$$
\begin{aligned}
E_n[f] &= E_n[f - p^*_{2n+1}] \\
&= \int_a^b [f(x) - p^*_{2n+1}(x)]w(x)dx - \sum_{j=0}^{n} A_j[f(x_j) - p^*_{2n+1}(x_j)].
\end{aligned} \tag{7.46}
$$

Taking the absolute value, using the triangle inequality, and the fact that the weights are positive we obtain

$$
\begin{aligned}
|E_n[f]| &\leq \int_a^b |f(x) - p^*_{2n+1}(x)|w(x)dx + \sum_{j=0}^{n} A_j|f(x_j) - p^*_{2n+1}(x_j)| \\
&\leq \|f - p^*_{2n+1}\|_\infty \int_a^b w(x)dx + \|f - p^*_{2n+1}\|_\infty \sum_{j=0}^{n} A_j \\
&= 2\|w\|_1 \|f - p^*_{2n+1}\|_\infty
\end{aligned}
$$

From the Weierstrass approximation theorem it follows that

$$
\|f - p^*_{2n+1}\|_\infty \to 0, \quad \text{as } n \to \infty \tag{7.47}
$$

and consequently $E_n[f] \to 0$ as $n \to \infty$. □

Moreover, it can be proved that if $f \in C^m[a, b]$

$$
|E_n[f]| \leq C(2n)^{-m}\|f^{(m)}\|_\infty. \tag{7.48}
$$

That is, the rate of convergence is not fixed; it depends on the number of derivatives the integrand has. In this case, we say that the approximation is *spectral*. In particular if $f \in C^\infty[a, b]$ then the error decreases down to zero faster than any power of $1/(2n)$.

7.3.2 Computing the Gaussian Nodes and Weights

Gaussian quadratures achieve high accuracy for smooth functions with a relatively small n. However, we need to compute numerically their nodes and weights (in most cases). One of the most popular methods to do this, at least for moderate n, is a method based on an eigenvalue problem, as we show next. More recent, fast and effective methods use polynomial root-finding in combination with Taylor series approximations or asymptotic expansions.

Orthogonal polynomials satisfy a three-term relation:

$$\psi_{k+1}(x) = (x - \alpha_k)\psi_k(x) - \beta_k \psi_{k-1}(x), \quad \text{for } k = 0, 1, \ldots, n, \tag{7.49}$$

where $\beta_0 = \int_a^b w(x)dx$, $\psi_0(x) = 1$ and $\psi_{-1}(x) = 0$. For several orthogonal polynomials the coefficients α_k, β_k are known. With this information, the problem of finding the Gaussian nodes and weights can be efficiently solved, as we show next.

We start by rewriting (7.49) as

$$x\psi_k(x) = \beta_k \psi_{k-1}(x) + \alpha_k \psi_k(x) + \psi_{k+1}(x), \quad \text{for } k = 0, 1, \ldots, n. \tag{7.50}$$

If we use the normalized orthogonal polynomials

$$\tilde{\psi}_k(x) = \frac{\psi_k(x)}{\sqrt{< \psi_k, \psi_k >}} \tag{7.51}$$

and recalling that

$$\beta_k = \frac{< \psi_k, \psi_k >}{< \psi_{k-1}, \psi_{k-1} >}$$

then (7.50) can be written as

$$x\tilde{\psi}_k(x) = \sqrt{\beta_k}\,\tilde{\psi}_{k-1}(x) + \alpha_k \tilde{\psi}_k(x) + \sqrt{\beta_{k+1}}\,\tilde{\psi}_{k+1}(x), \tag{7.52}$$

for $k = 0, 1, \ldots, n$. Now, evaluating this expression at a root x_j of ψ_{n+1} we get the eigenvalue problem

$$x_j v^{(j)} = J_{n+1} v^{(j)}, \tag{7.53}$$

where

$$J_{n+1} = \begin{bmatrix} \alpha_0 & \sqrt{\beta_1} & 0 & \cdots & 0 \\ \sqrt{\beta_1} & \alpha_1 & \sqrt{\beta_2} & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \\ & & & & \sqrt{\beta_n} \\ 0 & 0 & 0 & \sqrt{\beta_n} & \alpha_n \end{bmatrix}, \qquad v^{(j)} = \begin{bmatrix} \tilde{\psi}_0(x_j) \\ \tilde{\psi}_1(x_j) \\ \vdots \\ \tilde{\psi}_{n-1}(x_j) \\ \tilde{\psi}_n(x_j) \end{bmatrix}. \tag{7.54}$$

That is, the Gaussian nodes x_j, $j = 0, 1, \ldots, n$ are the eigenvalues of the $(n+1) \times (n+1)$, symmetric, tridiagonal matrix J_{n+1} with corresponding eigenvectors $v^{(j)}$, $j = 0, 1, \ldots, n$. There are efficient numerical methods (the

QR method described in Section 11.3) to solve the eigenvalue problem for a symmetric, tridiagonal matrix and this is one of most popular approaches to compute the Gaussian nodes and weights.

We derive now a formula to obtain the Gaussian weights A_j. Since $\langle\tilde{\psi}_k, \tilde{\psi}_0\rangle = 0$ for $k = 1, \ldots, n$ and the quadrature is exact for polynomial of degree $\leq 2n + 1$, it follows that

$$0 = \int_a^b \tilde{\psi}_k(x)w(x)dx = \sum_{j=0}^{n} A_j\tilde{\psi}_k(x_j), \quad k = 1, \ldots, n \tag{7.55}$$

and together with the relation

$$\sqrt{\beta_0} = \int_a^b \tilde{\psi}_0(x)w(x)dx = \sum_{j=0}^{n} A_j\tilde{\psi}_0(x_j), \tag{7.56}$$

we obtain the following linear system of equations for the weights

$$\begin{bmatrix} v^{(0)} & v^{(1)} & \cdots & v^{(n)} \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ \vdots \\ A_n \end{bmatrix} = \begin{bmatrix} \sqrt{\beta_0} \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{7.57}$$

Left-multiplying this expression by $v^{(j)T}$ and noting that for a symmetric matrix eigenvectors corresponding to different eigenvalues are orthogonal, we get

$$v^{(j)T}v^{(j)}A_j = \sqrt{\beta_0}\, v_0^{(j)} = 1, \tag{7.58}$$

where the last equality follows from $v_0^{(j)} = \tilde{\psi}_0(x_j) = 1/\sqrt{\beta_0}$. Now, if we use the normalized eigenvectors $u^{(j)} = v^{(j)}/\|v^{(j)}\|$ we note that $u_0^{(j)} = 1/(\|v^{(j)}\|\sqrt{\beta_0})$. Hence, multiplying (7.58) by $\left(u_0^{(j)}\right)^2$, we obtain

$$A_j = \beta_0 \left(u_0^{(j)}\right)^2, \quad j = 0, 1, \ldots, n. \tag{7.59}$$

7.4 Clenshaw-Curtis Quadrature

Gaussian quadratures are optimal in terms of the degree of precision and offer superalgebraic convergence for smooth integrands. However, the computation of Gaussian weights and nodes carries a significant cost, for large

n. There is an ingenious interpolatory quadrature that is a close competitor to the Gaussian quadrature due to its efficient and fast rate of convergence. This is the Clenshaw-Curtis quadrature, which we derive next.

Suppose f is a smooth function on the interval $[-1, 1]$ and we are interested in an accurate approximation of the integral

$$\int_{-1}^{1} f(x)dx.$$

The idea is to use the Chebyshev nodes $x_j = \cos(j\pi/n)$, $j = 0, 1, ..., n$ as the nodes of the corresponding interpolatory quadrature. The degree of precision is only n (or $n+1$ if n is even), not $2n+1$. However, as we know, for smooth functions the approximation by polynomial interpolation using the Chebyshev nodes converges rapidly. Hence, for smooth integrands this particular interpolatory quadrature can be expected to converge fast to the exact value of the integral.

As seen in Section 3.13, the interpolating polynomial p_n of f at the Chebyshev nodes (the Chebyshev interpolant) can be represented as

$$p_n(x) = \frac{a_0}{2} + \sum_{k=1}^{n-1} a_k T_k(x) + \frac{a_n}{2} T_n(x), \tag{7.60}$$

where the coefficients are given by

$$a_k = \frac{2}{n} \sum_{j=0}^{n}{}'' f(\cos\theta_j)\cos k\theta_j, \quad \theta_j = j\pi/n, \quad k = 0, 1, .., n. \tag{7.61}$$

These coefficients can be computed efficiently in $O(n \log n)$ operations with the fast DCT or with the FFT. With the change of variable $x = \cos\theta$, $\theta \in [0, \pi]$, we have

$$p_n(\cos\theta) = \frac{a_0}{2} + \sum_{k=1}^{n-1} a_k \cos k\theta + \frac{a_n}{2}\cos n\theta \tag{7.62}$$

and

$$\int_{-1}^{1} f(x)dx = \int_{0}^{\pi} f(\cos\theta)\sin\theta d\theta. \tag{7.63}$$

The quadrature is obtained by replacing $f(\cos\theta)$ by $p_n(\cos\theta)$

$$\int_{-1}^{1} f(x)dx \approx \int_{0}^{\pi} p_n(\cos\theta)\sin\theta d\theta. \tag{7.64}$$

Substituting (7.62) for $p_n(\cos\theta)$ we get

$$\begin{aligned}\int_{0}^{\pi} p_n(\cos\theta)\sin\theta d\theta = & \frac{a_0}{2}\int_{0}^{\pi}\sin\theta d\theta \\ & + \sum_{k=1}^{n-1} a_k \int_{0}^{\pi}\cos k\theta \sin\theta d\theta \\ & + \frac{a_n}{2}\int_{0}^{\pi}\cos n\theta\sin\theta d\theta.\end{aligned} \tag{7.65}$$

With the aid of the trigonometric identity

$$\cos k\theta \sin\theta = \frac{1}{2}[\sin(1+k)\theta + \sin(1-k)\theta] \tag{7.66}$$

we can perform the integrals on the right hand side of (7.65) and taking n even we get the Clenshaw-Curtis quadrature:

$$\int_{-1}^{1} f(x)dx \approx a_0 + \sum_{\substack{k=2 \\ k \text{ even}}}^{n-2} \frac{2a_k}{1-k^2} + \frac{a_n}{1-n^2}. \tag{7.67}$$

For a general interval $[a, b]$ we simply use the change of variables

$$x = \frac{a+b}{2} + \frac{b-a}{2}\cos\theta \tag{7.68}$$

for $\theta \in [0, \pi]$ and thus

$$\int_{a}^{b} f(x)dx = \frac{b-a}{2}\int_{0}^{\pi} F(\theta)\sin\theta d\theta, \tag{7.69}$$

where $F(\theta) = f(\frac{a+b}{2} + \frac{b-a}{2}\cos\theta)$ and so the formula (7.67) gets an extra factor of $(b-a)/2$.

Figure 7.1 shows a comparison of the approximations obtained with the Clenshaw-Curtis quadrature and the composite Simpson quadrature, which we discuss next, for the integral of $f(x) = e^x$ in $[0, 1]$. The Clenshaw-Curtis quadrature converges to the exact value of the integral notably fast. With just $n = 8$ nodes, it almost reaches machine precision while the composite Simpson rule requires more than 512 nodes for comparable accuracy.

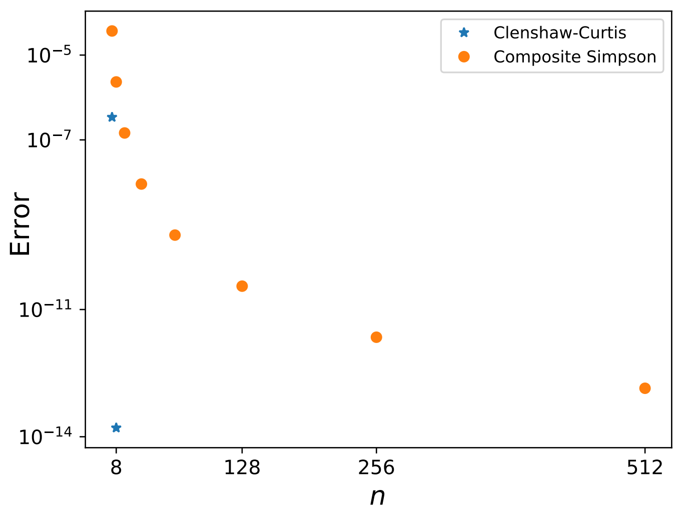

Figure 7.1: Clenshaw-Curtis quadrature and the composite Simpson rule for the integral of $f(x) = e^x$ in $[0, 1]$. The Clenshaw-Curtis almost reaches machine precision with just $n = 8$ nodes.

7.5 Composite Quadratures

We saw in Section 1.2.2 that one strategy to improve the accuracy of an elementary (simple) quadrature formula is to divide the interval of integration $[a, b]$ into small subintervals, use the elementary quadrature in each of them, and sum up all the contributions.

For simplicity, let us divide uniformly $[a, b]$ into N subintervals of equal length $h = (b - a)/N$, $[x_j, x_{j+1}]$, where $x_j = a + jh$ for $j = 0, 1, \ldots, N - 1$. If we use the elementary trapezoidal rule in each subinterval (as done in Section 1.2.2) we arrive at the composite trapezoidal rule:

$$\int_a^b f(x)dx = h\left[\frac{1}{2}f(a) + \sum_{j=1}^{N-1} f(x_j) + \frac{1}{2}f(b)\right] - \frac{1}{12}(b-a)h^2 f''(\eta), \quad (7.70)$$

where η is some point in (a, b).

To derive a corresponding *composite* Simpson's rule we take N even and split the integral over the $N/2$ intervals $[x_0, x_2], [x_2, x_4], \ldots [x_{N-2}, x_N]$:

$$\int_a^b f(x)dx = \int_{x_0}^{x_2} f(x)dx + \int_{x_2}^{x_4} f(x)dx + \cdots + \int_{x_{N-2}}^{x_N} f(x)dx. \quad (7.71)$$

Since the elementary Simpson's quadrature applied to $[x_j, x_{j+2}]$ is

$$\int_{x_j}^{x_{j+2}} f(x)dx = \frac{h}{3}\left[f(x_j) + 4f(x_{j+1}) + f(x_{j+2})\right] - \frac{1}{90}f^{(4)}(\eta_j)h^5, \quad (7.72)$$

for some $\eta_j \in (x_j, x_{j+2})$, summing up all the $N/2$ contributions we get the composite Simpson's rule:

$$\int_a^b f(x)dx = \frac{h}{3}\left[f(a) + 2\sum_{j=1}^{N/2-1} f(x_{2j}) + 4\sum_{j=1}^{N/2} f(x_{2j-1}) + f(b)\right] - \frac{1}{180}(b-a)h^4 f^{(4)}(\eta),$$

for some $\eta \in (a, b)$.

7.6 Modified Trapezoidal Rule

We consider now a modification to the trapezoidal rule that will yield a quadrature with an error of the same order as that of Simpson's rule. Moreover, this modified quadrature will give us some insight into the asymptotic form of error for the trapezoidal rule.

To simplify the derivation let us take the interval $[0,1]$ and let p_3 be the polynomial interpolating $f(0), f'(0), f(1), f'(1)$:

$$p_3(x) = f(0) + f[0,0]x + f[0,0,1]x^2 + f[0,0,1,1]x^2(x-1). \tag{7.73}$$

Thus,

$$\int_0^1 p_3(x)dx = f(0) + \frac{1}{2}f'(0) + \frac{1}{3}f[0,0,1] - \frac{1}{12}f[0,0,1,1]. \tag{7.74}$$

The divided differences are obtained in the tableau:

$$\begin{array}{c|lllll}
0 & f(0) & & & & \\
 & & f'(0) & & & \\
0 & f(0) & & f(1)-f(0)-f'(0) & & \\
 & & f(1)-f(0) & & f'(1)+f'(0)+2(f(0)-f(1)) & \\
1 & f(1) & & f'(1)-f(1)+f(0) & & \\
 & & f'(1) & & & \\
1 & f(1) & & & &
\end{array}$$

Therefore,

$$\begin{aligned}\int_0^1 p_3(x)dx &= f(0) + \frac{1}{2}f'(0) + \frac{1}{3}[f(1)-f(0)-f'(0)] \\ &\quad - \frac{1}{12}[f'(0)+f'(1)+2(f(0)-f(1))]\end{aligned} \tag{7.75}$$

and simplifying the right hand side we get

$$\int_0^1 p_3(x)dx = \frac{1}{2}[f(0)+f(1)] - \frac{1}{12}[f'(1)-f'(0)]. \tag{7.76}$$

This is the simple trapezoidal rule plus a correction involving the derivative of the integrand at the end points. We already obtained this quadrature in Section 1.2.4 using the error correction technique. We can now be more precise about the error of this approximation by recalling that, assuming $f \in C^4[0,1]$,

$$f(x) - p_3(x) = \frac{1}{4!} f^{(4)}(\xi(x)) x^2 (x-1)^2, \quad \forall x \in [0,1], \tag{7.77}$$

for some $\xi(x) \in (0,1)$. Since $x^2(x-1)^2$ does not change sign in $[0,1]$ we can use the mean value theorem for integrals to get the following expression for the error

$$\begin{aligned} E[f] = \int_0^1 [f(x) - p_3(x)]dx &= \frac{1}{4!} f^{(4)}(\eta) \int_0^1 x^2(x-1)^2 dx \\ &= \frac{1}{720} f^{(4)}(\eta), \end{aligned} \tag{7.78}$$

for some $\eta \in (0,1)$.

To obtain the quadrature in a general, finite interval $[a,b]$ we use the change of variables $x = a + (b-a)t,\ t \in [0,1]$

$$\int_a^b f(x)dx = (b-a) \int_0^1 F(t)dt, \tag{7.79}$$

where $F(t) = f(a + (b-a)t)$. Thus,

$$\begin{aligned} \int_a^b f(x)dx = \frac{b-a}{2}[f(a) + f(b)] - \frac{(b-a)^2}{12}[f'(b) - f'(a)] \\ + \frac{1}{720} f^{(4)}(\eta)(b-a)^5, \end{aligned} \tag{7.80}$$

for some $\eta \in (a,b)$. This is the simple (or elementary) modified trapezoidal rule. We construct the corresponding composite quadrature by subdividing $[a,b]$ in N subintervals $[x_0,x_1], \ldots, [x_{N-1},x_N]$ of equal length $h = x_{j+1} - x_j = (b-a)/N$, applying the simple rule in each subinterval, and adding up all the contributions:

$$\begin{aligned} \int_a^b f(x)dx &= \sum_{j=0}^{N-1} \int_{x_j}^{x_{j+1}} f(x)dx \\ &= \frac{h}{2} \sum_{j=0}^{N-1} [f(x_j) + f(x_{j+1})] - \frac{h^2}{12} \sum_{j=0}^{N-1} [f'(x_{j+1}) - f'(x_{j+1})] \\ &+ \frac{1}{720} h^5 \sum_{j=0}^{N-1} f^{(4)}(\eta_j). \end{aligned} \tag{7.81}$$

Noticing that $f^{(4)}$ is continuous, there is $\eta \in (a,b)$ such that

$$f^{(4)}(\eta) = \frac{1}{N}\sum_{j=0}^{N-1} f^{(4)}(\eta_j) = \frac{(b-a)}{h}\sum_{j=0}^{N-1} f^{(4)}(\eta_j) \tag{7.82}$$

and since the sum with the the first derivative in (7.81) telescopes, we finally arrive at the composite, modified trapezoidal rule:

$$\begin{aligned}\int_a^b f(x)dx = h\left[\frac{1}{2}f(x_0) + \sum_{j=1}^{N-1} f(x_j) + \frac{1}{2}f(x_N)\right] - \frac{h^2}{12}[f'(b) - f'(a)] \\ + (b-a)\frac{1}{720}h^4 f^{(4)}(\eta).\end{aligned} \tag{7.83}$$

7.7 The Euler-Maclaurin Formula

Let us consider again the error in the composite, modified trapezoidal rule as it appears in (7.81). We have asymptotically

$$\begin{aligned}\frac{1}{720}h^5\sum_{j=0}^{N-1} f^{(4)}(\eta_j) &= \frac{1}{720}h^4\left(h\sum_{j=0}^{N-1} f^{(4)}(\eta_j)\right) \\ &= \frac{1}{720}h^4\int_a^b f^{(4)}(x)dx + o(h^4) \\ &= \frac{1}{720}h^4[f'''(b) - f'''(a)] + o(h^4).\end{aligned} \tag{7.84}$$

This expression for the error together with (7.83) suggest a formula of the type

$$\begin{aligned}\int_a^b f(x)dx = h&\left[\frac{1}{2}f(x_0) + \sum_{j=1}^{N-1} f(x_j) + \frac{1}{2}f(x_N)\right] \\ &+ \sum_{k=1}^{m} C_{2k}h^{2k}[f^{2k-1}(b) - f^{2k-1}(a)] \\ &+ (b-a)C_{2m+2}h^{2m+2}f^{(2m+2)}(\eta)\end{aligned} \tag{7.85}$$

for each positive integer m and some coefficients C_{2k}, $k = 1, \ldots m+1$. Indeed, that is the case. We will derive next this formula and find explicitly the coefficients. The resulting expression is called the Euler-Maclaurin formula. The idea behind the derivation is to use integration by parts with the aid of suitable polynomials.

Let us consider again the interval $[0,1]$ and define $B_0(x) = 1$ and $B_1(x) = x - \frac{1}{2}$. Then,

$$\begin{aligned}\int_0^1 f(x)dx = \int_0^1 f(x)B_0(x)dx &= \int_0^1 f(x)B_1'(x)dx \\ &= f(x)\ B_1(x)\big|_0^1 - \int_0^1 f'(x)B_1(x)dx \\ &= \frac{1}{2}[f(0)+f(1)] - \int_0^1 f'(x)B_1(x)dx.\end{aligned} \tag{7.86}$$

We can continue the integration by parts using the polynomials B_k, called *Bernoulli Polynomials*, which satisfy the recursion formula

$$B_{k+1}'(x) = (k+1)B_k(x), \qquad k = 1, 2, \ldots \tag{7.87}$$

Since we start with $B_1(x) = x - \frac{1}{2}$ it is clear that B_k is a polynomial of degree exactly k with leading order coefficient 1, i.e. it is monic polynomial. These polynomials are determined by the recurrence relation (7.87) up to a constant. The constant is fixed by requiring that

$$B_k(0) = B_k(1) = 0, \qquad k = 3, 5, 7, \ldots \tag{7.88}$$

Indeed,

$$B_{k+1}''(x) = (k+1)B_k'(x) = (k+1)kB_{k-1}(x) \tag{7.89}$$

and $B_{k-1}(x)$ has the form

$$B_{k-1}(x) = x^{k-1} + a_{k-2}x^{k-2} + \ldots a_1x + a_0. \tag{7.90}$$

Integrating (7.89) twice we get

$$B_{k+1}(x) = k(k+1)\left[\frac{1}{k(k+1)}x^{k+1} + \frac{a_{k-2}}{(k-1)k}x^k + \ldots + \frac{1}{2}a_0x^2 + bx + c\right]$$

For $k+1$ odd, the two constants of integration b and c are determined by the condition (7.88). The B_k for k even are then given by $B_k = B_{k+1}'/(k+1)$.

We are going to need a few properties of the Bernoulli polynomials. By construction, B_k is an even (odd) polynomial in $x - \frac{1}{2}$ if k is even (odd). Equivalently, they satisfy the identity

$$(-1)^k B_k(1-x) = B_k(x). \tag{7.91}$$

This follows because the polynomials $A_k(x) = (-1)^k B_k(1-x)$ satisfy the same conditions that define the Bernoulli polynomials, i.e. $A'_{k+1}(x) = (k+1)A_k(x)$ and $A_k(0) = A_k(1) = 0$, for $k = 3, 5, 7, \ldots$ and since $A_1(x) = B_1(x)$ they are the same. From (7.91) and (7.88) we get that

$$B_k(0) = B_k(1), \qquad k = 2, 3, , \ldots \tag{7.92}$$

We define the *Bernoulli numbers* as

$$B_k := B_k(0) = B_k(1), \text{ for } k = 2, 4, 6, \ldots. \tag{7.93}$$

This together with the recurrence relation (7.87) implies that

$$\int_0^1 B_k(x)dx = \frac{1}{k+1}\int_0^1 B'_{k+1}(x)dx = \frac{1}{k+1}\left[B_{k+1}(1) - B_{k+1}(0)\right] = 0 \tag{7.94}$$

for $k = 1, 2, \ldots$.

Lemma 7.7.1. *The polynomials* $\tilde{B}_{2m}(x) = B_{2m}(x) - B_{2m}$, $m = 1, 2, \ldots$ *do not change sign in* $[0, 1]$.

Proof. We will prove it by contradiction. Let us suppose that $\tilde{B}_{2m}(x)$ changes sign. Then, it has at least 3 zeros $[B_{2m}(0) = B_{2m}(1) = 0]$ and, by Rolle's theorem, $\tilde{B}'_{2m}(x) = B'_{2m}(x)$ has at least 2 zeros in $(0, 1)$. This implies that $B_{2m-1}(x)$ has 2 zeros in $(0, 1)$. Since $B_{2m-1}(0) = B_{2m-1}(1) = 0$, again by Rolle's theorem, $B'_{2m-1}(x)$ has 3 zeros in $(0, 1)$, which implies that $B_{2m-2}(x)$ has 3 zeros, ...,etc. Then, we conclude that $B_{2l-1}(x)$ has 2 zeros in $(0, 1)$ plus the two at the end points, $B_{2l-1}(0) = B_{2l-1}(1)$ for all $l = 1, 2, \ldots$, which is a contradiction (for $l = 1, 2$). □

Here are the first few Bernoulli polynomials

$$B_0(x) = 1, \tag{7.95}$$

$$B_1(x) = x - \frac{1}{2}, \tag{7.96}$$

$$B_2(x) = \left(x - \frac{1}{2}\right)^2 - \frac{1}{12} = x^2 - x + \frac{1}{6}, \tag{7.97}$$

$$B_3(x) = \left(x - \frac{1}{2}\right)^3 - \frac{1}{4}\left(x - \frac{1}{2}\right) = x^3 - \frac{3}{2}x^2 + \frac{1}{2}x, \tag{7.98}$$

$$B_4(x) = \left(x - \frac{1}{2}\right)^4 - \frac{1}{2}\left(x - \frac{1}{2}\right)^2 + \frac{7}{5 \cdot 48} = x^4 - 2x^3 + x^2 - \frac{1}{30}, \tag{7.99}$$

$$B_5(x) = \left(x - \frac{1}{2}\right)^5 - \frac{5}{6}\left(x - \frac{1}{2}\right)^3 + \frac{7}{48}\left(x - \frac{1}{2}\right)^3. \tag{7.100}$$

Let us retake the idea of integration by parts that we started in (7.86):

$$\int_0^1 f(x)dx = \frac{1}{2}[f(0) + f(1)] - \int_0^1 f'(x)B_1(x)dx. \tag{7.101}$$

Now,

$$\begin{aligned} -\int_0^1 f'(x)B_1(x)dx &= -\frac{1}{2}\int_0^1 f'(x)B_2'(x)dx \\ &= -\frac{1}{2}B_2[f'(1) - f'(0)] + \frac{1}{2}\int_0^1 f''(x)B_2(x)dx \end{aligned} \tag{7.102}$$

and

$$\begin{aligned} \frac{1}{2}\int_0^1 f''(x)B_2(x)dx &= \frac{1}{2 \cdot 3}\int_0^1 f''(x)B_3'(x)dx \\ &= \frac{1}{2 \cdot 3}\left[f''(x)B_3(x)\Big|_0^1 - \int_0^1 f'''(x)B_3(x)dx\right] \\ &= -\frac{1}{2 \cdot 3}\int_0^1 f'''(x)B_3(x)dx \\ &= -\frac{1}{2 \cdot 3 \cdot 4}\int_0^1 f'''(x)B_4'(x)dx \\ &= -\frac{B_4}{4!}[f'''(1) - f'''(0)] + \frac{1}{4!}\int_0^1 f^{(4)}(x)B_4(x)dx. \end{aligned} \tag{7.103}$$

Continuing this way and combining (7.101), (7.102), (7.103), etc., we arrive at the Euler-Maclaurin formula for the simple trapezoidal rule in $[0, 1]$:

Theorem 7.5.

$$\int_0^1 f(x)dx = \frac{1}{2}[f(0) + f(1)] - \sum_{k=1}^{m} \frac{B_{2k}}{(2k)!}[f^{(2k-1)}(1) - f^{(2k-1)}(0)] + R_m \tag{7.104}$$

where

$$R_m = \frac{1}{(2m+2)!} \int_0^1 f^{(2m+2)}(x)[B_{2m+2}(x) - B_{2m+2}]dx. \tag{7.105}$$

Note that using (7.94), the mean value theorem for integrals, and Lemma 7.7.1, the remainder can be written as

$$R_m = -\frac{B_{2m+2}}{(2m+2)!} f^{(2m+2)}(\eta) \tag{7.106}$$

for some $\eta \in (0, 1)$.

It is now straight forward to obtain the Euler Maclaurin formula for the composite trapezoidal rule with equally spaced points:

Theorem 7.6. *(The Euler-Maclaurin Formula)*
Let m be a positive integer and $f \in C^{(2m+2)}[a, b]$, $h = \frac{b-a}{N}$ then

$$\begin{aligned} \int_a^b f(x)dx = {} & h\left[\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{j=1}^{N-1} f(a+jh)\right] \\ & - \sum_{k=1}^{m} \frac{B_{2k}}{(2k)!} h^{2k}[f^{2k-1}(b) - f^{2k-1}(a)] \\ & - \frac{B_{2m+2}}{(2m+2)!}(b-a)h^{2m+2} f^{(2m+2)}(\eta). \qquad \eta \in (a, b) \end{aligned} \tag{7.107}$$

Remarks: The error is in even powers of h. The formula gives m corrections to the composite trapezoidal rule. For a smooth periodic function and if $b-a$ is a multiple of its period, then the error of the composite trapezoidal rule, with equally spaced points, decreases faster than any power of h as $h \to 0$.

7.8 Romberg Integration

We know from the Euler-Maclaurin formula (7.107) that the trapezoidal rule $T_h[f]$ for a smooth integrand f has an asymptotic error of the form

$$\int_a^b f(x)dx - T_h[f] = c_2h^2 + c_4h^4 + \cdots \tag{7.108}$$

for some constants c_2, c_4, etc., and $h = (b-a)/N$. We are now going to apply successively Richardson's Extrapolation to the trapezoidal rule to exploit that the error only contains even power of h so that after a few extrapolations we end up with a highly accurate approximation. For short, we will write the composite trapezoidal rule as

$$T_h[f] := h\sum_{j=0}^{N}{}'' f(a+jh), \tag{7.109}$$

where $\sum''$ means that the first and the last terms have a $\frac{1}{2}$ factor.

We can perform one extrapolation to obtain a quadrature with a leading order error $O(h^4)$. If we have computed $T_{2h}[f]$ and $T_h[f]$ we combine them so as to eliminate the leading term in the error by noting that

$$\int_a^b f(x)dx = T_{2h}[f] + c_2(2h)^2 + c_4(2h)^4 + \cdots \tag{7.110}$$

so that

$$\int_a^b f(x)dx = \frac{4T_h[f] - T_{2h}[f]}{3} + \tilde{c}_4h^4 + \tilde{c}_6h^6 + \cdots, \tag{7.111}$$

for some constants $\tilde{c}_4$, $\tilde{c}_4$, etc. We can continue the Richardson's extrapolation process but it is more efficient to reuse the work we have done to compute $T_{2h}[f]$ to evaluate $T_h[f]$. To this end, we note that

$$\begin{aligned} T_h[f] - \frac{1}{2}T_{2h}[f] &= h\sum_{j=0}^{N}{}'' f(a+jh) - h\sum_{j=0}^{\frac{N}{2}}{}'' f(a+2jh) \\ &= h\sum_{j=1}^{\frac{N}{2}} f(a+(2j-1)h). \end{aligned} \tag{7.112}$$

Then, setting $h_k = (b-a)/2^k$ for any nonnegative integer k, we have

$$T_{h_k}[f] = \frac{1}{2}T_{h_{k-1}}[f] + h_k \sum_{j=1}^{2^{k-1}} f(a+(2j-1)h_k). \tag{7.113}$$

Beginning with the simple trapezoidal rule (two points):

$$T(0,0) := T_{h_0}[f] = \frac{b-a}{2}\left[f(a)+f(b)\right] \tag{7.114}$$

we can successively double the number of points in the quadrature by using (7.113). For $k = 1, 2, ..., M$

$$T(k,0) := \frac{1}{2}T(k-1,0) + h_k \sum_{j=1}^{2^{k-1}} f(a+2j-1)h_k) \tag{7.115}$$

and immediately extrapolate as follows. From $T(0,0)$ and $T(1,0)$ we extrapolate to obtain

$$T(1,1) := T(1,0) + \frac{1}{4-1}[T(1,0) - T(0,0)], \tag{7.116}$$

From $T(1,0)$ and $T(2,0)$ we get

$$T(2,1) := T(2,0) + \frac{1}{4-1}[T(2,0) - T(1,0)], \tag{7.117}$$

from $T(1,1)$ and $T(2,1)$ we compute $T(2,2)$, etc. For example, for $M = 4$ we generate a table of approximations like the following one:

$$\begin{array}{lllll}
T(0,0) & & & & \\
T(1,0) & T(1,1) & & & \\
T(2,0) & T(2,1) & T(2,2) & & \\
T(3,0) & T(3,1) & T(3,2) & T(3,3) & \\
T(4,0) & T(4,1) & T(4,2) & T(4,3) & T(4,4)
\end{array} \tag{7.118}$$

We can proceed row by row, where each of the $T(k,m)$ for $m \geq 1$ is obtained by extrapolation

$$T(k,m) = T(k,m-1) + \frac{1}{4^m-1}[T(k,m-1) - T(k-1,m-1)]. \tag{7.119}$$

This is Romberg's method and is listed in pseudo-code in Algorithm 7.1. $T(M,M)$ contains the most accurate approximation (neglecting round-off errors) to the integral.

Algorithm 7.1 Romberg Integration

$h \leftarrow b - a$;
$T(0,0) \leftarrow \frac{1}{2}(b-a)[f(a)+f(b)]$;
for $k = 1, \ldots, M$ **do**
  $h \leftarrow h/2$;
  $$T(k,0) \leftarrow \tfrac{1}{2}T(k-1,0) + h\sum_{j=1}^{2^{k-1}} f(a+(2j-1)h);$$
  for $m = 1, \ldots, k$ **do**
    $$T(k,m) \leftarrow T(k,m-1) + \frac{1}{4^m - 1}[T(k,m-1) - T(k-1,m-1)];$$
  end for
end for

Example 7.4. *We use Romberg's method to approximate the integral of* $f(x) = 3x^2 e^{x^3}/(e-1)$ *in* $[0,1]$*, whose value is 1. Table 7.1 shows the table of approximations corresponding to* (7.118)*. With* $M = 4$*, an accuracy of about 6 digits is obtained.*

Table 7.1: Romberg integration for $f(x) = 3x^2 e^{x^3}/(e-1)$ in $[0,1]$. M=4.

2.37296506				
1.43378228	1.12072136			
1.11897636	1.01404106	1.00692904		
1.03059109	1.00112933	1.00026855	1.00016283	
1.00770499	1.00007629	1.00000609	1.00000193	1.00000129

7.9 Bibliographic Notes

Section 7.1 . We limited the discussion to the two most used basic quadratures: the trapezoidal rule and Simpson's rule. They are two examples of Newton-Cotes formulas, which are the interpolatory quadratures with equispaced nodes and for $w(x) \equiv 1$. Krylov [Kry12] discusses Newton-Cotes formulas extensively and presents a table for the weights for $n = 1, \ldots, 10$. The use of the divided differences identity (7.10) for the error, in the case when the integral of $(x-x_0)\ldots(x-x_n)$ vanishes, appears in the book by

Conte and de Boor (Section 5.2) [CdB72]. General references for numerical integration are the classical texts by Krylov [Kry12] and by Davis and Rabinowitz [DR84]. The latter has also a chapter (Chap. 8) on integration in two or more dimensions.

Section 7.2 . Interpolatory quadratures are discussed in more detail in [Kry12, DR84]. For example, a general formula for the error can be derived using the interpolation (Cauchy's) remainder [Kry12].

Section 7.3 . Gauss derived the quadrature formula for $w(x) = 1$ using continued fractions [Gau16]. Gautschi [Gau81] provides an excellent historical account of the Gaussian quadrature. For the properties of this quadrature and its convergence for a bounded closed interval we followed Gautschi's text (Section 3.2.3) [Gau11]. A discussion of convergence when the interval is infinite (if $|f(x)| \leq A + Bx^{2m}$ for some $m \in \mathbb{N}$) is presented in Freud's book (Section III.1) [Fre71]. The Radau quadrature and the Lobatto quadrature, which are the special cases when one or both of the end points of the interval of integration are nodes, respectively, are discussed in Gautschi's text [Gau11] and in Hildebrand's book [Hil13] (Chapter 8), where the particular cases of the Legendre, Laguerre, Hermite, Jacobi, and Chebyshev quadratures are all presented. The method based on the eigenvalue problem to obtain the Gaussian nodes and weights is due to Golub and Welsch [GW69]. Glasier, Liu, and Rohklin [GLR07] proposed a fast algorithm to compute all the nodes and weights in $O(n)$ operations using Newton's root finding and Taylor series approximations. Hale and Townsend [HT13] designed an efficient, alternative method to compute Gaussian weights and nodes based also on Newton's root-finding method but with initial guesses obtained via asymptotic formulas. Their method allows for the computation of the n-point Gaussian quadrature in $O(n)$ operations to an accuracy close to double double precision for any $n \geq 100$.

Section 7.4 . Clenshaw and Curtis proposed their quadrature, which they called "Chebyshev formula", in 1960 [CC60]. Glentleman [Gen72], 12 years later, made the connection with the DCT for a fast computation of the quadrature. Trefethen [Tre08] has presented a compelling study that shows the Clenshaw-Curtis formula is a clear competitor of the Gaussian quadrature because in most cases the two quadratures achieve comparable accuracy for the same number of nodes.

Section 7.5 . The book by Davis and Rabinowitz [DR84] covers in detail composite quadratures, therein called "compound quadratures". Simpson's rule has a long history, going back to B. Cavalieri in 1639, who had found the formula in geometric form [Gol77]. It is named after T. Simpson who rediscovered the quadrature rule in 1743 [Sim43](pp. 109-110).

Section 7.83 . The derivation using Hermite interpolation follows that in [SB02].

Section 7.7 . The formula was obtained independently by Euler and Maclaurin. For a historical account see [Gol77], Section 2.6. The derivation using Bernoulli polynomials follows that in [SB02].

Section 7.8 . Romberg proposed his efficient, repeated extrapolation method for the trapezoidal rule in 1955 [Rom55]. A historical account has been provided by Brezinski [Bre10]

Chapter 8

Linear Algebra

In this chapter we review some important concepts of linear algebra in preparation for the presentation of numerical methods for linear systems of equations and eigenvalue problems.

8.1 Numerical Linear Algebra

There are two main problems in numerical linear algebra: solving large linear systems of equations and finding eigenvalues and eigenvectors. Related to the latter, there is also the problem of computing the singular value decomposition (SVD) of a large matrix. We describe these problems next.

8.1.1 Linear Systems

Linear systems of equations appear in a wide variety of applications and are an indispensable tool in scientific computing. Given a nonsingular, $n \times n$ matrix A and a vector $b \in \mathbb{R}^n$, where n could be a very large positive integer, we would like to find the unique solution x, satisfying

$$Ax = b \tag{8.1}$$

or an accurate approximation $\tilde{x}$ of x. Henceforth, we will assume, unless otherwise stated, that the matrix A is real.

We will study *direct methods* (for example Gaussian elimination), which compute the solution (up to roundoff errors) in a finite number of steps and *iterative methods*, which starting from an initial approximation $x^{(0)}$ of the solution produce subsequent approximations $x^{(1)}, x^{(2)}, \ldots$ from a given recipe

$$x^{(k+1)} = G(x^{(k)}, A, b), \quad k = 0, 1, \ldots \tag{8.2}$$

where G is a continuous function of the first variable. Consequently, if the iterations converge as $k \to \infty$ to the solution x of the linear system $Ax = b$, then

$$x = G(x, A, b). \tag{8.3}$$

That is, x is a *fixed point* of G.

One of the main strategies in the design of efficient numerical methods for linear systems is to transform the problem to one which is much easier to solve. Both direct and iterative methods use this strategy.

8.1.2 Eigenvalue Problems

The eigenvalue problem for an $n \times n$ matrix A consists of finding each or some of the scalars (eigenvalues) λ and the corresponding eigenvectors $v \neq 0$ such that

$$Av = \lambda v. \tag{8.4}$$

Equivalently, $(A - \lambda I)v = 0$ and so the eigenvalues are the roots of the characteristic polynomial of A

$$p(\lambda) = \det(A - \lambda I). \tag{8.5}$$

Clearly, we cannot solve this problem with a finite number of elementary operations (for $n \geq 5$ it would be a contradiction to Abel's theorem) so iterative methods have to be employed. Also, λ and v could be complex even if A is real.

The maximum of the absolute value (modulus) of the eigenvalues of a matrix is a useful concept in numerical linear algebra.

Definition 8.1. *Let A be an $n \times n$ matrix. The spectral radius ρ of A is defined as*

$$\rho(A) = \max\{|\lambda_1|, \ldots, |\lambda_n|\}, \tag{8.6}$$

where λ_i, $i = 1, \ldots, n$ are the eigenvalues (not necessarily distinct) of A.

Large eigenvalue-eigenvector problems arise for example in the study of steady state behavior of time-discrete Markov processes which are often used in a wide range of applications, such as finance, population dynamics, and data mining. The problem is to find an eigenvector v associated with the eigenvalue 1, i.e. $v = Av$. Such v is a probability vector so all its entries are positive, add up to 1, and represent the probabilities of the system (described by the Markov process) to be in a given state, in the limit as time goes to infinity. This eigenvector v is in effect a *fixed point* of the linear transformation represented by the Markov matrix A.

8.1.3 Singular Value Decomposition

The singular value decomposition (SVD) of a matrix is related to the eigenvalue problem and finds applications in image compression, model reduction techniques, data analysis, and many other fields. Given an $m \times n$ matrix A, the idea is to consider the eigenvalues and eigenvectors of the square, $n \times n$ matrix $A^T A$, where A^T is the transpose of A (or $A^* A$, where A^* is the conjugate transpose of A as defined below, if A is complex). As we will see, the eigenvalues are all real and nonnegative and $A^T A$ has a complete set of orthogonal eigenvectors. The *singular values* of a matrix A are the positive square roots of the eigenvalues of $A^T A$. Using this, it follows that any real $m \times n$ matrix A has the *singular value decomposition* (SVD)

$$U^T A V = \Sigma, \tag{8.7}$$

where U is an orthogonal $m \times m$ matrix (i.e. $U^T U = I$), V is an orthogonal $n \times n$ matrix, and Σ is a "diagonal" matrix of the form

$$\Sigma = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}, \qquad D = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r), \tag{8.8}$$

where $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_r > 0$ are the nonzero singular values of A. Here, $\mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ stands for the diagonal matrix with entries $\sigma_1, \sigma_2, \ldots, \sigma_r$ on its diagonal.

8.2 Notation

A matrix A with elements a_{ij} will be denoted $A = (a_{ij})$, this could be a square $n \times n$ matrix or an $m \times n$ matrix. A^T denotes the transpose of A, i.e. $A^T = (a_{ji})$.

A vector in $x \in \mathbb{R}^n$ will be represented as the n-tuple

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}. \tag{8.9}$$

The canonical vectors, corresponding to the standard basis in $\mathbb{R}^n$, will be denoted by $e_1, e_2, \dots, e_n$, where e_k is the n-vector with all entries equal to zero except the k-th one, which is equal to one.

The inner product of two real vectors x and y in $\mathbb{R}^n$ is

$$\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i = x^T y. \tag{8.10}$$

If the vectors are complex, i.e. x and y in $\mathbb{C}^n$ we define their inner product as

$$\langle x, y \rangle = \sum_{i=1}^{n} \bar{x}_i y_i, \tag{8.11}$$

where $\bar{x}_i$ denotes the complex conjugate of x_i.

With the inner product (8.10) in the real case or (8.11) in the complex case, we can define the Euclidean norm

$$\|x\|_2 = \sqrt{\langle x, x \rangle}. \tag{8.12}$$

Note that if A is an $n \times n$ real matrix and $x, y \in \mathbb{R}^n$ then

$$\begin{aligned} \langle x, Ay \rangle &= \sum_{i=1}^{n} x_i \left(\sum_{k=1}^{n} a_{ik} y_k \right) = \sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik} x_i y_k \\ &= \sum_{k=1}^{n} \left(\sum_{i=1}^{n} a_{ik} x_i \right) y_k = \sum_{k=1}^{n} \left(\sum_{i=1}^{n} a_{ki}^T x_i \right) y_k, \end{aligned} \tag{8.13}$$

that is

$$\langle x, Ay \rangle = \langle A^T x, y \rangle. \tag{8.14}$$

Similarly in the complex case we have

$$\langle x, Ay \rangle = \langle A^* x, y \rangle, \tag{8.15}$$

where A^* is the conjugate transpose of A, i.e. $A^* = (\overline{a_{ji}})$.

8.3 Some Important Types of Matrices

One useful type of linear transformations consists of those that preserve the Euclidean norm. That is, if $y = Ax$, then $\|y\|_2 = \|x\|_2$ but this implies

$$\langle Ax, Ax\rangle = \langle A^T Ax, x\rangle = \langle x, x\rangle \tag{8.16}$$

and consequently $A^T A = I$.

Definition 8.2. *An $n \times n$ real (complex) matrix A is called orthogonal (unitary) if $A^T A = I$ ($A^* A = I$).*

Two of the most important types of matrices in applications are symmetric (Hermitian) and positive definite matrices.

Definition 8.3. *An $n \times n$ real matrix A is called symmetric if $A^T = A$. If the matrix A is complex it is called Hermitian if $A^* = A$.*

Symmetric (Hermitian) matrices have real eigenvalues, for if v is an eigenvector associated to an eigenvalue λ of A, we can assumed it has been normalized so that $\langle v, v\rangle = 1$, and

$$\langle v, Av\rangle = \langle v, \lambda v\rangle = \lambda\langle v, v\rangle = \lambda. \tag{8.17}$$

But if $A^T = A$ then

$$\lambda = \langle v, Av\rangle = \langle Av, v\rangle = \langle \lambda v, v\rangle = \overline{\lambda}\langle v, v\rangle = \overline{\lambda}, \tag{8.18}$$

and $\lambda = \overline{\lambda}$ if and only if $\lambda \in \mathbb{R}$.

Definition 8.4. *An $n\times n$ matrix A is called positive definite if it is symmetric (Hermitian) and $\langle x, Ax\rangle > 0$ for all $x \in \mathbb{R}^n, x \neq 0$.*

By the preceding argument the eigenvalues of a positive definite matrix A are real because $A^T = A$. Moreover, if $Av = \lambda v$ with $\|v\|_2 = 1$ then $0 < \langle v, Av\rangle = \lambda$. Therefore, positive definite matrices have real, positive eigenvalues. Conversely, if all the eigenvalues of a symmetric matrix A are positive, then A is positive definite. This follows from the fact that symmetric matrices are diagonalizable by an orthogonal matrix S, i.e. $A = SDS^T$, where D is a diagonal matrix with the eigenvalues $\lambda_1, \ldots, \lambda_n$ (not necessarily distinct) of A. Then

$$\langle x, Ax \rangle = \sum_{i=1}^{n} \lambda_i y_i^2, \tag{8.19}$$

where $y = S^T x$. Thus *a symmetric (Hermitian) matrix A is positive definite if and only if all its eigenvalues are positive.* Moreover, since the determinant is the product of the eigenvalues, positive definite matrices have a positive determinant.

We now review another useful consequence of positive definiteness.

Definition 8.5. *Let $A = (a_{ij})$ be an $n \times n$ matrix. Its leading principal submatrices are the square matrices*

$$A_k = \begin{bmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & & \\ a_{k1} & \cdots & a_{kk} \end{bmatrix}, \quad k = 1, \ldots, n. \tag{8.20}$$

Theorem 8.1. *All the leading principal submatrices of a positive definite matrix are positive definite.*

Proof. Suppose A is an $n \times n$ positive definite matrix. Then, all its leading principal submatrices are symmetric (Hermitian). Moreover, if we take a vector $x \in \mathbb{R}^n$ of the form

$$x = \begin{bmatrix} y_1 \\ \vdots \\ y_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{8.21}$$

where $y = [y_1, \ldots, y_k]^T \in \mathbb{R}^k$ is an arbitrary nonzero vector then

$$0 < \langle x, Ax \rangle = \langle y, A_k y \rangle,$$

which shows that A_k for $k = 1, \ldots, n$ is positive definite. $\square$

The converse of Theorem 8.1 is also true but the proof is much more technical: *A is positive definite if and only if* $\det(A_k) > 0$ *for* $k = 1, \ldots, n$.

Note also that if A is positive definite then all its diagonal elements are positive because $0 < \langle e_j, Ae_j \rangle = a_{jj}$, for $j = 1, \ldots, n$.

8.4 Schur Theorem

Theorem 8.2. *(Schur) Let A be an $n \times n$ matrix, then there exists a unitary matrix T ($T^*T = I$) such that*

$$T^*AT = \begin{bmatrix} \lambda_1 & b_{12} & b_{13} & \cdots & b_{1n} \\ & \lambda_2 & b_{23} & \cdots & b_{2n} \\ & & \ddots & & \vdots \\ & & & & b_{n-1,n} \\ & & & & \lambda_n \end{bmatrix}, \tag{8.22}$$

where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A and all the elements below the diagonal are zero.

Proof. We will do a proof by induction. Let A be a 2×2 matrix with eigenvalues λ_1 and λ_2. Let u be a normalized, eigenvector u ($u^*u = 1$) corresponding to λ_1. Then we can take T as the matrix whose first column is u and its second column is a unit vector v orthogonal to u ($u^*v = 0$). We have

$$T^*AT = \begin{bmatrix} u^* \\ v^* \end{bmatrix} \begin{bmatrix} \lambda_1 u & Av \end{bmatrix} = \begin{bmatrix} \lambda_1 & u^*Av \\ 0 & v^*Av \end{bmatrix}. \tag{8.23}$$

The scalar v^*Av has to be equal to λ_2, as similar matrices have the same eigenvalues. We now assume the result is true for all $k \times k$ ($k \geq 2$) matrices and will show that it is also true for all $(k+1) \times (k+1)$ matrices. Let A be a $(k+1) \times (k+1)$ matrix and let u_1 be a normalized eigenvector associated with eigenvalue λ_1. Choose k unit vectors $t_1, \ldots, t_k$ so that the matrix $T_1 = [u_1\, t_1 \ldots t_k]$ is unitary. Then,

$$T_1^*AT_1 = \begin{bmatrix} \lambda_1 & c_{12} & c_{13} & \cdots & c_{1,k+1} \\ 0 & & & & \\ \vdots & & A_k & & \\ & & & & \\ 0 & & & & \end{bmatrix}, \tag{8.24}$$

where A_k is a $k \times k$ matrix. Now, the eigenvalues of the matrix on the right hand side of (8.24) are the roots of $(\lambda_1 - \lambda)\det(A_k - \lambda I)$ and since this matrix is similar to A, it follows that the eigenvalues of A_k are the remaining eigenvalues of A, $\lambda_2, \ldots, \lambda_{k+1}$. By the induction hypothesis there is a unitary matrix T_k such that $T_k^* A_k T_k$ is upper triangular with the eigenvalues $\lambda_2, \ldots, \lambda_{k+1}$ sitting on the diagonal. We can now use T_k to construct the $(k+1) \times (k+1)$ unitary matrix as

$$T_{k+1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & & & & \\ \vdots & & T_k & & \\ & & & & \\ 0 & & & & \end{bmatrix} \tag{8.25}$$

and define $T = T_1 T_{k+1}$. Then

$$T^* A T = T_{k+1}^* T_1^* A T_1 T_{k+1} = T_{k+1}^* (T_1^* A T_1) T_{k+1} \tag{8.26}$$

and using (8.24) and (8.25) we get

$$\begin{aligned} T^* A T &= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & & & & \\ \vdots & & T_k^* & & \\ & & & & \\ 0 & & & & \end{bmatrix} \begin{bmatrix} \lambda_1 & c_{12} & c_{13} & \cdots & c_{1,k+1} \\ 0 & & & & \\ \vdots & & A_k & & \\ & & & & \\ 0 & & & & \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & & & & \\ \vdots & & T_k & & \\ & & & & \\ 0 & & & & \end{bmatrix} \\ &= \begin{bmatrix} \lambda_1 & b_{12} & b_{13} & \cdots & b_{1,k+1} \\ & \lambda_2 & b_{23} & \cdots & b_{2,k+1} \\ & & \ddots & & \vdots \\ & & & & b_{k,k+1} \\ & & & & \lambda_{k+1} \end{bmatrix}. \end{aligned}$$

□

8.5 QR Factorization

Consider an $m \times n$ matrix A with columns $a_1, \ldots, a_n$ and suppose these form a linearly independent set, i.e. A is full rank. If we employ the Gram-Schmidt procedure to orthonormalize $\{a_1, \ldots, a_n\}$ we get (cf. Section 4.1.2.1) the orthonormal set $\{q_1, \ldots, q_n\}$ given by

$$
\begin{aligned}
b_1 &= a_1, \\
r_{11} &= \|b_1\|, \\
q_1 &= a_1/r_{11}, \\
\text{For } & k = 2, \ldots, n \\
b_k &= a_k - \sum_{j=1}^{k-1} r_{jk} q_j, \quad r_{jk} = \langle q_j, a_k \rangle, \\
r_{kk} &= \|b_k\|, \\
q_k &= b_k/r_{kk}.
\end{aligned}
\tag{8.27}
$$

Note that (8.27) implies that a_k is a linear combination of $q_1, \ldots, q_k$ and since $b_k = r_{kk} q_k$ we have

$$
a_k = \sum_{j=1}^{k} r_{jk} q_j, \quad k = 1, \ldots n \tag{8.28}
$$

or in matrix form

$$
A = \tilde{Q}\tilde{R}, \quad \tilde{Q} = \begin{bmatrix} \\ q_1 & \ldots & q_n \\ \\ \end{bmatrix}, \quad \tilde{R} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & \cdots & r_{2n} \\ & & \ddots & \vdots \\ & & & r_{nn} \end{bmatrix}. \tag{8.29}
$$

The $m \times n$ matrix $\tilde{Q}$ has columns $q_1, \ldots, q_n$ that are orthonormal. This is called a *reduced* QR factorization of A. A *full* QR factorization of A, where Q is an $m \times m$ orthogonal matrix and R is an $m \times n$ upper triangular matrix (in the sense shown below), can be obtained by appending $\tilde{Q}$ with $m - n$ orthonormal columns to complete an orthonormal basis of $\mathbb{R}^m$ and a corresponding block of $m - n$ rows of zeros to $\tilde{R}$ as follows:

$$
A = QR, \quad Q = \begin{bmatrix} & * & \cdots & * \\ & \vdots & \cdots & \vdots \\ \tilde{Q} & & & \\ & \vdots & & \vdots \\ & * & \cdots & * \end{bmatrix}, \quad R = \begin{bmatrix} & \tilde{R} & \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix}, \tag{8.30}
$$

where the $m \times (m-n)$ block marked with $*$'s represents the added columns so that $Q^TQ = QQ^T = I$. Note that orthonormality is defined up to a sign. Since we are taking $r_{kk} = \|b_k\|$ it follows that there is a unique QR factorization of the full rank matrix A such that $r_{kk} > 0$, for all $k = 1, \ldots, n$.

The Gram-Schmidt procedure is not numerically stable; round-off error can destroy orthogonality when there are columns almost linearly dependent. We will see in Section 11.2 a stable method to obtain QR by using a sequence of hyperplane reflections.

8.6 Matrix Norms

We reviewed the concept of a norm on a vector space in Section 2.1. In numerical linear algebra, we need to work with norms defined on matrices. Let A be an $n \times n$ matrix. We can view A as a vector in $\mathbb{R}^{n\times n}$ and define its corresponding Euclidean norm

$$\|A\| = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} |a_{ij}|^2}. \tag{8.31}$$

This is called the Frobenius norm for matrices. A different matrix norm can be obtained by using a given vector norm and matrix-vector multiplication. Given a vector norm $\|\cdot\|$ in $\mathbb{R}^n$ (or in $\mathbb{C}^n$), it is easy to show that

$$\|A\| = \max_{x\neq 0} \frac{\|Ax\|}{\|x\|}, \tag{8.32}$$

satisfies the properties $(i), (ii), (iii)$ of a norm for all $n \times n$ matrices A . That is, the vector norm induces a matrix norm.

Definition 8.6. *The matrix norm defined by (8.32) is called the subordinate or natural norm induced by the vector norm* $\|\cdot\|$.

Example 8.1.

$$\|A\|_1 = \max_{x\neq 0} \frac{\|Ax\|_1}{\|x\|_1}, \tag{8.33}$$

$$\|A\|_\infty = \max_{x\neq 0} \frac{\|Ax\|_\infty}{\|x\|_\infty}, \tag{8.34}$$

$$\|A\|_2 = \max_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2}. \tag{8.35}$$

Theorem 8.3. *Let $\|\cdot\|$ be an induced matrix norm. Then,*

(a) $\|Ax\| \leq \|A\|\|x\|$,

(b) $\|AB\| \leq \|A\|\|B\|$.

Proof. (a) if $x = 0$ the result holds trivially. Take $x \neq 0$, then the definition (8.32) implies

$$\frac{\|Ax\|}{\|x\|} \leq \|A\| \tag{8.36}$$

that is $\|Ax\| \leq \|A\|\|x\|$.
(b) Take $x \neq 0$. By (a) $\|ABx\| \leq \|A\|\|Bx\| \leq \|A\|\|B\|\|x\|$ and thus

$$\frac{\|ABx\|}{\|x\|} \leq \|A\|\|B\|. \tag{8.37}$$

Taking the max we get that $\|AB\| \leq \|A\|\|B\|$. □

The following theorem offers a more concrete way to compute the matrix norms (8.33)-(8.35).

Theorem 8.4. *Let $A = (a_{ij})$ be an $n \times n$ matrix then*

(a) $\|A\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}|$.

(b) $\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|$.

(c) $\|A\|_2 = \sqrt{\rho(A^TA)}$,

where $\rho(A^TA)$ is the spectral radius of A^TA, as defined in (8.6).

Proof. (a)

$$\|Ax\|_1 = \sum_{i=1}^{n} \left| \sum_{j=1}^{n} a_{ij}x_j \right| \leq \sum_{j=1}^{n} |x_j| \left(\sum_{i=1}^{n} |a_{ij}| \right) \leq \left(\max_j \sum_{i=1}^{n} |a_{ij}| \right) \|x\|_1.$$

Thus, $\|A\|_1 \leq \max_j \sum_{i=1}^{n} |a_{ij}|$. We just need to show there is a vector x for which the equality holds. Let j^* be the index such that

$$\sum_{i=1}^{n} |a_{ij^*}| = \max_j \sum_{i=1}^{n} |a_{ij}| \tag{8.38}$$

and take x to be given by $x_i = 0$ for $i \neq j^*$ and $x_{j^*} = 1$. Then, $\|x\|_1 = 1$ and

$$\|Ax\|_1 = \sum_{i=1}^{n} \left| \sum_{j=1}^{n} a_{ij} x_j \right| = \sum_{i=1}^{n} |a_{ij^*}| = \max_j \sum_{i=1}^{n} |a_{ij}|. \tag{8.39}$$

(b) Analogously to (a) we have

$$\|Ax\|_\infty = \max_i \left| \sum_{j=1}^{n} a_{ij} x_j \right| \leq \left(\max_i \sum_{j=1}^{n} |a_{ij}| \right) \|x\|_\infty. \tag{8.40}$$

Let i^* be the index such that

$$\sum_{j=1}^{n} |a_{i^*j}| = \max_i \sum_{j=1}^{n} |a_{ij}| \tag{8.41}$$

and take x given by

$$x_j = \begin{cases} \frac{a_{i^*j}}{|a_{i^*j}|} & \text{if } a_{i^*j} \neq 0, \\ 1 & \text{if } a_{i^*j} = 0. \end{cases} \tag{8.42}$$

Then, $|x_j| = 1$ for all j and $\|x\|_\infty = 1$. Hence

$$\|Ax\|_\infty = \max_i \left| \sum_{j=1}^{n} a_{ij} x_j \right| = \sum_{j=1}^{n} |a_{i^*j}| = \max_i \sum_{i=1}^{n} |a_{ij}|. \tag{8.43}$$

(c) By definition

$$\|A\|_2^2 = \max_{x \neq 0} \frac{\|Ax\|_2^2}{\|x\|_2^2} = \max_{x \neq 0} \frac{x^T A^T A x}{x^T x} \tag{8.44}$$

Note that the matrix $A^T A$ is symmetric and all its eigenvalues are nonnegative. Let us label them in increasing order, $0 \leq \lambda_1 \leq \lambda_1 \leq \cdots \leq \lambda_n$. Then,

$\lambda_n = \rho(A^T A)$. Now, since $A^T A$ is symmetric, there is an orthogonal matrix Q such that $Q^T A^T A Q = D = diag(\lambda_1, \ldots, \lambda_n)$. Therefore, changing variables, $x = Qy$, we have

$$\frac{x^T A^T A x}{x^T x} = \frac{y^T D y}{y^T y} = \frac{\lambda_1 y_1^2 + \cdots + \lambda_n y_n^2}{y_1^2 + \cdots + y_n^2} \leq \lambda_n. \tag{8.45}$$

Now take the vector y such that $y_j = 0$ for $j \neq n$ and $y_n = 1$ and the equality holds. Thus,

$$\|A\|_2 = \sqrt{\max_{x \neq 0} \frac{\|Ax\|_2^2}{\|x\|_2^2}} = \sqrt{\lambda_n} = \sqrt{\rho(A^T A)}. \tag{8.46}$$

□

Note that if $A^T = A$ then

$$\|A\|_2 = \sqrt{\rho(A^T A)} = \sqrt{\rho(A^2)} = \rho(A). \tag{8.47}$$

Let λ be an eigenvalue of the matrix A with eigenvector x, normalized so that $\|x\| = 1$. Then,

$$|\lambda| = |\lambda|\|x\| = \|\lambda x\| = \|Ax\| \leq \|A\|\|x\| = \|A\| \tag{8.48}$$

for any matrix norm with the property $\|Ax\| \leq \|A\|\|x\|$. Thus,

$$\rho(A) \leq \|A\| \tag{8.49}$$

for any induced norm. However, given an $n \times n$ matrix A and $\epsilon > 0$ there is at least one induced matrix norm such that $\|A\|$ is within ϵ of the spectral radius of A.

Theorem 8.5. *Let A be an $n \times n$ matrix. Given $\epsilon > 0$, there is at least one induced matrix norm $\|\cdot\|$ such that*

$$\rho(A) \leq \|A\| \leq \rho(A) + \epsilon. \tag{8.50}$$

Proof. By Schur's Theorem, there is a unitary matrix T such that

$$T^* A T = \begin{bmatrix} \lambda_1 & b_{12} & b_{13} & \cdots & b_{1n} \\ & \lambda_2 & b_{23} & \cdots & b_{2n} \\ & & \ddots & & \vdots \\ & & & & b_{n-1,n} \\ & & & & \lambda_n \end{bmatrix} = U, \tag{8.51}$$

where λ_j, $j = 1, \ldots, n$ are the eigenvalues of A. Take $0 < \delta < 1$ and define the diagonal matrix $D_\delta = diag(\delta, \delta^2, \ldots, \delta^n)$. Then

$$D_\delta^{-1} U D_\delta = \begin{bmatrix} \lambda_1 & \delta b_{12} & \delta^2 b_{13} & \cdots & \delta^{n-1} b_{1n} \\ & \lambda_2 & \delta b_{23} & \cdots & \delta^{n-2} b_{2n} \\ & & \ddots & & \vdots \\ & & & & \delta b_{n-1,n} \\ & & & & \lambda_n \end{bmatrix}. \tag{8.52}$$

Given $\epsilon > 0$, we can find δ sufficiently small so that $D_\delta^{-1} U D_\delta$ is "within ϵ" of a diagonal matrix, in the sense that the sum of the absolute values of the off diagonal entries is less than ϵ for each row:

$$\sum_{j=i+1}^{n} \left|\delta^{j-i} b_{ij}\right| \le \epsilon \qquad \text{for } i = 1, \ldots, n. \tag{8.53}$$

Now,

$$D_\delta^{-1} U D_\delta = D_\delta^{-1} T^* A T D_\delta = (T D_\delta)^{-1} A (T D_\delta) \tag{8.54}$$

Given a nonsingular matrix S and a matrix norm $\|\cdot\|$ then

$$\|A\|' = \|S^{-1} A S\| \tag{8.55}$$

is also a norm. Taking $S = T D_\delta$ and using the infinity norm we get

$$\begin{aligned} \|A\|' &= \|(T D_\delta)^{-1} A (T D_\delta)\|_\infty \\ &\le \left\| \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \right\|_\infty + \left\| \begin{bmatrix} 0 & \delta b_{12} & \delta^2 b_{13} & \cdots & \delta^{n-1} b_{1n} \\ & 0 & \delta b_{23} & \cdots & \delta^{n-2} b_{2n} \\ & & \ddots & & \vdots \\ & & & & \delta b_{n-1,n} \\ & & & & 0 \end{bmatrix} \right\|_\infty \\ &\le \rho(A) + \epsilon. \end{aligned}$$

$\square$

8.7 Condition Number of a Matrix

Consider the 5×5 Hilbert matrix

$$H_5 = \begin{bmatrix} \frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} \end{bmatrix} \tag{8.56}$$

and the linear system $H_5 x = b$ where

$$b = \begin{bmatrix} 137/60 \\ 87/60 \\ 153/140 \\ 743/840 \\ 1879/2520 \end{bmatrix}. \tag{8.57}$$

The exact solution of this linear system is $x = [1, 1, 1, 1, 1]^T$. Note that $b \approx [2.28, 1.45, 1.09, 0.88, 0.74]^T$. Let us perturb b slightly (about % 1)

$$b + \delta b = \begin{bmatrix} 2.28 \\ 1.46 \\ 1.10 \\ 0.89 \\ 0.75 \end{bmatrix} \tag{8.58}$$

The solution of the perturbed system (up to rounding at 12 digits of accuracy) is

$$x + \delta x = \begin{bmatrix} 0.5 \\ 7.2 \\ -21.0 \\ 30.8 \\ -12.6 \end{bmatrix}. \tag{8.59}$$

A relative perturbation of $\|\delta b\|_2/\|b\|_2 = 0.0046$ in the data produces a change in the solution equal to $\|\delta x\|_2 \approx 40$. The perturbations gets amplified nearly four orders of magnitude!

This high sensitivity of the solution to small perturbations is inherent to the matrix of the linear system, H_5 in this example.

Consider the linear system $Ax = b$ and the perturbed one $A(x + \delta x) = b + \delta b$. Then, $Ax + A\delta x = b + \delta b$ implies $\delta x = A^{-1}\delta b$ and so

$$\|\delta x\| \leq \|A^{-1}\|\|\delta b\| \tag{8.60}$$

for any induced norm. But also $\|b\| = \|Ax\| \leq \|A\|\|x\|$ or

$$\frac{1}{\|x\|} \leq \|A\|\frac{1}{\|b\|}. \tag{8.61}$$

Combining (8.60) and (8.61) we obtain

$$\frac{\|\delta x\|}{\|x\|} \leq \|A\|\|A^{-1}\|\frac{\|\delta b\|}{\|b\|}. \tag{8.62}$$

The right hand side of this inequality is actually a least upper bound, there are b and δb for which the equality holds.

Definition 8.7. *Given a matrix norm $\|\cdot\|$, the* **condition number** *of a matrix A, denoted by $\kappa(A)$ is defined by*

$$\kappa(A) = \|A\|\|A^{-1}\|. \tag{8.63}$$

Example 8.2. *The condition number of the 5×5 Hilbert matrix H_5, (8.56), in the 2 norm is approximately 4.7661×10^5. For the particular b and δb we chose we actually got a variation in the solution of $O(10^4)$ times the relative perturbation but now we know that the amplification factor could be as bad as $\kappa(A)$.*

Similarly, if we perturbed the entries of a matrix A for a linear system $Ax = b$ so that we have $(A + \delta A)(x + \delta x) = b$ we get

$$Ax + A\delta x + \delta A(x + \delta x) = b \tag{8.64}$$

that is, $A\delta x = -\delta A(x + \delta x)$, which implies that

$$\|\delta x\| \leq \|A^{-1}\|\|\delta A\|\|x + \delta x\| \tag{8.65}$$

for any induced matrix norm and consequently

$$\frac{\|\delta x\|}{\|x+\delta x\|} \leq \|A^{-1}\|\|A\|\frac{\|\delta A\|}{\|A\|} = \kappa(A)\frac{\|\delta A\|}{\|A\|}. \tag{8.66}$$

Because, for any induced norm, $1 = \|I\| = \|A^{-1}A\| \leq \|A^{-1}\|\|A\|$, we get that $\kappa(A) \geq 1$. We say that A is **ill-conditioned** if $\kappa(A)$ is very large.

Example 8.3. *The Hilbert matrix is ill-conditioned. We already saw that in the 2 norm $\kappa(H_5) = 4.7661 \times 10^5$. The condition number increases very rapidly as the size of the Hilbert matrix increases, for example $\kappa(H_6) = 1.4951 \times 10^7$, $\kappa(H_{10}) = 1.6025 \times 10^{13}$.*

8.7.1 What to Do When A is Ill-conditioned?

There are two ways to deal with a linear system with an ill-conditioned matrix A. One approach is to work with extended precision (using as many digits as required to obtain the solution up to a given accuracy). Unfortunately, computations using extended precision can be computationally expensive, several times the cost of regular double precision operations.

A more practical approach is often to replace the ill-conditioned linear system $Ax = b$ by an equivalent linear system with a much smaller condition number. This can be done for example by premultiplying by a nonsingular matrix C^{-1} so that the system $Ax = b$ gets transformed to $C^{-1}Ax = C^{-1}b$, which can be written as

$$C^{-1}AC^{-T}(C^Tx) = C^{-1}b. \tag{8.67}$$

C is selected so that $C^{-1}AC^{-T}$ has a much smaller condition number than that of A. This very useful technique, also employed to accelerate the convergence of some iterative methods, is called **preconditioning**.

8.8 Bibliographic Notes

Section 8.1 . There are several good texts on numerical linear algebra. The book by Golub and Van Loan [GVL13] has been a standard reference in this area of numerical analysis. Another good reference is the text by Demmel [Dem97]. The book by Trefethen and Bau [TB97], presented in the form of lectures at the undergraduate level, covers a selection of numerical linear algebra topics with great eloquence and clarity. For the graduate or advanced undergraduate level Ciarlet's book [Cia89] is a superb reference. Abel's insolvability theorem is beautifully presented in the monograph by Alekseev [Ale04].

Sections 8.2-8.6 . These sections review some basic concepts of matrix analysis. An excellent reference in this general area are the two volumes of Horn and Johnson [HJ13, HJ94]. Schur's triangularization theorem can be found in [HJ13, Cia89]. Also, the classical book by Bellman [Bel97] on matrix analysis is an insightful text full of applications.

Section 8.7 . Demmel [Dem97] relates ill-conditioning of a matrix A to the distance of the nearest singular matrix, and such distance is equal to $1/\kappa(A)$. Thus, A is ill-conditioned if a small perturbation of it renders it singular. Ciarlet [Cia89][Section 2.2] provides an excellent example of an ill-conditioned matrix with integer entries. A detailed discussion of preconditioners can be found in [BBC$^+$94][Chapter 3].

Chapter 9

Linear Systems of Equations I

In this chapter we focus on a problem that is central to many applications, namely, to find the solution to a large linear system of n linear equations in n unknowns $x_1, x_2, \ldots, x_n$:

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1, \\
a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2, \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n.
\end{aligned}
\tag{9.1}
$$

Or written in matrix form

$$
Ax = b, \tag{9.2}
$$

where A is the $n \times n$ matrix of coefficients

$$
A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \tag{9.3}
$$

x is a column vector whose components are the unknowns, and b is the given right hand side of the linear system

$$
x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}. \tag{9.4}
$$

We will assume, unless stated otherwise, that A is a nonsingular, real matrix. That is, the linear system (9.2) has a unique solution for each b. Equivalently, the determinant of A, $\det(A)$, is non-zero and A has an inverse.

While mathematically we can write the solution as $x = A^{-1}b$, this is not computationally efficient. Finding A^{-1} is several (about four) times more costly than solving $Ax = b$ for a given b.

In many applications n can be on the order of millions or much larger.

9.1 Easy to Solve Systems

When A **is diagonal**, i.e.

$$A = \begin{bmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{bmatrix} \tag{9.5}$$

(all the entries outside the diagonal are zero and since A is assumed non-singular $a_{ii} \neq 0$ for all i), then each equation can be solved with just one division:

$$x_i = b_i / a_{ii}, \qquad \text{for } i = 1, 2, \ldots, n. \tag{9.6}$$

If A **is lower triangular** and nonsingular,

$$A = \begin{bmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ \vdots & \vdots & \ddots & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \tag{9.7}$$

the solution can also be obtained easily by the process of *forward substitution*:

$$
\begin{aligned}
x_1 &= \frac{b_1}{a_{11}}, \\
x_2 &= \frac{b_2 - a_{21}x_1}{a_{22}}, \\
x_3 &= \frac{b_3 - [a_{31}x_1 + a_{32}x_2]}{a_{33}}, \\
&\vdots \\
x_n &= \frac{b_n - [a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{n,n-1}x_{n-1}]}{a_{nn}}.
\end{aligned}
\tag{9.8}
$$

The procedure in pseudo-code is listed in Algorithm 9.1.

Algorithm 9.1 Forward Subsitution

for $i = 1, \ldots, n$ **do**
  $x_i \leftarrow \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j \right) / a_{ii}$
end for

The assumption that A is nonsingular implies $a_{ii} \neq 0$ for all $i = 1, 2, \ldots, n$ since $\det(A) = a_{11}a_{22}\cdots a_{nn}$. Also observe that (9.8) shows x_i is a linear combination of $b_i, b_{i-1}, \ldots, b_1$ and since $x = A^{-1}b$ it follows that A^{-1} is also lower triangular.

To compute x_i we perform $i-1$ multiplications, $i-1$ additions/subtractions, and one division, so the total amount of computational work $W(n)$ for forward substitution is

$$W(n) = 2\sum_{i=1}^{n}(i-1) + n = n^2, \tag{9.9}$$

where we have used that

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}. \tag{9.10}$$

That is, $W(n) = O(n^2)$ to solve a lower triangular linear system.

If A is nonsingular and upper triangular

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}, \tag{9.11}$$

we solve the linear system $Ax = b$ starting from x_n, then we solve for x_{n-1}, etc. This is called *backward substitution*

$$\begin{aligned} x_n &= \frac{b_n}{a_{nn}}, \\ x_{n-1} &= \frac{b_{n-1} - a_{n-1,n}x_n}{a_{n-1,n-1}}, \\ x_{n-2} &= \frac{b_{n-2} - [a_{n-2,n-1}x_{n-1} + a_{n-2,n}x_n]}{a_{n-2,n-2}}, \\ &\vdots \\ x_1 &= \frac{b_1 - [a_{12}x_2 + a_{13}x_3 + \cdots a_{1n}x_n]}{a_{11}}. \end{aligned} \tag{9.12}$$

From these equations we deduce that x_i is a linear combination of $b_i.., b_{i+1}, ...b_n$ and so A^{-1} is an upper triangular matrix. Algorithm 9.2 shows the backward substitution procedure in pseudo-code.

Algorithm 9.2 Backward Subsitution

1: **for** $i = n, n-1, \ldots, 1$ **do**

2: $\quad x_i \leftarrow \left(b_i - \sum_{j=i+1}^{n} a_{ij}x_j \right) / a_{ii}$

3: **end for**

The operation count is the same as for forward substitution, $W(n) = O(n^2)$.

9.2 Gaussian Elimination

The central idea of Gaussian elimination is to reduce the linear system $Ax = b$ to an equivalent upper triangular system that has the same solution and can readily be solved with backward substitution. Such reduction is done with an elimination process employing linear combinations of rows. We illustrate first the method with a concrete example:

$$\begin{aligned} x_1 + 2x_2 - x_3 + x_4 &= 0, \\ 2x_1 + 4x_2 - x_4 &= -3, \\ 3x_1 + x_2 - x_3 + x_4 &= 3, \\ x_1 - x_2 + 2x_3 + x_4 &= 3. \end{aligned} \tag{9.13}$$

To do the elimination we form an *augmented matrix* A_b by appending one more column to the matrix of coefficients A, consisting of the right hand side b:

$$A_b = \begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 2 & 4 & 0 & -1 & -3 \\ 3 & 1 & -1 & 1 & 3 \\ 1 & -1 & 2 & 1 & 3 \end{bmatrix}. \tag{9.14}$$

The first step is to eliminate the first unknown in the second to last equations, i.e., to produce a zero in the first column of A_b for rows 2, 3, and 4:

$$\begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 2 & 4 & 0 & -1 & -3 \\ 3 & 1 & -1 & 1 & 3 \\ 1 & -1 & 2 & 1 & 3 \end{bmatrix} \xrightarrow[\substack{R_2 \leftarrow R_2 - 2R_1 \\ R_3 \leftarrow R_3 - 3R1 \\ R_4 \leftarrow R_4 - 1R_1}]{} \begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & -3 & 3 & 0 & 3 \end{bmatrix}, \tag{9.15}$$

where $R_2 \leftarrow R_2 - 2R_1$ means that the second row has been replaced by the second row minus two times the first row, etc. Since the coefficient of x_1 in the first equation is 1 it is easy to figure out the number we need to multiply rows 2, 3, and 4 to achieve the elimination of the first variable for each row, namely 2, 3, and 1. These numbers are called *multipliers*. In general, to obtain the multipliers we divide the coefficient of x_1 in the rows below the first one by the *nonzero* coefficient a_{11} (2/1=2, 3/1=3, 1/1=1). The coefficient we need to divide by to obtain the multipliers is called a *pivot* (1 in this case).

Note that the $(2, 2)$ element of the last matrix in (9.15) is 0 so we cannot use it as a pivot for the second round of elimination. Instead, we proceed by exchanging the second and the third rows

$$\begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & -3 & 3 & 0 & 3 \end{bmatrix} \xrightarrow[R_2 \leftrightarrow R_3]{} \begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & -3 & 3 & 0 & 3 \end{bmatrix}. \tag{9.16}$$

We can now use -5 as a pivot and do the second round of elimination:

$$\begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & -3 & 3 & 0 & 3 \end{bmatrix} \xrightarrow[\substack{R_3 \leftarrow R_3 - 0R_2 \\ R_4 \leftarrow R_4 - \frac{3}{5}R_2}]{} \begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & 0 & \frac{9}{5} & \frac{6}{5} & \frac{6}{5} \end{bmatrix}. \tag{9.17}$$

Clearly, the elimination step $R_3 \leftarrow R_3 - 0R_2$ is unnecessary as the coefficient to be eliminated is already zero but we include it to illustrate the general procedure. The last round of the elimination is

$$\begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & 0 & \frac{9}{5} & \frac{6}{5} & \frac{6}{5} \end{bmatrix} \xrightarrow[R_4 \leftarrow R_4 - \frac{9}{10}R_3]{} \begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & 0 & 0 & \frac{39}{10} & \frac{39}{10} \end{bmatrix}, \tag{9.18}$$

The last matrix, let us call it U_b, corresponds to the upper triangular system

$$\begin{bmatrix} 1 & 2 & -1 & 1 \\ 0 & -5 & 2 & -2 \\ 0 & 0 & 2 & -3 \\ 0 & 0 & 0 & \frac{39}{10} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ -3 \\ \frac{39}{10} \end{bmatrix}, \tag{9.19}$$

which we can solve with backward substitution to obtain the solution

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 1 \end{bmatrix}. \tag{9.20}$$

Each of the steps in the Gaussian elimination process are linear transformations and hence we can represent them with matrices. *Note, however, that these matrices are not constructed in practice, we only implement their* *effect (row exchange or elimination)*. The first round of elimination (9.15) is equivalent to multiplying (from the left) A_b by the lower triangular matrix

$$E_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ -3 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}, \tag{9.21}$$

that is

$$E_1 A_b = \begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & -3 & 3 & 0 & 3 \end{bmatrix}. \tag{9.22}$$

The matrix E_1 is formed by taking the 4×4 identity matrix and replacing the elements in the first column below 1 by negative the multiplier, i.e. $-2, -3, -1$. We can exchange rows 2 and 3 with the *permutation* matrix

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{9.23}$$

that is obtained by exchanging the second and third rows in the 4×4 identity matrix,

$$PE_1 A_b = \begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & -3 & 3 & 0 & 3 \end{bmatrix}. \tag{9.24}$$

To construct the matrix associated with the second round of elimination we take the 4×4 identity matrix and replace the elements in the second column below the diagonal by negative the multipliers we got with the pivot equal to -5:

$$E_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -\frac{3}{5} & 0 & 1 \end{bmatrix}, \tag{9.25}$$

and we get

$$E_2PE_1A_b = \begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & 0 & \frac{9}{5} & \frac{6}{5} & \frac{6}{5} \end{bmatrix}. \tag{9.26}$$

Finally, for the last elimination we have

$$E_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{9}{10} & 1 \end{bmatrix}, \tag{9.27}$$

and $E_3E_2PE_1A_b = U_b$.

Observe that $PE_1A_b = E_1'PA_b$, where

$$E_1' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ -2 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}, \tag{9.28}$$

i.e., we exchange rows in advance and then reorder the multipliers accordingly. If we focus on the matrix A, the first four columns of A_b, we have the matrix factorization

$$E_3E_2E_1'PA = U, \tag{9.29}$$

where U is the upper triangular matrix

$$U = \begin{bmatrix} 1 & 2 & -1 & 1 \\ 0 & -5 & 2 & -2 \\ 0 & 0 & 2 & -3 \\ 0 & 0 & 0 & \frac{39}{10} \end{bmatrix}. \tag{9.30}$$

Moreover, the product of upper (lower) triangular matrices is also an upper (lower) triangular matrix and so is the inverse. Hence, we obtain the so-called LU factorization

$$PA = LU, \tag{9.31}$$

where $L = (E_3 E_2 E_1')^{-1} = E_1'^{-1} E_2^{-1} E_3^{-1}$ is a lower triangular matrix. Now, recall that the matrices E_1', E_2, E_3 perform the transformation of subtracting the row of the pivot times the multiplier to the rows below. Therefore, the inverse operation is to add the subtracted row back, i.e., we simply remove the negative sign in front of the multipliers,

$$E_1'^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \quad E_2^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & \frac{3}{5} & 0 & 1 \end{bmatrix}, \quad E_3^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{9}{10} & 1 \end{bmatrix}.$$

It then follows that

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 1 & \frac{3}{5} & \frac{9}{10} & 1 \end{bmatrix}. \tag{9.32}$$

Note that L has all the multipliers below its diagonal and U has all the pivots on its diagonal. We will see that a factorization of the form $PA = LU$ is always possible for any nonsingular $n \times n$ matrix A and can be very useful.

We consider now the general linear system (9.1). The matrix of coefficients and the right hand size are

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}, \tag{9.33}$$

respectively. We form the augmented matrix A_b by appending b to A as the last column:

$$A_b = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{bmatrix}. \tag{9.34}$$

In principle, if $a_{11} \neq 0$ we can start the elimination. However, if $|a_{11}|$ is too small, dividing by it to compute the multipliers might lead to inaccurate results in the computer, i.e. using finite precision arithmetic. It is generally better to look for the coefficient of largest absolute value in the first column, to exchange rows, and then do the elimination. This is called partial *pivoting*. It is also possible after this to search for the element of largest absolute value in the first row and switch columns accordingly. This is called complete pivoting and works well provided the matrix is properly scaled. Henceforth, we will consider Gaussian elimination only with partial pivoting, which is less costly to apply.

To perform the first round of Gaussian elimination we do three steps:

1. Find the $\max_i |a_{i1}|$, let us say this corresponds to the m-th row, i.e. $|a_{m1}| = \max_i |a_{i1}|$. If $|a_{m1}| = 0$, the matrix is singular. Stop.

2. Exchange rows 1 and m.

3. Compute the multipliers and perform the elimination.

After these three steps, we have transformed A_b into

$$A_b^{(1)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{bmatrix}. \tag{9.35}$$

This corresponds to $A_b^{(1)} = E_1 P_1 A_b$, where P_1 is the permutation matrix that exchanges rows 1 and m ($P_1 = I$ if no exchange is made) and E_1 is the matrix representing the elimination of the entries below the first element in the first column. The same three steps above can be applied now to the smaller $(n-1) \times n$ matrix

$$\tilde{A}_b^{(1)} = \begin{bmatrix} a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{bmatrix}, \tag{9.36}$$

and so on. Performing this process $(n-1)$ times, we obtain the reduced, upper triangular system, which can be solved with backward substitution.

In matrix terms, the linear transformations in the Gaussian elimination process with partial pivoting correspond to

$$A_b^{(k)} = E_k P_k A_b^{(k-1)}, \quad k = 1, 2, \ldots, n-1, \tag{9.37}$$

with $A_b^{(0)} = A_b$ and where P_k and E_k are permutation and elimination matrices, respectively. $P_k = I$ if no row exchange is made prior to the k-th elimination round (but recall that we do not construct the matrices E_k and P_k in practice). Hence, the Gaussian elimination process for a nonsingular linear system produces the matrix factorization

$$U_b = A_b^{(n-1)} = E_{n-1}P_{n-1}E_{n-2}P_{n-2}\cdots E_1P_1A_b. \tag{9.38}$$

Arguing as in the introductory example we can rearrange the rows of A_b, with the permutation matrix $P = P_{n-1}\cdots P_1$ and the corresponding multipliers, as if we knew in advance the row exchanges that would be needed to get

$$U_b \equiv A_b^{(n-1)} = E'_{n-1}E'_{n-2}\cdots E'_1 PA_b. \tag{9.39}$$

Since the inverse of $E'_{n-1}E'_{n-2}\cdots E'_1$ is the lower triangular matrix

$$L = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{n,n-1} & 1 \end{bmatrix}, \tag{9.40}$$

where the l_{ij}, $j = 1,\ldots,n-1$, $i = j+1,\ldots,n$ are the multipliers (computed after all the rows have been rearranged), we arrive at the anticipated factorization $PA = LU$. Incidentally, up to sign, Gaussian elimination also produces the determinant of A because

$$\det(PA) = \pm\det(A) = \det(LU) = \det(U) = a_{11}^{(1)}a_{22}^{(2)}\cdots a_{nn}^{(n)} \tag{9.41}$$

and so $\pm\det(A)$ equals the product of all the pivots in the elimination process.

In the implementation of Gaussian elimination the array storing the augmented matrix A_b is overwritten to save memory. The pseudo-code of Gaussian elimination with partial pivoting (assuming $a_{i,n+1} = b_i$, $i = 1,\ldots,n$) is presented in Algorithm 9.3.

9.2.1 The Cost of Gaussian Elimination

We do now an operation count of Gaussian elimination to solve an $n \times n$ linear system $Ax = b$.

Algorithm 9.3 Gaussian Elimination with Partial Pivoting

for $j = 1, \ldots, n-1$ **do**
  Find m such that $|a_{mj}| = \max\limits_{j \le i \le n} |a_{ij}|$
  if $|a_{mj}| = 0$ **then**
    stop ▷ Matrix is singular
  end if
  $a_{jk} \leftrightarrow a_{mk}, \quad k = j, \ldots, n+1$ ▷ Exchange rows
  for $i = j+1, \ldots, n$ **do**
    $m \leftarrow a_{ij}/a_{jj}$ ▷ Compute multiplier
    $a_{ik} \leftarrow a_{ik} - m * a_{jk}, \quad k = j+1, \ldots, n+1$ ▷ Elimination
    $a_{ij} \leftarrow m$ ▷ Store multiplier
  end for
end for
for $i = n, n-1, \ldots, 1$ **do** ▷ Backward Substitution
  $x_i \leftarrow \left(a_{i,n+1} - \sum\limits_{j=i+1}^{n} a_{ij} x_j \right) / a_{ii}$
end for

We focus on the elimination as we already know the work for backward substitution is $O(n^2)$. For each round of elimination, $j = 1, \ldots, n-1$, we need one division to compute each of the $n-j$ multipliers, and $(n-j)(n-j+1)$ multiplications and $(n-j)(n-j+1)$ sums (subtracts) to perform the eliminations. Thus, the total number of operations is

$$\begin{aligned} W(n) &= \sum_{j=1}^{n-1} \left[2(n-j)(n-j+1) + (n-j) \right] \\ &= \sum_{j=1}^{n-1} \left[2(n-j)^2 + 3(n-j) \right] \end{aligned} \tag{9.42}$$

and using (9.10) and

$$\sum_{i=1}^{m} i^2 = \frac{m(m+1)(2m+1)}{6}, \tag{9.43}$$

we get

$$W(n) = \frac{2}{3} n^3 + O(n^2). \tag{9.44}$$

Thus, Gaussian elimination is computationally expensive for large systems of equations.

9.3 LU and Choleski Factorizations

If Gaussian elimination can be performed without row interchanges, then we obtain $A = LU$. This factorization can be advantageous when solving many linear systems with the same $n \times n$ matrix A but different right hand sides b because we can turn the problem $Ax = b$ into two triangular linear systems, which can be solved much more economically in $O(n^2)$ operations. Indeed, from $LUx = b$ and setting $y = Ux$ we have

$$Ly = b, \tag{9.45}$$

$$Ux = y. \tag{9.46}$$

Given b, we can solve the first system for y with forward substitution and then we solve the second system for x with backward substitution. Thus, while the LU factorization of A has an $O(n^3)$ cost, subsequent solutions to the linear system with the same matrix A but different b can be done in $O(n^2)$ operations.

When is it possible to get the factorization $A = LU$? the following result provides a useful sufficient condition.

Theorem 9.1. *Let A be an $n \times n$ matrix whose leading principal submatrices $A_1, \ldots, A_n$ are all nonsingular. Then, there exist an $n \times n$ lower triangular matrix L, with ones on its diagonal, and an $n \times n$ upper triangular matrix U such that $A = LU$ and this factorization is unique.*

Proof. Since A_1 is nonsingular, then $a_{11} \neq 0$ and $P_1 = I$. Suppose now that we do not need to exchange rows in steps $2, \ldots, k$ so that $A^{(k)} = E_k \cdots E_2 E_1 A$, that is

$$\left[\begin{array}{ccc|cc} a_{11} & \cdots & a_{1k} & \cdots & a_{1n} \\ & \ddots & & \cdots & \\ & & & \cdots & \\ & & a^{(k)}_{kk} & \cdots & a^{(k)}_{kn} \\ \hline & & \vdots & & \vdots \\ & & a^{(k)}_{nk} & \cdots & a^{(k)}_{nn} \end{array}\right] = \left[\begin{array}{ccc|c} 1 & & & \\ -m_{21} & \ddots & & \\ \vdots & & & \\ -m_{k1} & & 1 & \\ \hline \vdots & & & \\ -m_{n1} & \cdots & & 1 \end{array}\right] \left[\begin{array}{ccc|cc} a_{11} & \cdots & a_{1k} & \cdots & a_{1n} \\ \vdots & \ddots & & \cdots & \vdots \\ & & & \cdots & \\ a_{k1} & & a_{kk} & \cdots & a_{kn} \\ \hline \vdots & & \vdots & & \vdots \\ a_{n1} & \cdots & a_{nk} & \cdots & a_{nn} \end{array}\right].$$

The determinant of the boxed $k \times k$ leading principal submatrix on the left is $a_{11}a_{22}^{(2)} \cdots a_{kk}^{(k)}$ and this is equal to the determinant of the product of boxed blocks on the right hand side. Since the determinant of the first such block is one (it is a lower triangular matrix with ones on the diagonal), it follows that

$$a_{11}a_{22}^{(2)} \cdots a_{kk}^{(k)} = \det(A_k) \neq 0, \tag{9.47}$$

which implies that $a_{kk}^{(k)} \neq 0$ and so $P_{k+1} = I$. We conclude that $U = E_{n-1} \cdots E_1 A$ and therefore $A = LU$.

Let us now show that this decomposition is unique. Suppose $A = L_1U_1 = L_2U_2$ then

$$L_2^{-1}L_1 = U_2U_1^{-1}. \tag{9.48}$$

But the matrix on the left hand side is lower triangular (with ones in its diagonal) whereas the one on the right hand side is upper triangular. Therefore $L_2^{-1}L_1 = I = U_2U_1^{-1}$, which implies that $L_2 = L_1$ and $U_2 = U_1$. □

An immediate consequence of this result is that Gaussian elimination can be performed without row interchange for a strictly diagonally dominant (SDD) matrix, as each of its leading principal submatrices is itself SDD, and for a positive definite matrix, as each of its leading principal submatrices is itself positive definite, and hence nonsingular in both cases.

Corollary 9.3.1. *Let A be an $n \times n$ matrix. Then, $A = LU$, where L is an $n \times n$ lower triangular matrix , with ones on its diagonal, and U is an $n \times n$ upper triangular matrix if either*

(a) A is SDD or

(b) A is symmetric, positive definite.

In the case of a positive definite matrix the number number of operations can be cut down in approximately half by exploiting symmetry. The idea is to obtain a factorization $A = BB^T$, where B is a lower triangular matrix with positive entries in its diagonal. This representation is called Choleski factorization of a symmetric, positive definite matrix A.

Theorem 9.2. *Let A be a symmetric, positive definite matrix. Then, there is a unique lower triangular matrix B with positive entries in its diagonal such that $A = BB^T$.*

Proof. By Corollary 9.3.1 A has an LU factorization. Moreover, from (9.47) it follows that all the pivots are positive and thus $u_{ii} > 0$ for all $i = 1, \ldots, n$. We can split the pivots evenly in L and U by letting

$$D = \operatorname{diag}(\sqrt{u_{11}}, \ldots, \sqrt{u_{nn}}) \tag{9.49}$$

and writing $A = LDD^{-1}U = (LD)(D^{-1}U)$. Let $B = LD$ and $C = D^{-1}U$. Both matrices have diagonal elements $\sqrt{u_{11}}, \ldots, \sqrt{u_{nn}}$ but B is lower triangular while C is upper triangular. Moreover, $A = BC$ and because $A^T = A$ we have that $C^T B^T = BC$, which implies

$$B^{-1}C^T = C(B^T)^{-1}. \tag{9.50}$$

The matrix on the left hand side is lower triangular with ones in its diagonal while the matrix on the right hand side is upper triangular also with ones in its diagonal. Therefore, $B^{-1}C^T = I = C(B^T)^{-1}$ and thus, $C = B^T$ and $A = BB^T$.

To prove that this symmetric factorization is unique we go back to the LU factorization, which we know is unique, if we choose L to have ones in its diagonal. Given $A = BB^T$, where B is lower triangular with positive diagonal elements $b_{11}, \ldots, b_{nn}$, we can write

$$A = BD_B^{-1}D_B B^T, \tag{9.51}$$

where $D_B = \operatorname{diag}(b_{11}, \ldots, b_{nn})$. Then, $L = BD_B^{-1}$ and $U = D_B B^T$ yield the unique LU factorization of A. Now, suppose there is another Choleski factorization $A = CC^T$. Then, by the uniqueness of the LU factorization, we have

$$L = BD_B^{-1} = CD_C^{-1}, \tag{9.52}$$

$$U = D_B B^T = D_C C^T, \tag{9.53}$$

where $D_C = \operatorname{diag}(c_{11}, \ldots, c_{nn})$. Equation (9.53) implies that $b_{ii}^2 = c_{ii}^2$ for $i = 1, \ldots, n$ and since $b_{ii} > 0$ and $c_{ii} > 0$ for all i, then $D_C = D_B$ and consequently $C = B$. □

The Choleski factorization is usually written as $A = LL^T$ and is obtained by exploiting the lower triangular structure of L and symmetry as follows. First, $L = (l_{ij})$ is lower triangular then $l_{ij} = 0$ for $1 \leq i < j \leq n$ and thus

$$a_{ij} = \sum_{k=1}^{n} l_{ik}l_{jk} = \sum_{k=1}^{\min(i,j)} l_{ik}l_{jk}. \tag{9.54}$$

Now, because $A^T = A$ we only need a_{ij} for $i \leq j$, that is

$$a_{ij} = \sum_{k=1}^{i} l_{ik} l_{jk} \qquad 1 \leq i \leq j \leq n. \tag{9.55}$$

We can solve equations (9.55) to determine L, one column at a time. If we set $i = 1$ we get

$$\begin{aligned}
a_{11} &= l_{11}^2, \quad \rightarrow l_{11} = \sqrt{a_{11}}, \\
a_{12} &= l_{11} l_{21}, \\
&\vdots \\
a_{1n} &= l_{11} l_{n1}
\end{aligned}$$

and this allows us to get the first column of L. The second column is now found by using (9.55) for $i = 2$

$$\begin{aligned}
a_{22} &= l_{21}^2 + l_{22}^2, \quad \rightarrow l_{22} = \sqrt{a_{22} - l_{21}^2}, \\
a_{23} &= l_{21} l_{31} + l_{22} l_{32}, \\
&\vdots \\
a_{2n} &= l_{21} l_{n1} + l_{22} l_{n2},
\end{aligned}$$

etc. Algorithm 9.4 gives the pseudo code for the Choleski factorization.

Algorithm 9.4 Choleski factorization

for $i = 1, \ldots, n$ **do** $\triangleright$ Compute column i of L for $i = 1, \ldots, n$
- $l_{ii} \leftarrow \sqrt{\left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2\right)}$
- **for** $j = i + 1, \ldots, n$ **do**
  - $l_{ji} \leftarrow (a_{ij} - \sum_{k=1}^{i-1} l_{ik} l_{jk}) / l_{ii}$
- **end for**

end for

9.4 Tridiagonal Linear Systems

If the matrix of coefficients A has a tridiagonal structure

$$A = \begin{bmatrix} a_1 & b_1 & & & \\ c_1 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & & & b_{n-1} \\ & & & c_{n-1} & a_n \end{bmatrix} \tag{9.56}$$

its LU factorization can be computed at an $O(n)$ cost and the corresponding linear system can thus be solved efficiently.

Theorem 9.3. *If A is triadiagonal and all of its leading principal submatrices are nonsingular then*

$$\begin{aligned} &\begin{bmatrix} a_1 & b_1 & & & \\ c_1 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & & & b_{n-1} \\ & & & c_{n-1} & a_n \end{bmatrix} = \\ &\begin{bmatrix} 1 & & & & \\ l_1 & 1 & & & \\ & \ddots & \ddots & & \\ & & & & \\ & & & l_{n-1} & 1 \end{bmatrix} \begin{bmatrix} m_1 & b_1 & & & \\ & m_2 & b_2 & & \\ & & \ddots & \ddots & \\ & & & & b_{n-1} \\ & & & & m_n \end{bmatrix}, \end{aligned} \tag{9.57}$$

where

$$m_1 = a_1, \tag{9.58}$$

$$l_j = c_j/m_j, \quad m_{j+1} = a_{j+1} - l_j b_j, \quad \textit{for } j = 1, \ldots, n-1, \tag{9.59}$$

and this factorization is unique.

Proof. By Theorem 9.1 we know that A has a unique LU factorization, where L is unit lower triangular and U is upper triangular. We will show that we can solve uniquely for $l_1, \ldots, l_{n-1}$ and $m_1, \ldots, m_n$ so that (9.57) holds. Equating the matrix product on the right hand side of (9.57), row by row, we get

$$
\begin{aligned}
\text{1st row: } & a_1 = m_1, \quad b_1 = b_1, \\
\text{2nd row: } & c_1 = m_1 l_1, \quad a_2 = l_1 b_1 + m_2, \quad b_2 = b_2, \\
& \vdots \\
(n-1)\text{-st row: } & c_{n-2} = m_{n-2} l_{n-2}, \quad a_{n-1} = l_{n-2} b_{n-2} + m_{n-1}, \quad b_{n-1} = b_{n-1}, \\
n\text{-th row: } & c_{n-1} = m_{n-1} l_{n-1}, \quad a_n = l_{n-1} b_{n-1} + m_n
\end{aligned}
$$

from which (9.58)-(9.59) follows. Of course, we need the m_j's to be nonzero to use (9.59). We now prove this is the case.

Note that $m_{j+1} = a_{j+1} - l_j b_j = a_{j+1} - \frac{c_j}{m_j} b_j$. Therefore

$$
m_j m_{j+1} = a_{j+1} m_j - b_j c_j, \quad \text{for } j = 1, \ldots, n-1. \tag{9.60}
$$

Thus,

$$
\det(A_1) = a_1 = m_1, \tag{9.61}
$$

$$
\det(A_2) = a_2 a_1 - c_1 b_1 = a_2 m_1 - b_1 c_1 = m_1 m_2. \tag{9.62}
$$

We now do induction to show that $\det(A_k) = m_1 m_2 \cdots m_k$. Suppose $\det(A_j) = m_1 m_2 \cdots m_j$ for $j = 1, \ldots, k-1$. Expanding by the last row we get

$$
\det(A_k) = a_k \det(A_{k-1}) - b_{k-1} c_{k-1} \det(A_{k-2}) \tag{9.63}
$$

and using the induction hypothesis and (9.60) it follows that

$$
\det(A_k) = m_1 m_2 \cdots m_{k-2} \left[a_k m_{k-1} - b_{k-1} c_{k-1} \right] = m_1 \cdots m_k, \tag{9.64}
$$

for $k = 1, \ldots, n$. Since $\det(A_k) \neq 0$ for $k = 1, \ldots, n$ then $m_1, m_2, \ldots, m_n$ are all nonzero. $\square$

9.5 A 1D BVP: Deformation of an Elastic Beam

We saw in Section 4.3 an example of a very large system of equations in connection with the least squares problem for fitting high dimensional data.

Algorithm 9.5 Tridiagonal solver

$m_1 \leftarrow a_1$
for $j = 1, \ldots, n-1$ **do** ▷ Compute column L and U
  $l_j \leftarrow c_j / m_j$
  $m_{j+1} \leftarrow a_{j+1} - l_i * b_j$
end for
$y_1 \leftarrow d_1$ ▷ Forward substitution on $Ly = d$
for $j = 2, \ldots, n$ **do**
  $y_j \leftarrow d_j - l_{j-1} * y_{j-1}$
end for
$x_n \leftarrow y_n / m_n$ ▷ Backward substitution on $Ux = y$
for $j = n-1, n-2 \ldots, 1$ **do**
  $x_j \leftarrow (y_j - b_j * x_{j+1}) / m_j$
end for

We now consider another example that leads to a large linear system of equations.

Suppose we have a thin beam of unit length, stretched horizontally and occupying the interval $[0, 1]$. The beam is subjected to a load density $f(x)$ at each point $x \in [0, 1]$, and pinned at end points. Let $u(x)$ be the beam deformation from the horizontal position. Assuming that the deformations are small (linear elasticity regime), u satisfies

$$-u''(x) + c(x)u(x) = f(x), \quad 0 < x < 1, \tag{9.65}$$

where $c(x) \geq 0$ is related to the elastic, material properties of the beam. Because the beam is pinned at the end points we have the boundary conditions

$$u(0) = u(1) = 0. \tag{9.66}$$

The system (9.65)-(9.66) is called a *boundary value problem* (BVP). That is, we need to find a function u that satisfies the ordinary differential equation (9.65) and the boundary conditions (9.66) for any given, continuous f and c. The condition $c(x) \geq 0$ guarantees existence and uniqueness of the solution to this problem.

We will construct a discrete model whose solution gives an accurate approximation to the exact solution at a finite collection of selected points (called nodes) in $[0, 1]$. We take the nodes to be equally spaced and to include the interval's end points (boundary). Thus, we choose a positive integer

N and define the nodes or grid points

$$x_0 = 0, x_1 = h, x_2 = 2h, \ldots, x_N = Nh, x_{N+1} = 1, \tag{9.67}$$

where $h = 1/(N+1)$ is the *grid size* or node spacing. The nodes $x_1, \ldots, x_N$ are called interior nodes, because they lie inside the interval $[0, 1]$, and the nodes x_0 and x_{N+1} are called boundary nodes.

We now construct a discrete approximation to the ordinary differential equation by replacing the second derivative with a second order finite difference approximation. As we know, for sufficiently smooth u,

$$u''(x_j) = \frac{u(x_{j+1}) - 2u(x_j) + u(x_{j-1})}{h^2} + O(h^2). \tag{9.68}$$

Neglecting the $O(h^2)$ error and denoting the approximation of $u(x_j)$ by u_j (i.e. $u_j \approx u(x_j)$) we get

$$-\frac{u_{j-1} - 2u_j + u_{j+1}}{h^2} + c_j u_j \quad = \quad f_j, \quad j = 1, 2, \ldots, N, \tag{9.69}$$

where $f_j = f(x_j)$ and $c_j = c(x_j)$. The boundary condition (9.66) translates into

$$v_0 = v_{N+1} = 0. \tag{9.70}$$

Thus, (9.69) is a linear system of N equations in N unknowns $u_1, \ldots, u_N$, which we can write in matrix form as

$$\frac{1}{h^2}\begin{bmatrix} 2+c_1h^2 & -1 & 0 & \cdots & & \cdots & 0 \\ -1 & 2+c_2h^2 & -1 & \ddots & & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & 0 \\ & & & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & & & 0 & -1 & 2+c_Nh^2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ u_N \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ f_N \end{bmatrix}. \tag{9.71}$$

The matrix, let us call it A, of this system is tridiagonal and symmetric. A direct computation shows that for an arbitrary, nonzero, column vector

$v = [v_1, \ldots, v_N]^T$

$$v^T A v = \sum_{j=1}^{N} \left[\left(\frac{v_j + c_j v_j}{h} \right)^2 + c_j v_j^2 \right] > 0, \quad \forall\, v \neq 0 \tag{9.72}$$

and therefore, since $c_j \geq 0$ for all j, A is positive definite. Thus, there is a unique solution to (9.71) and this can be efficiently found with our tridiagonal solver, Algorithm 9.5. Since the expected numerical error is $O(h^2) = O(1/(N+1)^2)$, even a modest accuracy of $O(10^{-4})$ requires $N \approx 100$.

9.6 A 2D BVP: Dirichlet Problem for the Poisson's Equation

We now look at a simple 2D BVP for an equation that is central to many applications, namely Poisson's equation. For concreteness here, we can think of the equation as a model for small deformations u of a stretched, square membrane fixed to a wire at its boundary and subject to a force density f. Denoting by Ω, and $\partial\Omega$, the unit square $[0,1] \times [0,1]$ and its boundary, respectively, the BVP is to find u such that

$$-\Delta u(x,y) = f(x,y), \quad \text{for } (x,y) \in \Omega \tag{9.73}$$

and

$$u(x,y) = 0. \quad \text{for } (x,y) \in \partial\Omega. \tag{9.74}$$

In (9.73), Δu is the Laplacian of u, also denoted as $\nabla^2 u$, and is given by

$$\Delta u = \nabla^2 u = u_{xx} + u_{yy} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}. \tag{9.75}$$

Equation (9.73) is Poisson's equation (in 2D) and together with (9.74) specify a (homogeneous) Dirichlet problem because the value of u is given at the boundary.

To construct a numerical approximation to (9.73)-(9.74), we proceed as in the previous 1D BVP example by discretizing the domain. For simplicity, we will use uniformly spaced grid points. We choose a positive integer N and define the grid points of our domain $\Omega = [0,1] \times [0,1]$ as

$$(x_j, y_m) = (jh, mh), \quad \text{for } j, m = 0, \ldots, N+1, \tag{9.76}$$

where $h = 1/(N+1)$. The interior nodes correspond to $1 \leq j, m \leq N$ and the boundary nodes are those corresponding to the remaining values of indices j and m (j or m equal 0 and j or m equal $N+1$).

At each of the interior nodes we replace the Laplacian by its second order order *finite difference approximation*, called the *five-point discrete Laplacian*

$$\nabla^2 u(x_j, y_m) = \frac{u(x_{j-1}, y_m) + u(x_{j+1}, y_m) + u(x_j, y_{m-1}) + u(x_j, y_{m+1}) - 4u(x_j, y_m)}{h^2} + O(h^2). \tag{9.77}$$

Neglecting the $O(h^2)$ discretization error and denoting by u_{jm} the approximation to $u(x_j, y_m)$ we get:

$$-\frac{u_{j-1,m} + u_{j+1,m} + u_{j,m-1} + u_{j,m+1} - 4u_{jm}}{h^2} = f_{jm}, \text{ for } 1 \leq j, m \leq N. \tag{9.78}$$

This is a linear system of N^2 equations for the N^2 unknowns, u_{jm}, $1 \leq j, m \leq N$. We have freedom to order or label the unknowns any way we wish and that will affect the structure of the matrix of coefficients of the linear system but remarkably the matrix will be symmetric positive definite regardless of the ordering of the unknowns!.

The most common labeling is the so-called *lexicographical order*, which proceeds from the bottom row to the top one, left to right, $u_{11}, u_{12}, \ldots, u_{1N}$, $u_{21}, \ldots$, etc. Denoting by $u_1 = [u_{11}, u_{12}, \ldots, u_{1N}]^T$, $u_2 = [u_{21}, u_{22}, \ldots, u_{2N}]^T$, etc., and similarly for the right hand side f, the linear system (9.78) can be written in matrix form as

$$\begin{bmatrix} T & -I & 0 & & & & 0 \\ -I & T & -I & \ddots & & & \\ 0 & \ddots & \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & 0 \\ & & & \ddots & \ddots & \ddots & -I \\ 0 & & & & 0 & -I & T \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ u_N \end{bmatrix} = h^2 \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ f_N \end{bmatrix}. \tag{9.79}$$

Here, I is the $N \times N$ identity matrix and T is the $N \times N$ tridiagonal matrix

$$T = \begin{bmatrix} 4 & -1 & 0 & & & & 0 \\ -1 & 4 & -1 & \ddots & & & \\ 0 & \ddots & \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & 0 \\ & & & \ddots & \ddots & \ddots & -1 \\ 0 & & & & 0 & -1 & 4 \end{bmatrix}. \tag{9.80}$$

Thus, the matrix of coefficients in (9.79), is *sparse*, i.e. the vast majority of its entries are zeros. For example, for $N = 3$ this matrix is

$$\begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix}.$$

Gaussian elimination is hugely inefficient for a large system with a sparse matrix, as in this example. This is because the intermediate matrices in the elimination would be generally dense due to *fill-in* introduced by the elimination process. To illustrate the high cost of Gaussian elimination, if we merely use $N = 100$ (this corresponds to a modest discretization error of $O(10^{-4})$), we end up with $n = N^2 = 10^4$ unknowns and the cost of Gaussian elimination would be $O(10^{12})$ operations.

9.7 Linear Iterative Methods for $Ax = b$

As we have seen, Gaussian elimination is an expensive procedure for large linear systems of equations. An alternative is to seek not an exact (up to roundoff error) solution in a finite number of steps but an approximation to the solution that can be obtained from an iterative procedure applied to an initial guess $x^{(0)}$.

We are going to consider first a class of iterative methods where the central idea is to write the matrix A as the sum of a nonsingular matrix M, whose corresponding system is easy to solve, and a remainder $-N = A - M$, so that the system $Ax = b$ is transformed into the equivalent system

$$Mx = Nx + b. \tag{9.81}$$

Starting with an initial guess $x^{(0)}$, (9.81) defines a sequence of approximations generated by

$$Mx^{(k+1)} = Nx^{(k)} + b, \quad k = 0, 1, \ldots \tag{9.82}$$

The main questions are

1. When does this iteration converge?
2. What determines its rate of convergence?
3. What is the computational cost?

But first we look at three concrete iterative methods of the form (9.82). Unless otherwise stated, A is assumed to be a non-singular, $n \times n$ matrix and b a given n-column vector.

9.7.1 Jacobi, Gauss-Seidel, and SOR.

If the all the diagonal elements of A are nonzero we can take $M = \text{diag}(A)$ and then at each iteration (i.e. for each k) the linear system (9.82) can be easily solved to obtain the next iterate $x^{(k+1)}$. Note that we do not need to compute M^{-1} neither do we need to perform the matrix product $M^{-1}N$ (and due to its cost it should be avoided). We just need to solve the linear system with the matrix M, which in this case is trivial to do. We simply solve the first equation for the first unknown, the second equation for the second unknown, etc. This is called *Jacobi*'s iterative method:

for $k = 0, 1, \ldots$

$$x_i^{(k+1)} = \frac{-\displaystyle\sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} + b_i}{a_{ii}}, \; i = 1, 2, ..., n. \tag{9.83}$$

The iteration could be stopped when

$$\frac{\|x^{(k+1)} - x^{(k)}\|_\infty}{\|x^{(k+1)}\|_\infty} \leq \text{tolerance.} \tag{9.84}$$

Example 9.1. *Consider the* 4×4 *linear system*

$$\begin{aligned} 10x_1 - x_2 + 2x_3 &= 6, \\ -x_1 + 11x_2 - x_3 + 3x_4 &= 25, \\ 2x_1 - x_2 + 10x_3 - x_4 &= -11, \\ 3x_2 - x_3 + 8x_4 &= 15. \end{aligned} \tag{9.85}$$

It has the unique solution (1,2,-1,1). Jacobi's iteration for this system, for $k = 0, 1, \ldots,$ *is*

$$\begin{aligned} x_1^{(k+1)} &= \frac{1}{10}x_2^{(k)} - \frac{1}{5}x_3^{(k)} + \frac{3}{5}, \\ x_2^{(k+1)} &= \frac{1}{11}x_1^{(k)} + \frac{1}{11}x_3^{(k)} - \frac{3}{11}x_4^{(k)} + \frac{25}{11}, \\ x_3^{(k+1)} &= -\frac{1}{5}x_1^{(k)} + \frac{1}{10}x_2^{(k)} + \frac{1}{10}x_4^{(k)} - \frac{11}{10}, \\ x_4^{(k+1)} &= -\frac{3}{8}x_2^{(k)} + \frac{1}{8}x_3^{(k)} + \frac{15}{8}. \end{aligned} \tag{9.86}$$

Starting with $x^{(0)} = [0, 0, 0, 0]^T$ *we obtain*

$$x^{(1)} = \begin{bmatrix} 0.60000000 \\ 2.27272727 \\ -1.10000000 \\ 1.87500000 \end{bmatrix}, \quad x^{(2)} = \begin{bmatrix} 1.04727273 \\ 1.71590909 \\ -0.80522727 \\ 0.88522727 \end{bmatrix}, \quad x^{(3)} = \begin{bmatrix} 0.93263636 \\ 2.05330579 \\ -1.04934091 \\ 1.13088068 \end{bmatrix}. \tag{9.87}$$

In Jacobi's iteration, when we evaluate $x_2^{(k+1)}$ we have already $x_1^{(k+1)}$ available. When we evaluate $x_3^{(k+1)}$ we have already $x_1^{(k+1)}$ and $x_2^{(k+1)}$ available and so on. If we update Jacobi's iteration with the already computed components of $x^{(k+1)}$ we obtain the so-called *Gauss-Seidel*'s iteration:

for $k = 0, 1, \ldots$

$$x_i^{(k+1)} = \frac{-\displaystyle\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} + b_i}{a_{ii}}, \quad i = 1, 2, ..., n. \tag{9.88}$$

Gauss-Seidel's iteration is equivalent to that obtained by taking M as the lower triangular part of the matrix A, including its diagonal.

Example 9.2. *For the system (9.85), starting again with the initial guess* $[0, 0, 0, 0]^T$*, Gauss-Seidel's iteration produces the following approximations:*

$$x^{(1)} = \begin{bmatrix} 0.60000000 \\ 2.32727273 \\ -0.98727273 \\ 0.87886364 \end{bmatrix}, \quad x^{(2)} = \begin{bmatrix} 1.03018182 \\ 2.03693802 \\ -1.0144562 \\ 0.98434122 \end{bmatrix}, \quad x^{(3)} = \begin{bmatrix} 1.00658504 \\ 2.00355502 \\ -1.00252738 \\ 0.99835095 \end{bmatrix}. \tag{9.89}$$

We could also put some weight in the diagonal part of A and split this into the matrices M and N of the iterative method (9.82) to try to accelerate convergence. Specifically, for some $\omega > 0$ we can write

$$\mathrm{diag}(A) = \frac{1}{\omega}\mathrm{diag}(A) - \frac{1-\omega}{\omega}\mathrm{diag}(A), \tag{9.90}$$

where the first term and the second term of the right hand side go into M and N parts of A for the Gauss-Seidel iteration. This weighted iterative method can be written as

for $k = 0, 1, \ldots$

$$x_i^{(k+1)} = \frac{a_{ii}x_i^{(k)} - \omega\left[\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} + \sum_{j=i}^{n} a_{ij}x_j^{(k)} - b_i\right]}{a_{ii}}, \; i = 1, 2, ..., n. \tag{9.91}$$

Note that $\omega = 1$ corresponds to Gauss-Seidel's method. Iteration (9.91) is generically called *SOR* (successive over-relaxation), even though we refer to over-relaxation only when $\omega > 1$ and under-relaxation when $\omega < 1$. We will see (Theorem 9.8) that a necessary condition for convergence of the SOR method is that $0 < \omega < 2$.

9.7.2 Convergence of Linear Iterative Methods

To study convergence of iterative methods of the form

$$Mx^{(k+1)} = Nx^{(k)} + b, \; k = 0, 1, \ldots$$

we use the equivalent iteration

$$x^{(k+1)} = Tx^{(k)} + c, \;\; k = 0, 1, \ldots \tag{9.92}$$

where

$$T = M^{-1}N = I - M^{-1}A \tag{9.93}$$

is called the iteration matrix and $c = M^{-1}b$.

The issue of convergence is that of existence of a *fixed point* for the map $F(x) = Tx + c$ defined for all $x \in \mathbb{R}^n$. That is, whether or not there is an $x \in \mathbb{R}^n$ such that $F(x) = x$. For if the sequence defined in (9.92) converges to a vector x then, by continuity of F, we would have $x = Tx + c = F(x)$. For any $x, y \in \mathbb{R}^n$ and for any induced matrix norm we have

$$\|F(x) - F(y)\| = \|Tx - Ty\| \leq \|T\| \; \|x - y\|. \tag{9.94}$$

If for some induced norm $\|T\| < 1$, F is a *contracting map or contraction* and we will show that this guarantees the existence of a unique fixed point. We will also show that the rate of convergence of the sequence generated by iterative methods of the form (9.92) is given by the spectral radius $\rho(T)$ of the iteration matrix T. These conclusions will follow from the following result.

Theorem 9.4. *Let T be an $n \times n$ matrix. Then the following statements are equivalent:*

(a) $\lim_{k\to\infty} T^k = 0$.

(b) $\lim_{k\to\infty} T^k x = 0$ *for all* $x \in \mathbb{R}^n$.

(c) $\rho(T) < 1$.

(d) $\|T\| < 1$ *for at least one induced norm.*

Proof. (a) $\Rightarrow$ (b): For any induced norm we have that

$$\|T^k x\| \leq \|T^k\| \; \|x\| \tag{9.95}$$

and so if $T^k \to 0$ as $k \to \infty$ then $\|T^k\| \to 0$ as $k \to \infty$ and consequently $\|T^k x\| \to 0$. That is, $T^k x \to 0$ for all $x \in \mathbb{R}^n$.

(b) $\Rightarrow$ (c): Let us suppose that $\lim_{k\to\infty} T^k x = 0$ for all $x \in \mathbb{R}^n$ but that $\rho(T) \geq 1$. Then, there is a eigenvector v such that $Tv = \lambda v$ with $|\lambda| \geq 1$ and the sequence $T^k v = \lambda^k v$ does not converge, which is a contradiction.

(c) $\Rightarrow$ (d): By Theorem 8.5, for each $\epsilon > 0$, there is at least one induced norm $\|\cdot\|$ such that $\|T\| \leq \rho(T) + \epsilon$ from which the statement follows.

(d) $\Rightarrow$ (a): This follows immediately from $\|T^k\| \leq \|T\|^k$. □

Theorem 9.5. *The iterative method (9.92) is convergent for any initial guess $x^{(0)}$ if and only if $\rho(T) < 1$ or equivalently if and only if $\|T\| < 1$ for at least one induced norm.*

Proof. Let x be the exact solution of $Ax = b$. Then

$$\begin{aligned}
x - x^{(1)} &= Tx + c - (Tx^{(0)} + c) = T(x - x^{(0)}), \\
x - x^{(2)} &= Tx + c - (Tx^{(1)} + c) = T(x - x^{(1)}) = T^2(x - x^{(0)}), \\
&\vdots \\
x - x^{(k)} &= Tx + c - (Tx^{(k-1)} + c) = T(x - x^{(k-1)}) = \ldots = T^k(x - x^{(0)}).
\end{aligned}$$

That is, the error of the k iterate, $e_k = x - x^{(k)}$, satisfies

$$e_k = T^k e_0, \tag{9.96}$$

for $k = 1, 2, \ldots$. Here, $e_0 = x - x^{(0)}$ is the error of the initial guess. The conclusion now follows immediately from Theorem 9.4. □

The spectral radius $\rho(T)$ of the iteration matrix T measures the rate of convergence of the method. For if T is normal, then $\|T\|_2 = \rho(T)$ and from (9.96) we get

$$\|e_k\|_2 \leq \rho(T)^k \|e_0\|_2. \tag{9.97}$$

But for each k we can find a vector e_0 for which the equality holds so $\rho(T)^k \|e_0\|_2$ is a least upper bound for the error $\|e_k\|_2$. If T is not normal, the following result shows that, asymptotically $\|T^k\| \approx \rho(T)^k$, for any matrix norm.

Theorem 9.6. *Let T be any $n \times n$ matrix. Then, for any matrix norm $\|\cdot\|$*

$$\lim_{k\to\infty} \|T^k\|^{1/k} = \rho(T). \tag{9.98}$$

Proof. We know that $\rho(T^k) = \rho(T)^k$ and that $\rho(T) \leq \|T\|$. Therefore

$$\rho(T) \leq \|T^k\|^{1/k}. \tag{9.99}$$

We will now show that for any given $\epsilon > 0$, $\|T^k\|^{1/k} \leq \rho(T) + \epsilon$ for all k sufficiently large, and together with (9.99) the conclusion of the theorem follows. To this effect, for any $\epsilon > 0$ we construct the auxiliary matrix $T_\epsilon = T/(\rho(T) + \epsilon)$. Then, $\lim_{k\to\infty} T_\epsilon^k = 0$ as $\rho(T_\epsilon) < 1$. Since $\|T_\epsilon^k\| \to 0$ as $k \to \infty$, there is an integer K_ϵ such that

$$\|T_\epsilon^k\| = \frac{\|T^k\|}{(\rho(T)+\epsilon)^k} \leq 1, \text{ for all } k \geq K_\epsilon. \tag{9.100}$$

Thus, for all $k \geq K_\epsilon$ we have

$$\rho(T) \leq \|T^k\|^{1/k} \leq \rho(T) + \epsilon. \tag{9.101}$$

□

Theorem 9.7. *Let A be an $n\times n$ strictly diagonally dominant matrix. Then, for any initial guess $x^{(0)} \in \mathbb{R}^n$*

(a) The Jacobi iteration converges to the exact solution of $Ax = b$.

(b) The Gauss-Seidel iteration converges to the exact solution of $Ax = b$.

Proof. (a) The Jacobi iteration matrix T has entries $T_{ii} = 0$ and $T_{ij} = -a_{ij}/a_{ii}$ for $i \neq j$. Therefore,

$$\|T\|_\infty = \max_{1\leq i\leq n} \sum_{\substack{j=1\\ j\neq i}}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| = \max_{1\leq i\leq n} \frac{1}{|a_{ii}|} \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}| < 1. \tag{9.102}$$

(b) We will proof that $\rho(T) < 1$ for the Gauss-Seidel iteration. Let x be an eigenvector of T with eigenvalue λ, normalized to have $\|x\|_\infty = 1$. Recall that $T = I - M^{-1}A$, that is $MT = M - A$. Then, $Tx = \lambda x$ implies $Mx - Ax = \lambda Mx$ from which we get

$$-\sum_{j=i+1}^{n} a_{ij}x_j = \lambda \sum_{j=1}^{i} a_{ij}x_j = \lambda a_{ii}x_i + \lambda \sum_{j=1}^{i-1} a_{ij}x_j. \tag{9.103}$$

Now choose i such that $\|x\|_\infty = |x_i| = 1$. Then,

$$|\lambda|\,|a_{ii}| \leq |\lambda| \sum_{j=1}^{i-1} |a_{ij}| + \sum_{j=i+1}^{n} |a_{ij}|$$

$$|\lambda| \leq \frac{\displaystyle\sum_{j=i+1}^{n} |a_{ij}|}{\displaystyle |a_{ii}| - \sum_{j=1}^{i-1} |a_{ij}|} < \frac{\displaystyle\sum_{j=i+1}^{n} |a_{ij}|}{\displaystyle\sum_{j=i+1}^{n} |a_{ij}|} = 1,$$

where the last inequality was obtained by using that A is SDD. Thus, $|\lambda| < 1$ and so $\rho(T) < 1$. $\square$

Theorem 9.8. *A necessary condition for convergence of the SOR iteration is* $0 < \omega < 2$*.*

Proof. We will show that $\det(T) = (1-\omega)^n$ and because $\det(T)$ is equal, up to a sign, to the product of the eigenvalues of T we have that $|\det(T)| \leq \rho^n(T)$ and this implies that

$$|1 - \omega| \leq \rho(T). \tag{9.104}$$

Since $\rho(T) < 1$ is required for convergence, the conclusion follows. Now, $T = M^{-1}N$ and $\det(T) = \det(M^{-1})\det(N)$. From the definition of the SOR iteration (9.91) we get that

$$\frac{a_{ii}}{\omega} x_i^{(k+1)} + \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} = \frac{a_{ii}}{\omega} x_i^{(k)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)} + b_i. \tag{9.105}$$

Therefore, M is lower triangular with $\operatorname{diag}(M) = \frac{1}{\omega}\operatorname{diag}(A)$. Consequently, $\det(M^{-1}) = \det(\omega \operatorname{diag}(A)^{-1})$. Similarly, $\det(N) = \det((1/\omega - 1)\operatorname{diag}(A))$.

Thus,

$$\begin{aligned}\det(T) &= \det(M^{-1})\det(N)\\ &= \det(\omega\ \mathrm{diag}(A)^{-1})\det((1/\omega-1)\mathrm{diag}(A))\\ &= \det(\mathrm{diag}(A)^{-1}(1-\omega)\mathrm{diag}(A))\\ &= \det((1-\omega)I) = (1-\omega)^n.\end{aligned} \tag{9.106}$$

□

If A is positive definite SOR converges for any initial guess. However, as we will see, there are more efficient iterative methods for positive definite linear systems.

9.8 Bibliographic Notes

The solution of large linear systems of equations is a central and classical topic in numerical linear algebra. The main reference for this chapter has been the textbook by Ciarlet [Cia89].

Section 9.1 . The solution of triangular systems and its algorithmic implementation is covered in detail in [GVL13][Section 3.1].

Section 9.2 . The complex and ancient history of Gaussian elimination and the evolution of this important numerical method are beautifully recounted by Grcar [Grc11].

Section 9.3 . The *LU* and the Cholesky factorizations are discussed in any numerical methods text covering linear systems. André-Louis Cholesky, a French army officer, described his method in a manuscript [Cho05] that was only found in 2003 [BT14]. However, Choleski's work was known by his army colleague Ernest Benoit, who published it in 1924 [Ben24].

Section 9.4 . This material was adapted from [Sch89][Subsection 1.3.3] and [Cia89][Theorem 4.3.2]. A variant of the tridiagonal solver Algorithm 9.5 is the so-called Thomas method, which is a direct modification of Gaussian elimination for a tridiagonal linear system.

Section 9.5-Section 9.6 . The idea to present these examples of large linear systems is from Ciarlet's book [Cia89].

Section 9.7 . The presentation of general iterative methods was motivated by that in [Hac94]. The results on convergence are a selection of those in [Cia89, SB02]. There are several specialized books on iterative methods for linear systems, for example [Hac94, Saa03, Gre97]

Chapter 10

Linear Systems of Equations II

In this chapter we consider the special case of $Ax = b$ when A is a symmetric, positive definite matrix. We will show first that this problem is equivalent to an optimization problem for a quadratic function, whose gradient is $Ax - b$, and will look at two gradient-based methods to solve it that are preferable to direct methods when A is sparse.

10.1 Positive Definite Linear Systems as an Optimization Problem

Suppose that A is an $n \times n$ symmetric, positive definite matrix and we are interested in solving the linear system $Ax = b$, given $b \in \mathbb{R}^n$. Henceforth, we are going to assume that the eigenvalues of A are ordered from smallest to largest

$$0 < \lambda_1 \leq \lambda_2 \leq \ldots \lambda_n.$$

Let x^* be the unique, exact solution of $Ax = b$. Since A is positive definite, we can define the norm

$$\|x\|_A = \sqrt{x^T A x} = \sqrt{\langle x, Ax \rangle}, \tag{10.1}$$

where we used the inner product notation $\langle x, y \rangle = x^T y$ in the last equality. We are now going to consider the following quadratic function of $x \in \mathbb{R}^n$

$$J(x) = \frac{1}{2} \|x - x^*\|_A^2. \tag{10.2}$$

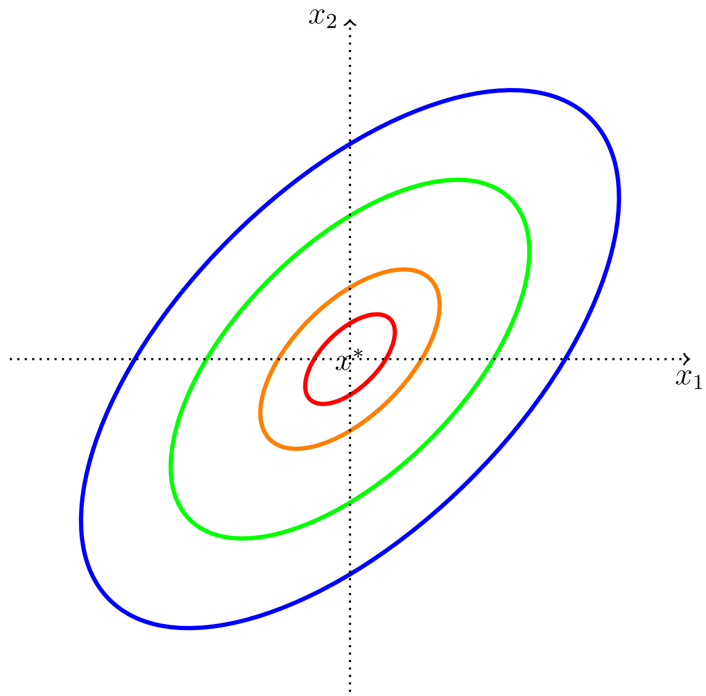

Figure 10.1: Levels set of J in 2 dimensions.

Note that $J(x) \geq 0$ and $J(x) = 0$ if and only if $x = x^*$ because A is positive definite. Therefore, x minimizes J if and only if $x = x^*$. If A is a multiple of the identity, then the level sets of J are circles centered at x^*. Otherwise, the level sets of J are n-dimensional ellipsoids with center at x^* and with axes aligned with the orthogonal eigenvectors $u_1, u_2, \ldots, u_n$ of A, as shown in Fig. 10.1 for the two dimensional case. The axis corresponding to the u_j eigenvector has a semi-length equal to $1/\sqrt{\lambda_j}$.

In optimization, the function to be minimized (maximized), J in our case, is called the *objective function*. For several optimization methods it is useful to consider the one-dimensional problem of minimizing the objective function along a fixed direction. For given x and a direction $v \neq 0$, both in $\mathbb{R}^n$, we consider the *line minimization* problem consisting in minimizing J along the line that passes through x and is in the direction of v, i.e.

$$\min_{t \in \mathbb{R}} J(x + tv). \tag{10.3}$$

Denoting $g(t) = J(x + tv)$ and using the definition (10.2) of J we get

$$
\begin{aligned}
g(t) &= \frac{1}{2}\langle x - x^* + tv, A(x - x^* + tv)\rangle \\
&= J(x) + \langle x - x^*, Av\rangle\, t + \frac{1}{2}\langle v, Av\rangle\, t^2 \qquad (10.4) \\
&= J(x) + \langle Ax - b, v\rangle\, t + \frac{1}{2}\langle v, Av\rangle\, t^2.
\end{aligned}
$$

This is a parabola opening upward because $\langle v, Av\rangle > 0$ for all $v \neq 0$. Thus, its minimum is given by the critical point

$$
0 = g'(t^*) = -\langle v, b - Ax\rangle + t^*\langle v, Av\rangle, \tag{10.5}
$$

that is

$$
t^* = \frac{\langle v, b - Ax\rangle}{\langle v, Av\rangle}. \tag{10.6}
$$

Plugging this value in (10.4) we obtain that the minimum of J along the line $x + tv$, $t \in \mathbb{R}$ is

$$
g(t^*) = J(x) - \frac{1}{2}\frac{\langle v, b - Ax\rangle^2}{\langle v, Av\rangle}. \tag{10.7}
$$

Thus, the value of $J(x)$ is decreased unless $b - Ax = 0$, i.e. x minimizes J if and only if $x = x^*$.

Finally, we note that, using the definition of $\|\cdot\|_A$ and $Ax^* = b$, we have

$$
J(x) = \frac{1}{2}\|x - x^*\|_A^2 = \frac{1}{2}x^T Ax - b^T x + \frac{1}{2}\|x^*\|_A^2 \tag{10.8}
$$

and so it follows that

$$
\nabla J(x) = Ax - b. \tag{10.9}
$$

10.2 Line Search Methods

We just saw in the previous section that the problem of solving $Ax = b$, when A is a symmetric positive definite matrix is equivalent to a convex, minimization problem of a quadratic objective function $J(x) = \frac{1}{2}\|x - x^*\|_A^2$.

An important class of methods for this type of optimization problems is called *line search methods*.

Line search methods produce a sequence of approximations to the minimizer x^* in the form

$$x^{(k+1)} = x^{(k)} + t_k v^{(k)}, \quad k = 0, 1, \ldots, \tag{10.10}$$

where the vector $v^{(k)}$ and the scalar t_k are called the *search direction* and the *step length* at the k-th iteration, respectively. Then, the central question is how to select the search directions and the step lengths to converge effectively to the minimizer. Most line search methods are of *descent* type because they require that the value of J is decreased with each iteration. Going back to (10.4) this means that descent line search methods must satisfy the condition

$$\langle \nabla J(x^{(k)}), v^{(k)} \rangle < 0, \tag{10.11}$$

which guarantees a decrease of J for sufficiently small step length t_k.

Starting with an initial guess $x^{(0)}$, line search methods generate

$$x^{(1)} = x^{(0)} + t_0 v^{(0)}, \tag{10.12}$$

$$x^{(2)} = x^{(1)} + t_1 v^{(1)} = x^{(0)} + t_0 v^{(0)} + t_1 v^{(1)}, \tag{10.13}$$

etc., so that the k-th element of the sequence is $x^{(0)}$ plus a linear combination of $v^{(0)}, v^{(1)}, \ldots, v^{(k-1)}$:

$$x^{(k)} = x^{(0)} + t_0 v^{(0)} + t_1 v^{(0)} + \cdots + t_{k-1} v^{(k-1)}. \tag{10.14}$$

That is,

$$x^{(k)} \in x^{(0)} + \operatorname{span}\{v^{(0)}, v^{(1)}, \ldots, v^{(k-1)}\}. \tag{10.15}$$

Unless otherwise noted, we will take the step length t_k to be given by the one-dimensional minimizer (10.6) evaluated at the k-step, i.e.

$$t_k = \frac{\langle v^{(k)}, r^{(k)} \rangle}{\langle v^{(k)}, A v^{(k)} \rangle}, \tag{10.16}$$

where

$$r^{(k)} = b - A x^{(k)} \tag{10.17}$$

is the *residual* of the linear equation $Ax = b$ associated with the approximation $x^{(k)}$.

10.2.1 Steepest Descent

One way to satisfy the descent condition (10.11) is to choose $v^{(k)} = -\nabla J(x^{(k)})$, which is locally the fastest rate of decrease of J. Recalling that $\nabla J(x^{(k)}) = -r^{(k)}$, we take $v^{(k)} = r^{(k)}$. The optimal step length is selected according to (10.16) so that we choose the line minimizer (in the direction of $-\nabla J(x^{(k)})$) of J. The resulting method is called *steepest descent* and, starting from an initial guess $x^{(0)}$, it can be written as follows

$$\text{for } k = 0,1,\ldots$$

$$t_k = \frac{\langle r^{(k)}, r^{(k)}\rangle}{\langle r^{(k)}, Ar^{(k)}\rangle}, \tag{10.18}$$

$$x^{(k+1)} = x^{(k)} + t_k r^{(k)}, \tag{10.19}$$

$$r^{(k+1)} = r^{(k)} - t_k A r^{(k)}. \tag{10.20}$$

Formula (10.20), which comes from subtracting A times (10.19) to b, is preferable to using the definition of the residual, i.e. $r^{(k+1)} = b - Ax^{(k)}$, due to round-off errors. The iteration can be stopped when $\|r^{(k)}\|$ is smaller than a prescribed tolerance.

Note that the method only employs the product of the matrix A and a (residual) vector r. If the matrix A is sparse, the computation of Ar at every iteration is economical (by avoiding the zero entries of A, which are the majority). Thus, rather than inputing the matrix A in the steepest descent algorithm, a procedure should be implemented for computing Ar efficiently when A is sparse. For example, for the linear system (9.79) to find an approximation of Poisson's equation, it only takes 5n operations to compute Ar for any $r \in \mathbb{R}^n$, instead of the $O(n^2)$ operation needed when A is a full, dense matrix.

Let us go back to the line minimization problem. Consider the function $g(t) = J(x^{(k)} + tr^{(k)}$, for $t \in \mathbb{R}$. Then, by the Chain Rule

$$g'(t) = \langle \nabla J(x^{(k)} + tr^{(k)}), r^{(k)}\rangle. \tag{10.21}$$

But $g'(t_k) = 0$ and consequently

$$0 = \langle \nabla J(x^{(k+1)}), r^{(k)}\rangle = -\langle r^{(k+1)}, r^{(k)}\rangle. \tag{10.22}$$

That is, consecutive residuals (gradients) are orthogonal.

Now, $J(x^{k+1}) = J(x^{(k)} + t_k r^{(k)})$ and from (10.7) it follows that

$$J(x^{k+1}) = J(x^{(k)}) - \frac{1}{2}\frac{\langle r^{(k)}, r^{(k)}\rangle^2}{\langle r^{(k)}, Ar^{(k)}\rangle}. \tag{10.23}$$

Note that $r^{(k)} = b - Ax^{(k)} = A(x^* - x^{(k)})$ so that $x^{(k)} - x^* = -A^{-1}r^{(k)}$ and consequently

$$\|x^{(k)} - x^*\|_A^2 = \langle r^{(k)}, A^{-1}r^{(k)}\rangle. \tag{10.24}$$

Therefore, we can rewrite (10.23) as

$$J(x^{k+1}) = \left[1 - \frac{\langle r^{(k)}, r^{(k)}\rangle^2}{\langle r^{(k)}, Ar^{(k)}\rangle\langle r^{(k)}, A^{-1}r^{(k)}\rangle}\right] J(x^{(k)}). \tag{10.25}$$

Clearly, if we can get a bound for the term in brackets by a number less than one, convergence will follow. The next lemma will give us that bound.

Lemma 10.2.1 (Kantorovich Inequality)**.** *Let A be a symmetric, positive definite matrix. Then, for any unit vector x,*

$$\langle x, Ax\rangle\langle x, A^{-1}x\rangle \leq \frac{1}{4}\frac{(\lambda_1 + \lambda_n)^2}{\lambda_1\lambda_n}, \tag{10.26}$$

where λ_1 and λ_n are the smallest and largest eigenvalues of A, respectively.

Proof. Since A is symmetric, positive definite $A = QDQ^T$, where Q is an orthogonal matrix and D is a diagonal matrix. Set $y = Q^Tx$. Note that y is also a unit vector and

$$\langle x, Ax\rangle\langle x, A^{-1}x\rangle = \left(\sum_{j=1}^{n} \lambda_j y_j^2\right)\left(\sum_{j=1}^{n} \lambda_j^{-1} y_j^2.\right). \tag{10.27}$$

We are going to assume $\lambda_1 < \lambda_n$ (otherwise all the eigenvalues are equal and the inequality holds trivially as an equality). The estimate (10.26) can be done directly by noting that it is possible to write

$$\lambda_j = u_j\lambda_1 + v_j\lambda_n, \tag{10.28}$$
$$\lambda_j^{-1} = u_j\lambda_1^{-1} + v_j\lambda_n^{-1} \tag{10.29}$$

for unique $u_j \geq 0$ and $v_j \geq 0$, for all $j = 1, \ldots, n$. Therefore,

$$1 = \lambda_j \lambda_j^{-1} = (u_j + v_j)^2 + \frac{(\lambda_1 - \lambda_n)^2}{\lambda_1 \lambda_n} u_j v_j, \tag{10.30}$$

which implies that $u_j + v_j \leq 1$. Now, set $u = \sum_{j=1}^{n} u_j y_j^2$, $v = \sum_{j=1}^{n} v_j y_j^2$, then

$$u + v = \sum_{j=1}^{n} (u_j + v_j) y_j^2 \leq \sum_{j=1}^{n} y_j^2 = 1. \tag{10.31}$$

On the other hand,

$$\begin{aligned} \left(\sum_{j=1}^{n} \lambda_j y_j^2 \right) \left(\sum_{j=1}^{n} \lambda_j^{-1} y_j^2 \right) &= (\lambda_1 u + \lambda_n v)(\lambda_1^{-1} u + \lambda_n^{-1} v) \\ &= (u + v)^2 + \frac{(\lambda_1 - \lambda_n)^2}{\lambda_1 \lambda_n} uv. \end{aligned} \tag{10.32}$$

But

$$(u + v)^2 + \frac{(\lambda_1 - \lambda_n)^2}{\lambda_1 \lambda_n} uv = (u + v)^2 \left[1 + \frac{(\lambda_1 - \lambda_n)^2}{\lambda_1 \lambda_n} \frac{uv}{(u + v)^2}, \right] \tag{10.33}$$

and

$$\frac{uv}{(u + v)^2} \leq \frac{1}{4}. \tag{10.34}$$

Therefore, using that $u + v \leq 1$ we obtain

$$\begin{aligned} \left(\sum_{j=1}^{n} \lambda_j y_j^2 \right) \left(\sum_{j=1}^{n} \lambda_j^{-1} y_j^2 \right) &\leq 1 + \frac{1}{4} \frac{(\lambda_1 - \lambda_n)^2}{\lambda_1 \lambda_n} \\ &= \frac{1}{4} \frac{(\lambda_1 + \lambda_n)^2}{\lambda_1 \lambda_n} \end{aligned} \tag{10.35}$$

□

Going back to equation (10.25) for the decrease of J and using (10.35) we have

$$\frac{J(x^{k+1})}{J(x^{(k)})} \leq 1 - \frac{4 \lambda_1 \lambda_n}{(\lambda_1 + \lambda_n)^2} = \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1} \right)^2 = \left(\frac{\kappa_2(A) - 1}{\kappa_2(A) + 1} \right)^2, \tag{10.36}$$

where $\kappa_2(A)$ is the 2-norm condition number of A. Thus, we obtain the following bound for consecutive errors in the steepest descent method

$$\|x^{(k+1)} - x^*\|_A \leq \left(\frac{\kappa_2(A) - 1}{\kappa_2(A) + 1}\right) \|x^{(k)} - x^*\|_A \tag{10.37}$$

and therefore

$$\|x^{(k)} - x^*\|_A \leq \left(\frac{\kappa_2(A) - 1}{\kappa_2(A) + 1}\right)^k \|x^{(0)} - x^*\|_A, \tag{10.38}$$

which implies that the method converges to the exact solution x^* if A is a symmetric, positive definite matrix. If A is a multiple of the identity, $\kappa_2(A) = 1$, the method converges in just one iteration. However, *in general*, convergence can be very slow and the bound (10.38) gives an appropriate estimate of the rate of convergence in the unfavorable case $\kappa_2(A) >> 1$ and A non-diagonal. Of course, the actual rate also depends on $x^{(0)}$. The following simple example shows that even for a diagonal matrix, the steepest descent method might not converge to the exact solution in a finite number of steps.

Example 10.1. *Let*

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{10.39}$$

Then, $x^* = [0,0]^T$ *and* $J(x) = \frac{1}{2}\left(x_1^2 + 2x_2^2\right)$. *Directly evaluating* $x^{(k+1)} = x^{(k)} + t_k r^{(k)}$ *we get*

$$x_1^{(k+1)} = \frac{4(x_2^{(k)})^2}{(x_1^{(k)})^2 + 8(x_2^{(k)})^2} x_1^{(k)}, \tag{10.40}$$

$$x_2^{(k+1)} = \frac{-(x_1^{(k)})^2}{(x_1^{(k)})^2 + 8(x_2^{(k)})^2} x_2^{(k)}. \tag{10.41}$$

Therefore, if $x_1^{(0)} \neq 0$ *and* $x_2^{(0)} \neq 0$ *then* $x_1^{(k)} \neq 0$ *and* $x_2^{(k)} \neq 0$ *for all* $k \geq 1$ *and the method cannot converge to* $x^* = [0,0]^T$ *in a finite number of iterations. The approximation will zig-zag in orthogonal directions due to* (10.22). *On the other hand if either* $x_1^{(0)}$ *or* $x_2^{(0)}$ *is equal to zero the method converges in one iteration.*

10.3 The Conjugate Gradient Method

The steepest descent method uses an optimal search direction *locally* but not *globally*. As a result, it may converge very slowly to the minimizer.

A key strategy to accelerate convergence in line search methods is to widen the search space by considering previous search directions and not just the current one.

Recall that $\left(x^{(k)} - x^{(0)}\right) \in \operatorname{span}\{v^{(0)}, v^{(1)}, \ldots, v^{(k-1)}\}$. We are going to denote

$$V_k = \operatorname{span}\{v^{(0)}, v^{(1)}, \ldots, v^{(k-1)}\} \tag{10.42}$$

and write $x \in x^{(0)} + V_k$ to mean that $x = x^{(0)} + v$ with $v \in V_k$.

The key idea is to select the search directions $v^{(0)}, v^{(1)}, \ldots, v^{(k-1)}$, i.e. V_k, such that

$$x^{(k)} = \min_{x \,\in\, x^{(0)} + V_k} J(x). \tag{10.43}$$

If the search directions are linearly independent, as k increases the search space expands and thus the minimizer x^*, the solution of $Ax = b$, will be found in at most n steps, when $V_n = \mathbb{R}^n$.

Let us derive a condition for the minimizer of $J(x) = \frac{1}{2}\|x - x^*\|_A^2$ over $x^{(0)} + V_k$. Suppose $x^{(k)} \in x^{(0)} + V_k$. Then, there are scalars $c_0, c_1, \ldots, c_{k-1}$ such that

$$x^{(k)} = x^{(0)} + c_0 v^{(0)} + c_1 v^{(1)} + \cdots + c_{k-1} v^{(k-1)}. \tag{10.44}$$

For fixed $v^{(0)}, v^{(1)}, \ldots, v^{(k-1)}$, define the following function of $c_0, c_1, \ldots, c_{k-1}$

$$G(c_0, c_1, ..., c_{k-1}) := J\left(x^{(0)} + c_0 v^{(0)} + c_1 v^{(1)} + \cdots + c_{k-1} v^{(k-1)}\right). \tag{10.45}$$

Because J is a quadratic function, the minimizer of G is the critical point $c_0^*, c_1^*, ..., c_{k-1}^*$

$$\frac{\partial G}{\partial c_j}(c_0^*, c_1^*, ..., c_{k-1}^*) = 0, \quad j = 0, \ldots, k-1. \tag{10.46}$$

But by the Chain Rule

$$0 = \frac{\partial G}{\partial c_j} = \nabla J(x^{(k)}) \cdot v^{(j)} = -\langle r^{(k)}, v^{(j)} \rangle, \quad j = 0, 1, \ldots, k-1. \tag{10.47}$$

We have proved the following theorem.

Theorem 10.1. *The vector $x^{(k)} \in x^{(0)} + V^k$ minimizes $J(x) = \frac{1}{2}\|x - x^*\|_A^2$ over $x^{(0)} + V_k$, for $k = 0, 1, \ldots$ if and only if*

$$\langle r^{(k)}, v^{(j)} \rangle = 0, \quad j = 0, 1, \ldots, k-1. \tag{10.48}$$

That is, the residual $r^{(k)} = b - Ax^{(k)}$ is orthogonal to all the previous search directions $v^{(0)}, \ldots, v^{(k-1)}$.

Let us go back to one step of a line search method, $x^{(k+1)} = x^{(k)} + t_k v^{(k)}$, where t_k is given by the one-dimensional minimizer (10.16). As noted in the steepest descent method, the corresponding residual satisfies

$$r^{(k+1)} = r^{(k)} - t_k A v^{(k)}.$$

Starting with an initial guess $x^{(0)}$, we compute $r^{(0)} = b - Ax^{(0)}$ and take $v^{(0)} = r^{(0)}$. Then,

$$x^{(1)} = x^{(0)} + t_0 v^{(0)}, \tag{10.49}$$

$$r^{(1)} = r^{(0)} - t_0 A v^{(0)} \tag{10.50}$$

and

$$\langle r^{(1)}, v^{(0)} \rangle = \langle r^{(0)}, v^{(0)} \rangle - t_0 \langle v^{(0)}, Av^{(0)} \rangle = 0, \tag{10.51}$$

where the last equality follows from the definition (10.16) of t_0. Now,

$$r^{(2)} = r^{(1)} - t_1 A v^{(1)} \tag{10.52}$$

and consequently

$$\langle r^{(2)}, v^{(0)} \rangle = \langle r^{(1)}, v^{(0)} \rangle - t_1 \langle v^{(0)}, Av^{(1)} \rangle = -t_1 \langle v^{(0)}, Av^{(1)} \rangle. \tag{10.53}$$

Thus if

$$\langle v^{(0)}, Av^{(1)} \rangle = 0 \tag{10.54}$$

then $\langle r^{(2)}, v^{(0)} \rangle = 0$. Moreover, $r^{(2)} = r^{(1)} - t_1 A v^{(1)}$ from which it follows that

$$\langle r^{(2)}, v^{(1)} \rangle = \langle r^{(1)}, v^{(1)} \rangle - t_1 \langle v^{(1)}, Av^{(1)} \rangle = 0, \tag{10.55}$$

where in the last equality we have used again (10.16) for t_1. Thus, if condition (10.54) holds we can guarantee that $\langle r^{(1)}, v^{(0)} \rangle = 0$ and $\langle r^{(2)}, v^{(j)} \rangle = 0$ for $j = 0, 1$, i.e. we satisfy the conditions of Theorem 10.1 for $k = 1, 2$. This observation motivates the following definition.

Definition 10.1. *Let A be an $n \times n$ matrix. We say that two vectors $x, y \in \mathbb{R}^n$ are conjugate with respect to A if*

$$\langle x, Ay \rangle = 0. \tag{10.56}$$

We can now proceed by induction to prove the following theorem.

Theorem 10.2. *Suppose $v^{(0)}, ..., v^{(k-1)}$ are conjugate with respect to A, then for $k = 1, 2, \ldots$*

$$\langle r^{(k)}, v^{(j)} \rangle = 0, \quad j = 0, 1, \ldots, k-1. \tag{10.57}$$

Proof. Let us do induction in k. We know the statement is true for $k = 1$. Suppose

$$\langle r^{(k-1)}, v^{(j)} \rangle = 0, \quad j = 0, 1, ..., k-2. \tag{10.58}$$

Recall $r^{(k)} = r^{(k-1)} - t_{k-1} A v^{(k-1)}$, then for $j = 0, 1, \ldots, k-2$

$$\langle r^{(k)}, v^{(j)} \rangle = \langle r^{(k-1)}, v^{(j)} \rangle - t_{k-1} \langle v^{(j)}, A v^{(k-1)} \rangle = 0, \tag{10.59}$$

where the first term is zero because of the induction hypothesis and the second term is zero because the search directions are conjugate. Moreover,

$$\langle r^{(k)}, v^{(k-1)} \rangle = \langle r^{(k-1)}, v^{(k-1)} \rangle - t_{k-1} \langle v^{(k-1)}, A v^{(k-1)} \rangle = 0 \tag{10.60}$$

because of the choice (10.16) of t_{k-1}. Therefore,

$$\langle r^{(k)}, v^{(j)} \rangle = 0, \quad j = 0, 1, \ldots, k-1.$$

□

Combining Theorems 10.1 and 10.2 we get the following important conclusion.

Theorem 10.3. *If the search directions, $v^{(0)}, v^{(1)}, \ldots, v^{(k-1)}$ are conjugate with respect to A then $x^{(k)} = x^{(k-1)} + t_{k-1} v^{(k-1)}$ is the minimizer of $J(x) = \frac{1}{2} \|x - x^*\|_A^2$ over $x^{(0)} + V_k$.*

10.3.1 Generating the Conjugate Search Directions

The conjugate gradient method[1], due to Hestenes and Stiefel, is an ingenious approach for generating efficiently the set of conjugate search directions. The crucial step is to modify the negative gradient direction, $r^{(k)}$, by adding information about the previous search direction, $v^{(k-1)}$. Specifically, we start with

$$v^{(k)} = r^{(k)} + s_k v^{(k-1)}, \tag{10.61}$$

where the scalar s_k is chosen so that $v^{(k)}$ is conjugate to $v^{(k-1)}$ with respect to A, i.e.

$$0 = \langle v^{(k)}, Av^{(k-1)}\rangle = \langle r^{(k)}, Av^{(k-1)}\rangle + s_k\langle v^{(k-1)}, Av^{k-1)}\rangle \tag{10.62}$$

which gives

$$s_k = -\frac{\langle r^{(k)}, Av^{(k-1)}\rangle}{\langle v^{(k-1)}, Av^{(k-1)}\rangle}. \tag{10.63}$$

Surprisingly, this simple construction renders all the search directions conjugate and the residuals orthogonal!

Theorem 10.4.

1. $\langle r^{(i)}, r^{(j)}\rangle = 0, \quad i \neq j,$

2. $\langle v^{(i)}, Av^{(j)}\rangle = 0, \quad i \neq j.$

Proof. By the choice of t_k and s_k it follows that for $k = 0, 1, \ldots$

$$\langle r^{(k+1)}, r^{(k)}\rangle = 0, \tag{10.64}$$
$$\langle v^{(k+1)}, Av^{(k)}\rangle = 0. \tag{10.65}$$

Let us now proceed by induction. We know from (10.64) and (10.65) that $\langle r^{(1)}, r^{(0)}\rangle = 0$ and $\langle v^{(1)}, Av^{(0)}\rangle = 0$. Suppose $\langle r^{(i)}, r^{(j)}\rangle = 0$ and $\langle v^{(i)}, Av^{(j)}\rangle =$

[1] Perhaps a more appropriate name would be "the conjugate directions method".

0 holds for $0 \le j < i \le k$. We need to prove that this holds also for $0 \le j < i \le k+1$. In view of (10.64) and (10.65) we can assume $j < k$. Now,

$$\begin{aligned} \langle r^{(k+1)}, r^{(j)} \rangle &= \langle r^{(k)} - t_k A v^{(k)}, r^{(j)} \rangle \\ &= \langle r^{(k)}, r^{(j)} \rangle - t_k \langle r^{(j)}, A v^{(k)} \rangle \\ &= -t_k \langle r^{(j)}, A v^{(k)} \rangle, \end{aligned} \tag{10.66}$$

where we have used the induction hypothesis on the orthogonality of the residuals for the last equality. But $v^{(j)} = r^{(j)} + s_j v^{(j-1)}$ and so $r^{(j)} = v^{(j)} - s_j v^{(j-1)}$. Thus,

$$\begin{aligned} \langle r^{(k+1)}, r^{(j)} \rangle &= -t_k \langle v^{(j)} - s_j v^{(j-1)}, A v^{(k)} \rangle \\ &= -t_k \langle v^{(j)}, A v^{(k)} \rangle + t_k s_j \langle v^{(j-1)}, A v^{(k)} \rangle = 0 \end{aligned} \tag{10.67}$$

by the induction hypothesis. Also for $j < k$

$$\begin{aligned} \langle v^{(k+1)}, A v^{(j)} \rangle &= \langle r^{(k+1)} + s_{k+1} v^{(k)}, A v^{(j)} \rangle \\ &= \langle r^{(k+1)}, A v^{(j)} \rangle + s_{k+1} \langle v^{(k)}, A v^{(j)} \rangle \\ &= \langle r^{(k+1)}, \frac{1}{t_j} (r^{(j)} - r^{(j+1)}) \rangle \\ &= \frac{1}{t_j} \langle r^{(k+1)}, r^{(j)} \rangle - \frac{1}{t_j} \langle r^{(k+1)}, r^{(j+1)} \rangle = 0. \end{aligned} \tag{10.68}$$

□

The conjugate gradient method is completely specified by (10.10), (10.16), (10.61), (10.63). We are now going to do some algebra to get computationally better formulas for t_k and s_k.

Recall that

$$t_k = \frac{\langle v^{(k)}, r^{(k)} \rangle}{\langle v^{(k)}, A v^{(k)} \rangle}.$$

Now,

$$\begin{aligned} \langle v^{(k)}, r^{(k)} \rangle &= \langle r^{(k)} + s_k v^{(k-1)}, r^{(k)} \rangle \\ &= \langle r^{(k)}, r^{(k)} \rangle + s_k \langle v^{(k-1)}, r^{(k)} \rangle = \langle r^{(k)}, r^{(k)} \rangle, \end{aligned} \tag{10.69}$$

where we have used (10.57). Therefore,

$$t_k = \frac{\langle r^{(k)}, r^{(k)} \rangle}{\langle r^{(k)}, A r^{(k)} \rangle}. \tag{10.70}$$

Let us now work on the numerator of s_{k+1}, the inner product $\langle r^{(k+1)}, Av^{(k)}\rangle$. First, since $r^{(k+1)} = r^{(k)} - t_k Av^{(k)}$ then $t_k Av^{(k)} = r^{(k)} - r^{(k+1)}$. Thus,

$$-\langle r^{(k+1)}, Av^{(k)}\rangle = \frac{1}{t_k}\langle r^{(k+1)}, r^{(k+1)} - r^{(k)}\rangle = \frac{1}{t_k}\langle r^{(k+1)}, r^{(k+1)}\rangle. \tag{10.71}$$

And for the denominator, we have

$$\begin{aligned}
\langle v^{(k)}, Av^{(k)}\rangle &= \frac{1}{t_k}\langle v^{(k)}, r^{(k)} - r^{(k+1)}\rangle \\
&= \frac{1}{t_k}\langle v^{(k)}, r^{(k)}\rangle - \frac{1}{t_k}\langle v^{(k)}, r^{(k+1)}\rangle \\
&= \frac{1}{t_k}\langle r^{(k)} + s_k v^{(k-1)}, r^{(k)}\rangle \\
&= \frac{1}{t_k}\langle r^{(k)}, r^{(k)}\rangle + \frac{s_k}{t_k}\langle v^{(k-1)}, r^{(k)}\rangle = \frac{1}{t_k}\langle r^{(k)}, r^{(k)}\rangle.
\end{aligned} \tag{10.72}$$

Therefore, we can write

$$s_{k+1} = \frac{\langle r^{(k+1)}, r^{(k+1)}\rangle}{\langle r^{(k)}, r^{(k)}\rangle}. \tag{10.73}$$

A pseudo-code for the conjugate gradient method is given in Algorithm 10.1. The main cost per iteration of the conjugate gradient method is the evaluation of $Av^{(k)}$. As noted for the steepest descent method, the product $Av^{(k)}$ can be evaluated cheaply is A is sparse.

Theorem 10.5. *Let A be an $n\times n$ symmetric, positive definite matrix. Then, the conjugate gradient method converges to the exact solution (assuming no round-off errors) of $Ax = b$ in at most n steps.*

Proof. By Theorem 10.4, the residuals are orthogonal hence linearly independent. After n steps, $r^{(n)}$ is orthogonal to $r^{(0)}, r^{(1)}, \ldots, r^{(n-1)}$. Since the dimension of the space is n, $r^{(n)}$ has to be the zero vector. $\square$

10.3.2 Krylov Subspaces

In the conjugate gradient method we start with an initial guess $x^{(0)}$, compute the residual $r^{(0)} = b - Ax^{(0)}$ and set $v^{(0)} = r^{(0)}$. We then get $x^{(1)} = x^{(0)} + t_0 r^{(0)}$

Algorithm 10.1 The conjugate gradient method

Given $x^{(0)}$,TOL, and $kmax$, set $r^{(0)} = b - Ax^{(0)}$, $v^{(0)} = r^{(0)}$, and $k = 0$.

while $\|r^{(k)}\|_2 > TOL$ and $k \leq kmax$ **do**

$$t_k \leftarrow \frac{\langle r^{(k)}, r^{(k)} \rangle}{\langle v^{(k)}, Av^{(k)} \rangle}$$

$$x^{(k+1)} \leftarrow x^{(k)} + t_k v^{(k)}$$

$$r^{(k+1)} \leftarrow r^{(k)} - t_k A v^{(k)}$$

$$s_{k+1} \leftarrow \frac{\langle r^{(k+1)}, r^{(k+1)} \rangle}{\langle r^{(k)}, r^{(k)} \rangle}$$

$$v^{(k+1)} \leftarrow r^{(k+1)} + s_{k+1} v^{(k)}$$

$$k \leftarrow k + 1$$

end while

and evaluate the residual $r^{(1)}$, etc. Using the definition of the residual we have

$$r^{(1)} = b - Ax^{(1)} = b - Ax^{(0)} - t_0 Ar^{(0)} = r^{(0)} - t_0 Ar^{(0)} \tag{10.74}$$

so that $r^{(1)}$ is a linear combination of $r^{(0)}$ and $Ar^{(0)}$. Similarly,

$$\begin{aligned} x^{(2)} &= x^{(1)} + t_1 v^{(1)} \\ &= x^{(0)} + t_0 r^{(0)} + t_1 r^{(1)} + t_1 s_0 r^{(0)} \\ &= x^{(0)} + (t_0 + t_1 s_0) r^{(0)} + t_1 r^{(1)} \end{aligned} \tag{10.75}$$

and using (10.74) we get $x^{(2)} = x^{(0)} + c_0 r^{(0)} + c_1 Ar^{(0)}$, where $c_0 = t_0 + t_1 + s_0 t_1$ and $c_1 = -t_0 t_1$ so that $r^{(2)} = b - Ax^{(2)}$ is a linear combination of $r^{(0)}$, $Ar^{(0)}$, and $A^2 r^{(0)}$. Likewise, $r^{(3)}$ is a linear combination of $r^{(0)}$, $Ar^{(0)}$, $A^2 r^{(0)}$, and $A^3 r^{(0)}$ and so on.

Definition 10.2. *The set $\mathcal{K}_k(A, r) = span\{r, Ar, ..., A^{k-1} r\}$ is called the Krylov subspace of degree (or order) k generated by the matrix A and the vector r.*

Krylov subspaces are central to an important class of numerical methods that rely on getting approximations through matrix-vector multiplication, like the conjugate gradient method.

The following theorem provides a reinterpretation of the conjugate gradient method. The approximation $x^{(k)}$ is the minimizer of $J(x) = \frac{1}{2}\|x - x^*\|_A^2$ over $x^{(0)} + \mathcal{K}_k(A, r^{(0)})$.

Theorem 10.6. $\mathcal{K}_k(A, r^{(0)}) = span\{r^{(0)}, ..., r^{(k-1)}\} = span\{v^{(0)}, ..., v^{(k-1)}\}$.

Proof. We will prove it by induction. The case $k = 1$ holds by construction. Let us now assume that it holds for k and we will prove that it also holds for $k + 1$.

By the induction hypothesis $r^{(k-1)}, v^{(k-1)} \in \mathcal{K}_k(A, r^{(0)})$ then

$$Av^{(k-1)} \in \text{span}\{Ar^{(0)}, ..., A^k r^{(0)}\}$$

but $r^{(k)} = r^{(k-1)} - t_{k-1} Av^{(k-1)}$ and so

$$r^{(k)} \in \mathcal{K}_{k+1}(A, r^{(0)}).$$

Consequently,

$$\operatorname{span}\{r^{(0)}, ..., r^{(k)}\} \subseteq \mathcal{K}_{k+1}(A, r^{(0)}). \tag{10.76}$$

We now prove the reverse inclusion, $\operatorname{span}\{r^{(0)}, ..., r^{(k)}\} \supseteq \mathcal{K}_{k+1}(A, r^{(0)})$. Note that $A^k r^{(0)} = A(A^{k-1} r^{(0)})$ and by the induction hypothesis

$$\operatorname{span}\{r^{(0)}, Ar^{(0)}, ..., A^{k-1} r^{(0)}\} = \operatorname{span}\{v^{(0)}, ..., v^{(k-1)}\}$$

we have that

$$A^k r^{(0)} = A(A^{k-1} r^{(0)}) \in \operatorname{span}\{Av^{(0)}, ..., Av^{(k-1)}\}.$$

Moreover, since

$$Av^{(j)} = \frac{1}{t_j}(r^{(j)} - r^{(j+1)})$$

it follows that $A^k r^{(0)} \in \operatorname{span}\{r^{(0)}, r^{(1)}, ..., r^{(k)}\}$ and therefore

$$\operatorname{span}\{r^{(0)}, ..., r^{(k)}\} \supseteq \mathcal{K}_{k+1}(A, r^{(0)}). \tag{10.77}$$

Thus,

$$\operatorname{span}\{r^{(0)}, ..., r^{(k)}\} = \mathcal{K}_{k+1}(A, r^{(0)}). \tag{10.78}$$

Finally, we observe that $\operatorname{span}\{v^{(0)}, ..., v^{(k)}\} = \operatorname{span}(\{v^{(0)}, ..., v^{(k-1)}, r^{(k)}\}$ because $v^{(k)} = r^{(k)} + s_k v^{(k-1)}$ and by the induction hypothesis

$$\begin{aligned}
\operatorname{span}\{v^{(0)}, ..., v^{(k)}, r^{(k)}\} &= \operatorname{span}\{r^{(0)}, Ar^{(0)}, ..., A^k r^{(0)}, r^{(k)}\} \\
&= \operatorname{span}\{r^{(0)}, r^{(1)}, ..., r^{(k)}, r^{(k)}\} \qquad (10.79) \\
&= \mathcal{K}_{k+1}(A, r^{(0)}).
\end{aligned}$$

□

10.3.3 Convergence of the Conjugate Gradient Method

Let us define the initial error as $e^{(0)} = x^{(0)} - x^*$. Then $Ae^{(0)} = Ax^{(0)} - Ax^*$ implies that

$$r^{(0)} = -Ae^{(0)}. \tag{10.80}$$

For the conjugate gradient method $x^{(k)} \in x^{(0)} + \mathcal{K}_k(A, r^{(0)})$, i.e.

$$x^{(k)} = x^{(0)} + c_1 r^{(0)} + c_2 A r^{(0)} + \cdots + c_k A^{k-1} r^{(0)} \tag{10.81}$$

for some real constants $c_1, \ldots, c_k$. Therefore, using (10.80) we have that

$$x^{(k)} - x^* = e^{(0)} - c_1 A e^{(0)} - c_2 A^2 e^{(0)} - \cdots - c_k A^k e^{(0)}. \tag{10.82}$$

In fact,

$$\|x^{(k)} - x^*\|_A = \min_{p \in \tilde{\mathbb{P}}_k} \|p(A)e^{(0)}\|_A, \tag{10.83}$$

where $\tilde{\mathbb{P}}_k$ is the set of all polynomials of degree $\leq k$ that are equal to one at 0. Since A is symmetric, positive definite all its eigenvalues are real and positive. Let us order them as $0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, with associated orthonormal eigenvectors $v_1, v_2, \ldots v_n$. Then, we can write $e^{(0)} = \alpha_1 v_1 + \ldots \alpha_n v_n$ for some scalars $\alpha_0, \ldots, \alpha_n$ and

$$p(A)e^{(0)} = \sum_{j=1}^{n} p(\lambda_j)\alpha_j v_j. \tag{10.84}$$

Therefore,

$$\begin{aligned} \|p(A)e^{(0)}\|_A^2 = \langle p(A)e^{(0)}, Ap(A)e^{(0)} \rangle &= \sum_{j=1}^{n} p^2(\lambda_j)\lambda_j \alpha_j^2 \\ \leq \left(\max_j p^2(\lambda_j) \right) &\sum_{j=1}^{n} \lambda_j \alpha_j^2 \end{aligned} \tag{10.85}$$

and since

$$\|e^{(0)}\|_A^2 = \sum_{j=1}^{n} \lambda_j \alpha_j^2 \tag{10.86}$$

we get

$$\|x^{(k)} - x^*\|_A \leq \min_{p \in \tilde{\mathbb{P}}_k} \max_j |p(\lambda_j)| \, \|e^{(0)}\|_A. \tag{10.87}$$

If, for example, A has only $m < n$ distinct eigenvalues, then we can construct a polynomial $p \in \tilde{\mathbb{P}}_m$ that vanishes at those eigenvalues. From (10.87), it then follows that $\|x^{(m)} - x^*\|_A = 0$ and the conjugate gradient method would converge in at most m steps instead of n steps.

The min max term can be estimated using the Chebyshev polynomial T_k with the change of variables

$$f(\lambda) = \frac{2\lambda - \lambda_1 - \lambda_n}{\lambda_n - \lambda_1} \tag{10.88}$$

to map $[\lambda_1, \lambda_n]$ to $[-1, 1]$. The polynomial

$$p(\lambda) = \frac{1}{T_k(f(0))} T_k(f(\lambda)) \tag{10.89}$$

is in $\tilde{\mathbb{P}}_k$ and since $|T_k(f(\lambda))| \leq 1$

$$\max_j |p(\lambda_j)| = \frac{1}{|T_k(f(0))|}. \tag{10.90}$$

Now

$$\begin{aligned} |T_k(f(0))| &= \left| T_k\left(\frac{\lambda_1 + \lambda_n}{\lambda_n - \lambda_1}\right)\right| = \left| T_k\left(\frac{\lambda_n/\lambda_1 + 1}{\lambda_n/\lambda_1 - 1}\right)\right| \\ &= \left| T_k\left(\frac{\kappa_2(A) + 1}{\kappa_2(A) - 1}\right)\right| \end{aligned} \tag{10.91}$$

because $\kappa_2(A) = \lambda_n/\lambda_1$ is the condition number of A in the 2-norm. We use now an identity of Chebyshev polynomials, namely if $x = (z + 1/z)/2$ then $T_k(x) = (z^k + 1/z^k)/2$. Noting that

$$\frac{\kappa_2(A) + 1}{\kappa_2(A) - 1} = \frac{1}{2}(z + 1/z) \tag{10.92}$$

for

$$z = (\sqrt{\kappa_2(A)} + 1)/(\sqrt{\kappa_2(A)} - 1) \tag{10.93}$$

we obtain

$$\left| T_k\left(\frac{\kappa_2(A) + 1}{\kappa_2(A) - 1}\right)\right|^{-1} \leq 2\left(\frac{\sqrt{\kappa_2(A)} - 1}{\sqrt{\kappa_2(A)} + 1}\right)^k, \tag{10.94}$$

from which it follows that the error in the conjugate gradient method has the bound

$$\|x^{(k)} - x^*\|_A \leq 2 \left(\frac{\sqrt{\kappa_2(A)} - 1}{\sqrt{\kappa_2(A)} + 1} \right)^k \|x^{(0)} - x^*\|_A. \tag{10.95}$$

This is a similar error bound as (10.38) for the steepest descent method, except that for the conjugate gradient method the convergence rate depends on $\sqrt{\kappa_2(A)}$ instead of $\kappa_2(A)$.

10.3.4 Preconditioned Conjugate Gradient

The error bound (10.95) suggests that convergence of the conjugate gradient method can be accelerated by transforming the original linear system $Ax = b$ to one with a reduced condition number

$$C^{-1}AC^{-T}(C^T x) = C^{-1}b, \tag{10.96}$$

as mentioned in Section 8.7.1. This is called preconditioning. Note that $C^{-1}AC^{-T}$ is symmetric, positive definite, assuming A is so. We would like the matrix $\tilde{A} = C^{-1}AC^{-T}$ to have a much smaller condition number than A. Since A is symmetric, positive definite $A = LL^T$, where L is the (lower triangular) Choleski factor, and hence if $C \approx L$ the condition number of $\tilde{A}$ would be close to one. If we apply the conjugate gradient method to the preconditioned linear system (10.96), we will see that only the matrix $M = CC^T$ arises in the algorithm and not C per se. The preconditioned conjugate gradient method in pseudo-code is shown in Algorithm 10.2. Note that in Step 6 of the algorithm one needs to solve a linear system of the form $Mw = r$. Thus, for the method to be effective, solving $Mw = r$ must be significantly cheaper than solving the original system $Ax = b$.

Again, since $A = LL^T$ (A symmetric, positive definite) the idea is to take $M = CC^T$, where C is an approximation, in some sense, of the lower triangular factor L of A. The simplest choice is $M = D = \mathrm{diag}(A)$, which corresponds to one iteration of Jacobi's method. However, this is rarely an effective preconditioner. We can employ SOR but because the SOR matrix is not symmetric, we need to use a symmetrized version. Writing $A = D - L - L^T$ (now L stands for minus the lower triangular part of A with zeros on the diagonal), the symmetric SOR preconditioner corresponds to

$$M = \frac{1}{2-\omega}\left(\frac{1}{\omega}D - L\right)\left(\frac{1}{\omega}D\right)^{-1}\left(\frac{1}{\omega}D - L^T\right), \tag{10.97}$$

for some appropriate $\omega \in (0,2)$. Note that the corresponding linear system $Mw = r$ is easy to solve because is M is the product of a lower and an upper triangular matrix.

A more general and often effective preconditioner is the *incomplete Choleski factorization*, in which $C \approx L$ (L being the Choleski factor) but such that C has some structured sparsity (e.g. the same sparsity as L). One way is to achieve this is to follow the Choleski procedure (Algorithm 9.4) and set to zero l_{ij} if the corresponding $a_{ij} = 0$.

Algorithm 10.2 The preconditioned conjugate gradient method

Given $x^{(0)}$, a preconditioner M, TOL, and $kmax$, set $r^{(0)} = b - Ax^{(0)}$, $v^{(0)} = r^{(0)}$, $k = 0$, and solve $Mw^{(0)} = r^{(0)}$.

while $\|r^{(k)}\|_2 > TOL$ and $k \leq kmax$ **do**

$$t_k \leftarrow \frac{\langle r^{(k)}, w^{(k)}\rangle}{\langle v^{(k)}, Av^{(k)}\rangle}$$

$$x^{(k+1)} \leftarrow x^{(k)} + t_k v^{(k)}$$

$$r^{(k+1)} \leftarrow r^{(k)} - t_k A v^{(k)}$$

$$Mw^{(k+1)} \leftarrow r^{(k+1)}$$

$$s_{k+1} \leftarrow \frac{\langle r^{(k+1)}, w^{(k+1)}\rangle}{\langle r^{(k)}, w^{(k)}\rangle}$$

$$v^{(k+1)} \leftarrow w^{(k+1)} + s_{k+1} v^{(k)}$$

$$k \leftarrow k+1$$

end while

10.4 Bibliographic Notes

Section 10.1 . The equivalence of $Ax = b$ to an optimization problem, $\min J(x)$, when A is a symmetric, positive definite matrix appears in most textbooks on numerical linear algebra and optimization (e.g. [NW06, LY08, Cia89]). But typically $J(x)$ is taken to be $\frac{1}{2}\langle x, Ax\rangle - \langle b, x\rangle$. We find it more natural to use instead the square A-norm of the error, $J(x) = \frac{1}{2}\|x - x^*\|_A^2$, which differs from the usual J by a just constant [see (10.8)].

Section 10.2 . The presentation of line search methods and in particular of steepest descent follows that in Nocedal and Wright's book [NW06]. There are many proofs (and versions) of Kantarovich inequality. The one presented here is due to Henrici [Hen61]. The estimate of the bound for the error of the steepest descent follows the derivation in the excellent optimization book by Luenberger and Ye [LY08]. Example 10.1 is a special case of one in [Cia89].

Section 10.3 . Hestenes and Stiefel proposed the conjugate gradient method in 1952 [HS52]. The presentation and motivation of the method as a subspace expanding minimization given here were inspired by Section 5.1 in [NW06]. The proof of the error bound and the discussion of preconditioning follows that in [SB02]. The topic of preconditioning is a vast one (see for example [BBC+94]). Here, we simply presented the main idea in the context of the conjugate gradient method. More details of incomplete Choleski factorizations can be found in [GVL13].

Chapter 11

Eigenvalue Problems

In this chapter we take a brief look at two numerical methods for the standard eigenvalue problem, i.e. given a square matrix A find scalars λ (eigenvalues) and non-zero vectors v (eigenvectors) such that $Av = \lambda v$. Eigenvalue problems appear in many applications areas, for example in stability analysis of differential equations, quantum mechanics, pattern recognition, search engines, and data analysis. One method is a simple iteration for finding a dominant eigenvalue of a matrix (the power method) and the other is a much more expensive iteration for finding all the eigenvalues of a general matrix (the QR method). Both iterations can be accelerated by doing a suitable (inverse) shift. Special orthogonal transformations, known as Householder reflections play an important role in several of the most commonly used eigenvalue and SVD methods. Thus, we devote a section to them and their use to obtain the QR factorization and in the QR method for eigenvalues itself.

11.1 The Power Method

Suppose that A has a dominant eigenvalue:

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n| \tag{11.1}$$

and a complete set of eigenvectors $v_1, \ldots, v_n$ associated to $\lambda_1, \ldots, \lambda_n$, respectively (i.e. A is diagonalizable). Then, any vector $u_0 \in \mathbb{R}^n$ can be written in terms of the eigenvectors as

$$u_0 = c_1 v_1 + \cdots + c_n v_n \tag{11.2}$$

and

$$A^k u_0 = c_1 \lambda_1^k v_1 + \cdots + c_n \lambda_1^k v_n = c_1 \lambda_1^k \left[v_1 + \sum_{j=2}^{n} \frac{c_j}{c_1} \left(\frac{\lambda_j}{\lambda_1} \right)^k v_j \right]. \qquad (11.3)$$

Assuming $c_1 \neq 0$, it follows that $A^k u_0 / c_1 \lambda_1^k \to v_1$ and we get a method to approximate the eigenpair λ_1, v_1 that is equivalent to the iteration

$$u_k = A u_{k-1}, \quad k = 1, 2, \ldots \qquad (11.4)$$

To avoid overflow, we normalize the approximating vector at each iteration as Algorithm 11.1 shows. The dominant eigenvalue λ_1 is then approximated by $u^T A u$, where u is the normalized approximation to v_1 at the k-th iteration.

Algorithm 11.1 The Power Method

Set $k = 0$, $\|r\|_2 >> 1$.
while $\|r\|_2 > TOL$ **do**
  $u \leftarrow Au$
  $u \leftarrow u / \|u\|_2$
  $\lambda \leftarrow u^T A u$
  $r = Au - \lambda u$
  $k \leftarrow k + 1$
end while

The rate of convergence of the power method is determined by the ratio λ_2/λ_1. From (11.3), it follows that

$$|\lambda^{(k)} - \lambda_1| = O(|\lambda_2/\lambda_1|^k), \qquad (11.5)$$

where $\lambda^{(k)}$ is the approximation to λ_1 at the k-th iteration.

Example 11.1. *Let us apply the power method to the matrix*

$$A = \begin{bmatrix} 196 & -24 & -30 & -12 \\ 72 & 112 & -60 & -24 \\ 48 & 48 & 10 & -36 \\ 24 & 24 & 0 & -8 \end{bmatrix}. \qquad (11.6)$$

The eigenvalues of this matrix are $160, 100, 40, 10$. *Table 11.1 shows the approximation to the dominant eigenvalue,* $\lambda_1 = 160$, *for a few iterations* *starting from* $u_0 = [1,0,0,0]^T$. *The corresponding relative error of the k-th approximation* $\lambda^{(k)}$ *and the decrease factor* $|\lambda^{(k+1)} - \lambda_1|/|\lambda^{(k)} - \lambda_1|$ *are also shown. Note the slow convergence of the method. The decrease factor is asymptotically approaching* $\lambda_2/\lambda_1 = 100/160 = 0.625$ *in accordance to* (11.5). *The approximate eigenvector after 10 iterations is*

$$u^{(10)} = [0.73224283, 0.5460497, 0.36403314, 0.18201657]^T.$$

The exact, normalized eigenvector is $v_1 = \frac{1}{\sqrt{30}}[4,3,2,1]^T$ *and thus the error measured in the 2-norm is* $\|v_1 - u^{(10)}\|_2 \approx 0.0029$.

Table 11.1: The power method for the matrix A in (11.6) and with initial vector $u_0 = [1,0,0,0]^T$.

k	$\lambda^{(k)}$	$\frac{\lvert\lambda^{(k)}-\lambda_1\rvert}{\lambda_1}$	$\frac{\lvert\lambda^{(k+1)}-\lambda_1\rvert}{\lvert\lambda^{(k)}-\lambda_1\rvert}$
1	192.530120	0.203313	
2	178.984591	0.118654	0.583600
3	171.076488	0.069228	0.583446
4	166.607851	0.041299	0.596566
5	164.009151	0.025057	0.606725
6	162.459449	0.015372	0.613459
7	161.519338	0.009496	0.617755
8	160.942693	0.005892	0.620463
9	160.586507	0.003666	0.622161
10	160.365526	0.002285	0.623225

As in all iterative methods, convergence is also dependent on the initial guess. Let us take now $u_0 = [1,1,1,1]^T$. *Table 11.2 shows the much faster convergence of the power method starting with this* u_0. *More than 12 digits of accuracy are obtained for the approximation of* λ_1 *in just 10 iterations; the decrease factor is 10 times smaller than* λ_2/λ_1. *How can this be explained? A calculation reveals that the eigenvectors associated to the eigenvalues* 160, 100, 40, *and* 10 *of* (11.6) *are, without normalization,* $v_1 = [4,3,2,1]^T$, $v_2 = [3/2,3,2,1]^T$, $v_3 = [1,2,3,3/2]^T$, $v_4 = [1,2,3,4]^T$, *respectively. Thus,* $u_0 = \frac{1}{5}v_1 + \frac{1}{5}v_4$, *i.e.* $c_2 = c_3 = 0$ *in* (11.3) *and hence the power method iteration converges at a rate* $\lambda_4/\lambda_1 = 0.0625$ *instead of at the typical rate* λ_2/λ_1.

The power method is useful and efficient for computing the dominant eigenpair λ_1, v_1 when A is sparse, so that the evaluation of Av is economical,

Table 11.2: The power method for the matrix A in (11.6) and with initial vector $u_0 = [1, 1, 1, 1]^T$.

k	$\lambda^{(k)}$	$\frac{\lvert\lambda^{(k)}-\lambda_1\rvert}{\lambda_1}$	$\frac{\lvert\lambda^{(k+1)}-\lambda_1\rvert}{\lvert\lambda^{(k)}-\lambda_1\rvert}$
1	153.7125748503	0.039296	
2	159.6091279379	0.002443	0.06216727
3	159.9755849459	0.000153	0.06246303
4	159.9984741172	9.536767×10^{-6}	0.06249762
5	159.9999046326	5.960465×10^{-7}	0.06249985
6	159.9999940395	3.725290×10^{-8}	0.06249999
7	159.9999996275	2.328306×10^{-9}	0.06250000
8	159.9999999767	1.455192×10^{-10}	0.06250000

and when $|\lambda_2/\lambda_1| << 1$. To present the method we have assumed that λ_1 is a simple eigenvalue and that A is diagonalizable so that (11.3) is valid. However, a similar argument can be given when λ_1 has multiplicity greater than one. Moreover, the method can also be applied when A is not diagonalizable but has a dominant eigenvalue.

We have also assumed that u_0 is chosen such that $c_1 \neq 0$ in (11.3). In theory, if $c_1 = 0$ the method would converge to another eigenvalue (e.g. λ_2 if $c_2 \neq 0$) and not to λ_1. However, due to roundoff errors $c_1 = O(\text{eps})$ and so the method will eventually converge toward the dominant eigenpair λ_1, v_1.

We can use shifts in the matrix A to decrease $|\lambda_2/\lambda_1|$ and improve convergence. We apply the power method with the shifted matrix $A - sI$, where the shift s is chosen to accelerate convergence. For example, for the matrix A in (11.6), with eigenvalues $160, 100, 40, 10$, the matrix $A - 50I$ has eigenvalues $110, 50, -10, -40$ and the power method would converge at a rate of $50/110 = 0.4545...$ instead of at the rate $100/160 = 0.625$.

A variant of the shift power method is the inverse power method, which applies the iteration to the matrix $(A - \tilde{\lambda}I)^{-1}$, where $\tilde{\lambda}$ is an approximation to one of the eigenvalues of A. Let us suppose $\tilde{\lambda} \approx \lambda_i$ in the sense that

$$0 < |\lambda_i - \tilde{\lambda}| << |\lambda_j - \tilde{\lambda}|, \quad \text{for all } j \neq i. \tag{11.7}$$

Then, $(A - \tilde{\lambda}I)^{-1}$, whose eigenvalues are $1/(\lambda_j - \tilde{\lambda})$, $j = 1, \ldots, n$ has a dominant eigenvalue:

$$\left|\frac{1}{\lambda_i - \tilde{\lambda}}\right| >> \left|\frac{1}{\lambda_j - \tilde{\lambda}}\right| \quad \text{for all } j \neq i. \tag{11.8}$$

Assuming A has a complete set of eigenvectors $v_1, \ldots, v_n$, we can write as before $u_0 = c_1 v_1 + \ldots + c_n v_n$ and iterate

$$u_k = (A - \tilde{\lambda} I)^{-1} u_{k-1}, \quad k = 1, 2, \ldots \tag{11.9}$$

Thus,

$$u_k = [(A - \tilde{\lambda} I)^{-1}]^k u_0 = \frac{1}{(\lambda_i - \tilde{\lambda})^k} \left[c_i v_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(\frac{\lambda_i - \tilde{\lambda}}{\lambda_j - \tilde{\lambda}} \right)^k c_j v_j \right]. \tag{11.10}$$

Consequently, $(\lambda_i - \tilde{\lambda})^k u_k / c_i \to v_i$ at a rate given by $\max_{j \neq i} \left| \frac{\lambda_i - \tilde{\lambda}}{\lambda_j - \tilde{\lambda}} \right|$.

Unless A is of small size, the inverse matrix is not actually computed. Instead, the linear system

$$(A - \tilde{\lambda} I) u^{(k)} = u^{(k-1)} \tag{11.11}$$

is solved at every iteration. The method will converge to the eigenvalue λ_i for which $\tilde{\lambda}$ is a good approximation.

Example 11.2. *Considering again the matrix A in* (11.6)*, we let $\tilde{\lambda} = 37$ and apply the inverse power method with initial vector $u_0 = [1, -1, -1, 1]$. We obtain the approximations shown in Table 11.3. The method converges to the eigenpair λ_3, v_3 at a rate that is approaching $|\lambda_3 - \tilde{\lambda}| / |\lambda_4 - \tilde{\lambda}| \approx 0.11$.*

11.2 Householder QR Factorization

One of the most general methods for finding eigenvalues is the QR method, which makes repeated use of QR factorizations (see Section 8.5). Hence, we begin by presenting first a numerically stable method to obtain the QR factorization of an $m \times n$ matrix A. This is the method of Householder, which is based on the observation that one can reduce A to an upper triangular form by applying a sequence of elementary, orthogonal transformations of the type made precise in the following definition.

Table 11.3: The inverse power method for the matrix A in (11.6) with initial vector $u_0 = [1, -1, -1, 1]^T$ and $\tilde{\lambda} = 37$ ($\lambda_i = 40$).

k	$\lambda^{(k)}$	$\frac{\lvert\lambda^{(k)}-\lambda_i\rvert}{\lambda_i}$	$\frac{\lvert\lambda^{(k+1)}-\lambda_i\rvert}{\lvert\lambda^{(k)}-\lambda_i\rvert}$
1	39.7434832246	0.006413	
2	40.0062323469	0.000156	0.024296
3	39.9982970629	4.257343×10^{-5}	0.273241
4	40.0001424865	3.562162×10^{-6}	0.083671
5	39.9999819781	4.505464×10^{-7}	0.126481
6	40.0000018990	4.747521×10^{-8}	0.105373
7	39.9999997841	5.397610×10^{-9}	0.113693
8	40.0000000238	5.939102×10^{-10}	0.110032

Definition 11.1. *Let $v \in \mathbb{R}^n$, $v \neq 0$, a Householder reflection is an $n \times n$ matrix of the form*

$$P = I - 2\frac{vv^T}{\langle v, v\rangle}. \tag{11.12}$$

Note that P is a symmetric and orthogonal matrix, i.e. $P^T = P$ and $P^T P = I$. Orthogonal matrices preserve the 2-norm:

$$\langle Pu, Pu\rangle = \langle P^T Pu, u\rangle = \langle u, u\rangle. \tag{11.13}$$

Thus,

$$\|Pu\| = \|u\|. \tag{11.14}$$

Moreover, $Pv = -v$ and $Pu = u$ for all u orthogonal to v. Therefore, Pu may be interpreted as the reflection of u across the hyperplane with normal v, i.e. the span$\{v\}^\perp = \{w \in \mathbb{R}^n : \langle v, w\rangle = 0\}$. Since the eigenvalues of P are 1 (with eigenvectors in span$\{v\}$, with multiplicity -1) and 1 (with eigenvectors in span$\{v\}^\perp$, with multiplicity $n-1$) the determinant of P is -1.

The central idea is to find a Householder reflection than turns a given, nonzero vector $a \in \mathbb{R}^n$ into a multiple of $e_1 = [1, 0, \dots, 0] \in \mathbb{R}^n$. That is, we want v such that $Pa = \gamma e_1$, for some $\gamma \in \mathbb{R}$, which implies $v \in \text{span}\{e_1, a\}$. Writing $v = a + \alpha e_1$ we have

$$Pa = a - 2\frac{\langle v, a\rangle}{\langle v, v\rangle}v = \left[1 - 2\frac{\langle v, a\rangle}{\langle v, v\rangle}\right] a - 2\alpha\frac{\langle v, a\rangle}{\langle v, v\rangle}e_1. \tag{11.15}$$

Thus, we need

$$2\frac{\langle v, a\rangle}{\langle v, v\rangle} = 1. \tag{11.16}$$

But

$$\langle v, a\rangle = \langle a, a\rangle + \alpha a_1, \tag{11.17}$$
$$\langle v, v\rangle = \langle a, a\rangle + 2\alpha a_1 + \alpha^2, \tag{11.18}$$

where a_1 is the first component of a. Consequently, (11.16) implies

$$2\left[\langle a, a\rangle + \alpha a_1\right] = \langle a, a\rangle + 2\alpha a_1 + \alpha^2. \tag{11.19}$$

from which it follows that $\alpha^2 = \langle a, a\rangle$. Therefore, $\alpha = \pm\|a\|$ and

$$v = a \pm \|a\| e_1, \tag{11.20}$$
$$Pa = \mp \|a\| e_1. \tag{11.21}$$

Note that we have a choice of a sign for v. To avoid dividing by a possibly small $\langle v, v\rangle$ when applying P, we select the sign in front of the $\|a\|e_1$ term in v as follows

$$v = \begin{cases} a + \|a\|e_1 & \text{if } a_1 \geq 0, \\ a - \|a\|e_1 & \text{if } a_1 < 0. \end{cases} \tag{11.22}$$

Example 11.3. *Let* $a = [-2, 2, 1, 4]^T$. *Then,* $\|a\| = 5$ *and*

$$v = a - \|a\|e_1 = [-7, 2, 1, 4]^T. \tag{11.23}$$
$$\tag{11.24}$$

Let us verify that $Pa = \|a\|e_1 = 5e_1$*:*

$$Pa = \begin{bmatrix} -2 \\ 2 \\ 1 \\ 4 \end{bmatrix} - 2\left(\frac{35}{70}\right)\begin{bmatrix} -7 \\ 2 \\ 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{11.25}$$

We now describe the Householder procedure, which is very similar to Gaussian elimination. Let A be an $m \times n$ matrix. We assume here $m \geq n$

and A full rank (dimension of column space equal to n). First, we transform the matrix A so that its first column a_1 becomes a multiple of e_1 by using the Householder reflection $P_1 = I - 2v_1v_1^T/\langle v_1, v_1\rangle$, where

$$v_1 = a_1 + \operatorname{sign}(a_{11})\|a_1\|e_1. \tag{11.26}$$

That is,

$$P_1A = \begin{bmatrix} * & * & \cdots & * \\ 0 & \mathrm{x} & \cdots & \mathrm{x} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \mathrm{x} & \cdots & \mathrm{x} \end{bmatrix} \tag{11.27}$$

Now, we repeat the process for the $(m-1)\times(n-1)$ block marked with x's, etc. After n steps, we obtain the $m \times n$ upper triangular matrix

$$R = \begin{bmatrix} r_{11} & \cdots & r_{12} \\ 0 & \ddots & \vdots \\ \vdots & & r_{nn} \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix}. \tag{11.28}$$

If we let $A^{(0)} = A$, we can view mathematically the j-th step of this process as

$$A^{(j)} = P_jA^{(j-1)}, \tag{11.29}$$

where P_j is the $m \times m$ orthogonal matrix

$$P_j = \begin{bmatrix} I_{j-1} & 0 \\ 0 & \tilde{P}_j \end{bmatrix}. \tag{11.30}$$

Here I_{j-1} is the identity matrix of size $(j-1)\times(j-1)$, the zeros stand for zero blocks, and $\tilde{P}_j$ is the $(m-j+1)\times(m-j+1)$ Householder matrix needed at this step. Thus,

$$P_n \cdots P_1 A = R. \tag{11.31}$$

Noting that $P_n \cdots P_1$ is orthogonal and setting $Q = P_1 \cdots P_n$ we get $A = QR$.

We discuss now implementation. In actual computations, the Householder matrices are never formed. We instead compute their effect taking into account that they are a rank-one modification to the identity matrix. For example, to evaluate Pu for $u \in \mathbb{R}^n$, we first compute

$$\beta = \frac{2}{\langle v, v\rangle} = \left(\|a\|^2 + \|a\|\ |a_1|\right)^{-1}, \tag{11.32}$$

the inner product $\langle u, v\rangle$, and set

$$Pu = u - \beta\langle u, v\rangle v. \tag{11.33}$$

Similarly, when we need to apply a Householder transformation to a matrix A we do

$$\beta = \frac{2}{\langle v, v\rangle}, \tag{11.34}$$

$$w = \beta A^T v, \tag{11.35}$$

$$PA = A - vw^T, \tag{11.36}$$

i.e. first we compute the vector w and then we modify A with the outer product $-vw^T$. Note that the latter is simply the matrix with entries $-v_i w_j$. Thus, this is much more economical than computing the full matrix product.

During the Householder QR procedure, if memory is an issue, the lower triangular part of A could be overwritten to store the vectors v_j's which define each of the employed Householder transformations. However, there is not enough space to store all the components because for v_j we need $m-j+1$ array entries and we only have $m-j$ available. One approach to overcome this is to store the diagonal elements of $A^{(j)}$ in a separate one-dimensional array to free up the needed space to store the v_j's. The Householder QR method is presented in pseudocode in Algorithm 11.2.

In applications, very often $Q^T f$ or Qf, for $f \in \mathbb{R}^n$, is needed instead of the full orthogonal matrix Q. Again, these products should be computed

Algorithm 11.2 Householder QR

```
function HV(a)                                  ▷ Computes the Householder vector v
    Compute ‖a‖ and set v ← a
    if ‖a‖ ≠ 0 then
        v[1] ← a[1] + sign(a[1])‖a‖
    end if
end function
function HPA(A, v)                              ▷ Performs PA
    β ← 2/⟨v, v⟩
    w ← βA^T v
    A ← A − wv^T
end function
function HQR(A)                                 ▷ Householder's QR factorization, m ≥ n
    for j = 1, ..., n do
        v[j : m] ← HV(A[j : m, j])
        A[j : m, j : n] ← HPA(A[j : m, j : n], v[j : m])
        r[j] ← A[j, j]              ▷ Store the diagonal to free up space for v_j
        A[j : m, j] ← v[j : m]      ▷ Store v_j in the lower triangular part of A
    end for
end function
```

exploiting the simple structure of a Householder matrix. For example to compute $Q^T f = P_n \cdots P_1 f$ we apply (11.33) repeatedly:

$$\begin{gathered} \text{for } j = 1, \ldots, n, \\ f \leftarrow P_j f. \end{gathered} \tag{11.37}$$

If needed, $Q = P_1 \cdots P_n$ can be computed similarly using repeatedly (11.34)-(11.36).

11.3 The QR Method for Eigenvalues

The most successful numerical method for the eigenvalue problem of a general square matrix A is the QR method. It is based on the QR factorization. Here Q is an orthogonal matrix and R is upper triangular.

Given an $n \times n$ matrix A, we set $A_1 = A$ and obtain its QR factorization using the Householder procedure

$$A_1 = Q_1 R_1. \tag{11.38}$$

Define $A_2 = R_1 Q_1$ so that

$$A_2 = R_1 Q_1 = Q_1^T A Q_1. \tag{11.39}$$

Now get the QR factorization of A_2, $Q_2 R_2$, and set $A_3 = R_2 Q_2$, etc.

The $k+1$-st similar matrix is generated by

$$A_{k+1} = R_k Q_k = Q_k^T A_k Q_k = (Q_1 \cdots Q_k)^T A (Q_1 \cdots Q_k). \tag{11.40}$$

It can be proved that if A is diagonalizable and with distinct eigenvalues in modulus then the sequence of matrices A_k, $k = 1, 2, \ldots$ produced by the QR method will converge to a diagonal matrix with the eigenvalues of A on the diagonal. There is no convergence proof for a general matrix A but the method is remarkably robust and fast to converge.

Example 11.4. *Consider the* 5×5 *matrix*

$$A = \begin{bmatrix} 12 & 13 & 10 & 7 & 7 \\ 13 & 18 & 9 & 8 & 15 \\ 10 & 9 & 10 & 4 & 12 \\ 7 & 8 & 4 & 4 & 6 \\ 7 & 15 & 12 & 6 & 18 \end{bmatrix} \tag{11.41}$$

the $A_{20} = R_{19}Q_{19}$ produced by the QR method gives the eigenvalues of A within 4 digits of accuracy

$$A_{20} = \begin{bmatrix} 51.7281 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 8.2771 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 4.6405 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & -2.8486 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.2028 \end{bmatrix}. \tag{11.42}$$

11.4 Reductions Prior to Applying the QR Method.

The QR method has remarkable applicability to general square matrices but it is computationally expensive in the form we presented it. Each step in the iteration requires $O(n^3)$ flops for an $n \times n$ matrix due to cost of the QR factorization. To decrease this high cost, in practice the QR method is applied to matrices that has been already suitably reduced. The idea is to reduce A to a similar matrix B, with the simplest form that can be achieved by using orthogonal transformations, i.e. $B = P^T AP$, where P is the product of orthogonal matrices. For a general matrix A, the simplest form that B can have is

$$\begin{bmatrix} b_{11} & b_{12} & \cdots & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & \cdots & b_{2n} \\ 0 & b_{32} & b_{33} & \cdots & b_{3n} \\ \vdots & \ddots & \ddots & & \vdots \\ 0 & \cdots & 0 & b_{n,n-1} & b_{nn} \end{bmatrix}, \tag{11.43}$$

This is called an upper *Hessenberg* matrix. If A is symmetric then B would be tridiagonal. This pre-processing reduction makes sense because the matrices A_k (11.40) in each iteration of the QR eigenvalue algorithm preserve the Hessenberg form and the cost of the QR factorization can be cut down to $O(n^2)$ flops for a Hessenberg matrix and to $O(n)$ flops a tridiagonal matrix.

Next, we go over the procedure to reduce a symmetric matrix to a tridiagonal one. The reduction of a general square matrix to Hessenberg form is similar.

Given a symmetric, $n \times n$ matrix A we first consider the vector $a_1 := [a_{21}, \ldots, a_{n1}]^T$ and find a Householder transformation $\tilde{P}_1$ (from $\mathbb{R}^{n-1}$ to $\mathbb{R}^{n-1}$

that renders a_1 a multiple of $e_1 \in \mathbb{R}^{n-1}$. Note that by symmetry the same transformation can be produced on the first row by $a_1^T \tilde{P}_1$. Thus, if we define

$$P_1 = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \tilde{P}_1 & \\ & & & \\ 0 & & & \end{bmatrix} \tag{11.44}$$

Then (noting that $P_1^T = P_1$),

$$P_1^T A P_1 = \begin{bmatrix} * & * & 0 & \cdots & 0 \\ * & \text{x} & & \cdots & \text{x} \\ 0 & \text{x} & & \cdots & \text{x} \\ \vdots & & & & \\ 0 & \text{x} & & \cdots & \text{x} \end{bmatrix}. \tag{11.45}$$

The same procedure can now be applied to the $(n-1) \times (n-1)$ sub-matrix marked with x's, etc. We can summarize the reduction as follows. Setting $A_1 = A$ we obtain

$$A_{k+1} = P_k^T A_k P_k, \quad k = 1, \ldots, n-2. \tag{11.46}$$

Here, P_k is the orthogonal matrix formed with the Householder transformation $\tilde{P}_k$ of the k-th step:

$$P_k = \begin{bmatrix} I_k & 0 \\ 0 & \tilde{P}_k \end{bmatrix}, \tag{11.47}$$

where I_k is the $k \times k$ identity matrix and the zeros represent zero blocks of the corresponding size. After $n-2$ steps the resulting matrix A_{n-1} is tridiagonal.

Since

$$A_{k+1} = P_k^T A_k P_k = P_k^T P_{k-1}^T A_{k-1} P_{k-1} P_k \tag{11.48}$$

it follows that

$$A_{n-1} = P_{n-2}^T \cdots P_1^T A P_1 \cdots P_{n-2}. \tag{11.49}$$

Moreover, symmetry implies that $P_{n-2}^T \cdots P_1^T = (P_1 \cdots P_{n-2})^T$. Thus, defining $P = P_1 \cdots P_{n-2}$ we obtain

$$A_{n-1} = P^T A P. \tag{11.50}$$

To summarize, given a symmetric matrix A, there is an orthogonal and symmetric matrix P, constructed with Householder transformations, such that $P^T AP$ is tridiagonal.

For symmetric tridiagonal matrices there are also specialized algorithms to find a particular eigenvalue or set of eigenvalues, to any desired accuracy, located in a given interval by using bisection.

Householder reflections are also used in the effective Golub-Reinsch method to compute the SVD of an $m \times n$ matrix ($m \geq n$). Rather than applying directly the QR algorithm to $A^T A$, which could result in a loss of accuracy, this method uses Householder transformations to reduce A to bidiagonal form. Then, the SVD of this bidiagonal matrix is obtained using other orthogonal transformations related to the QR method with a shift.

11.5 Bibliographic Notes

Section 11.1 . The presentation of this section is a simplified version of Section 6.6.3 in [SB02], where additionally the issue of ill-conditioning is examined. The use of $u^T Au$ to approximate the eigenvalue follows that in [GVL13] [7.3.1]. A thorough discussion of the convergence of the power method and its variants and of the effects of round-off errors are extensively discussed in the treatise by Wilkinson [Wil65].

Section 11.2 . Alston Householder proposed his approach to obtain the QR factorization of a matrix in 1958 [Hou58]. The reason for employing orthogonal transformations in the reduction of A to upper triangular form is that the condition number of orthogonal transformations is 1. Prior to Householder's work, the method of W. Givens, based on orthogonal transformations constructed with planar rotations, was a popular approach to do this reduction [Hou58]. The method of Givens is covered in most numerical linear algebra texts, see for example [GVL13, Cia89].

Section 11.3 . The QR algorithm is due to J.G.F. Francis [Fra61, Fra62] and V.N. Kublanovskaya [Kub62]. According to Wilkinson [Wil65][p.569], the work of Francis' dates from 1959 but was published only until 1961 and Kublanovskaya discovered the algorithm independently. A proof for convergence in the special case when A is invertible and with all eigenvalues distinct in modulus is given in [Cia89].

Section 11.4 . This section is modeled after Section 6.2 in [Cia89]. Implementation details for both tridiagonal reduction in the symmetric case and Hessenberg form in the non-symmetric case can be found in [GVL13]. The Golub-Reinsch [GR71] method to compute the SVD is discussed in [SB02].

Chapter 12

Non-Linear Equations

In this chapter we consider the problem of finding zeros of a continuous function f, i.e. solving $f(x) = 0$ or a system of nonlinear equations:

$$\begin{aligned} f_1(x_1, x_2, \cdots, x_m) &= 0, \\ f_2(x_1, x_2, \cdots, x_m) &= 0, \\ &\vdots \\ f_m(x_1, x_2, \cdots, x_m) &= 0. \end{aligned} \tag{12.1}$$

We will write this general system as

$$f(x) = 0, \tag{12.2}$$

where $f : U \subseteq \mathbb{R}^m \to \mathbb{R}^m$. Unless otherwise noted the function f is assumed to be smooth in its domain U. More precisely, we present some basic numerical methods for approximating solutions of $f(x) = 0$ in the scalar case $m = 1$ and discuss briefly the case of a system of nonlinear equations ($m > 1$) in the last section.

The numerical approximation of solutions to nonlinear equations is important in many applications and is also needed as part of some numerical methods for the solution of nonlinear differential equations and nonlinear optimization problems.

12.1 Bisection

We are going to start with a very simple but robust method that relies only on the continuity of f and the existence of a zero.

Suppose we are interested in solving a nonlinear equation in one unknown

$$f(x) = 0, \tag{12.3}$$

where f is a continuous function on an interval $[a, b]$ and has at least one zero there. Suppose also that f has values of different sign at the end points of the interval, i.e.

$$f(a)f(b) < 0. \tag{12.4}$$

By the intermediate value theorem, f has at least one zero x^* in (a, b). To locate it, we bisect $[a, b]$ to obtain the two subintervals $[a, c]$ and $[c, b]$ with $c = \frac{1}{2}(a+b)$. If $f(c) = 0$, we are done. Otherwise, we select the subinterval on which f changes sign ($[a, c]$ if $f(a)f(c) < 0$, else $[c, b]$) and repeat the process until we bracket a zero within a desired accuracy. The resulting algorithm is called *bisection* and is listed in pseudocode in Algorithm 12.1.

Algorithm 12.1 The Bisection Method

Given f, a and b ($a < b$), TOL, and N_{max}, set $k = 1$ and do:
while $(b - a) > TOL$ and $k \le N_{max}$ **do**
$\quad c = (a + b)/2$
$\quad$ **if** $f(c) == 0$ **then**
$\quad\quad x^* = c$ $\qquad\qquad$ ▷ This is the solution
$\quad\quad$ stop
$\quad$ **end if**
$\quad$ **if** sign$(f(c))$ == sign$(f(a))$ **then**
$\quad\quad a \leftarrow c$
$\quad$ **else**
$\quad\quad b \leftarrow c$
$\quad$ **end if**
$\quad k \leftarrow k + 1$
end while
$x^* \leftarrow (a + b)/2$

Example 12.1. *Let $f(x) = e^{-x} - 2x$. Note that $f(0)f(1) < 0$ and hence his continuous function has a zero x^* in $[0, 1]$. In fact it is the only zero f has as $f'(x) < 0$ for all x. If we apply the bisection algorithm to find x^* we get* *following sequence of subintervals*

$$
\begin{aligned}
[a_1, b_1] &= [0, 1] \\
[a_2, b_2] &= [0, 0.5] \\
[a_3, b_3] &= [0.25, 0.5] \\
[a_4, b_4] &= [0.25, 0.375] \\
[a_5, b_5] &= [0.3125, 0.375] \\
[a_6, b_6] &= [0.34375, 0.375] \\
[a_7, b_7] &= [0.34375, 0.359375] \\
[a_8, b_8] &= [0.3515625, 0.359375] \\
&\vdots
\end{aligned}
$$

Thus, it follows that within two digits of accuracy $x^* \approx 0.35$.

12.1.1 Convergence of the Bisection Method

Starting with $a_1 = a$ and $b_1 = b$, the bisection method generates a sequence of midpoints

$$c_n = \frac{a_n + b_n}{2}, \quad n = 1, 2, \ldots \tag{12.5}$$

where each a_n and b_n, are the endpoints of the subinterval in which f changes sign at each bisection step. Since

$$b_n - a_n = \frac{b - a}{2^{n-1}}, \quad n = 1, 2, \ldots \tag{12.6}$$

and c_n is the midpoint of the interval then

$$|c_n - x^*| \leq \frac{1}{2}(b_n - a_n) = \frac{b - a}{2^n} \tag{12.7}$$

and consequently $c_n \to x^*$, as $n \to \infty$.

12.2 Rate of Convergence

We define now in precise terms the rate of convergence of a sequence of approximations to a value x^*.

Definition 12.1. *Suppose a sequence* $\{x_n\}_{n=1}^{\infty}$ *converges to* x^* *as* $n \to \infty$. *We say that* $x_n \to x^*$ *of order* p $(p \geq 1)$ *if there is a positive integer* N *and a constant* C *such that*

$$|x_{n+1} - x^*| \leq C\,|x_n - x^*|^p\,, \quad \text{for all } n \geq N. \tag{12.8}$$

or equivalently

$$\lim_{n\to\infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|^p} = C. \tag{12.9}$$

For $p = 1$ *we require* $C < 1$ *and we say that the sequence converges linearly to* x^*.

Example 12.2. *The sequence generated by the bisection method converges linearly to* x^* *because*

$$\frac{|c_{n+1} - x^*|}{|c_n - x^*|} \leq \frac{\frac{b-a}{2^{n+1}}}{\frac{b-a}{2^n}} = \frac{1}{2}. \tag{12.10}$$

Let us examine the significance of the rate of convergence. Consider first, $p = 1$, linear convergence. Suppose

$$|x_{n+1} - x^*| \approx C|x_n - x^*|, \quad n \geq N. \tag{12.11}$$

Then

$$\begin{aligned}
&|x_{N+1} - x^*| \approx C|x_N - x^*|,\\
&|x_{N+2} - x^*| \approx C|x_{N+1} - x^*| \approx C(C|x_N - x^*|) = C^2|x_N - x^*|.
\end{aligned}$$

Continuing this way we get

$$|x_{N+k} - x^*| \approx C^k|x_N - x^*|, \quad k = 0, 1, \ldots \tag{12.12}$$

and this is the reason for the requirement $C < 1$ when $p = 1$. If the error at the N-th step, $|x_N - x^*|$, is small enough it will be further reduced by a factor of C^k after k more steps. Setting $C^k = 10^{-d_k}$, this reduction corresponds to approximately

$$d_k = \left(\log_{10} \frac{1}{C}\right) k \tag{12.13}$$

digits.

Let us now do a similar analysis for $p = 2$, quadratic convergence. We have

$$\begin{aligned}
|x_{N+1} - x^*| &\approx C|x_N - x^*|^2, \\
|x_{N+2} - x^*| &\approx C|x_{N+1} - x^*|^2 \approx C(C|x_N - x^*|^2)^2 = C^3|x_N - x^*|^4, \\
|x_{N+3} - x^*| &\approx C|x_{N+2} - x^*|^2 \approx C(C^3|x_N - x^*|^4)^2 = C^7|x_N - x^*|^8.
\end{aligned}$$

It is easy to prove by induction that

$$|x_{N+k} - x^*| \approx C^{2^k-1}|x_N - x^*|^{2^k}, \quad k = 0, 1, \ldots \tag{12.14}$$

To see how many digits of accuracy we gain in k steps beginning from x_N, we write $C^{2^k-1}|x_N - x^*|^{2^k} = 10^{-d_k}|x_N - x^*|$, and solving for d_k we get

$$d_k = \left(\log_{10}\frac{1}{C} + \log_{10}\frac{1}{|x_N - x^*|}\right)(2^k - 1). \tag{12.15}$$

It is not difficult to prove that for the general $p > 1$ and as $k \to \infty$ we get $d_k = \alpha_p p^k$, where $\alpha_p = \frac{1}{p-1}\log_{10}\frac{1}{C} + \log_{10}\frac{1}{|x_N - x^*|}$.

12.3 Interpolation-Based Methods

Assuming again that f is a continuous function in $[a, b]$ and $f(a)f(b) < 0$ we can proceed as in the bisection method but instead of using the midpoint $c = \frac{1}{2}(a + b)$ to subdivide $[a, b]$ we could use the zero of the interpolating polynomial of $(a, f(a))$ and $(b, f(b))$. This is called the *method of false position*. Unfortunately, this method only converges linearly and under stronger assumptions than the bisection method.

An alternative, interpolation approach to find approximations to a solution of $f(x) = 0$ is to proceed as follows: Given $m + 1$ approximations $x_0, \ldots, x_m$ of a zero x^* of f, construct the interpolating polynomial of f, p_m, at those points, and set the root of p_m closest to x_m as the new approximation of x^*. In practice, only $m = 1, 2$ are used. The method for $m = 1$ is called the *secant method* and we will look at it in some detail later. The method for $m = 2$ is called *Muller's Method*.

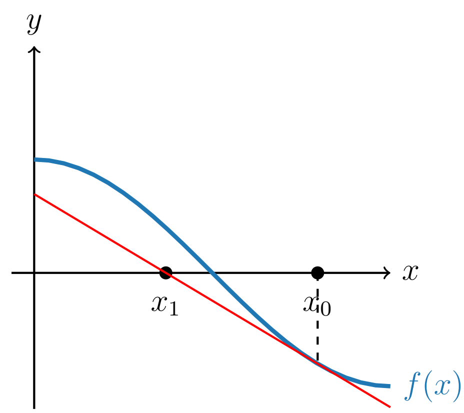

Figure 12.1: Geometric illustration of Newton's method. Given an approximation x_0 of a zero of f, x_1 is the zero of the tangent line (in red) of f at x_0.

12.4 Newton's Method

If the function f is at least $C^2[a, b]$, and we have already a good approximation x_0 of a zero x^* of f, then the tangent line of f at x_0, $y = f(x_0) + f'(x_0)(x - x_0)$ provides a good approximation to f in a small neighborhood of x_0, i.e.

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0). \tag{12.16}$$

We can define the next approximation as the zero of this tangent line, i.e.

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}. \tag{12.17}$$

Figure 12.1 illustrates the geometric meaning of x_1. Next, we consider the tangent line of f at x_1, set x_1 as its zero, etc. At the k-th step of this iterative process we get the new approximation x_{k+1} according to:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, \ldots \tag{12.18}$$

This iteration is called *Newton's method* or Newton-Raphson's method. There are some conditions for this method to converge. But when it does, it converges at least quadratically [$p = 2$ in (12.8)]. Indeed, a Taylor expansion of f around x_k gives

$$f(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2} f''(\xi_k(x))(x - x_k)^2, \tag{12.19}$$

where $\xi_k(x)$ is a point between x and x_k. Evaluating at $x = x*$ and using that $f(x^*) = 0$ we get

$$0 = f(x_k) + f'(x_k)(x^* - x_k) + \frac{1}{2} f''(\xi_k(x^*))(x^* - x_k)^2, \tag{12.20}$$

which, assuming $f'(x_k) \neq 0$, we can recast as

$$\begin{aligned} x^* &= x_k - \frac{f(x_k)}{f'(x_k)} - \frac{1}{2} \frac{f''(\xi_k^*)}{f'(x_k)}(x^* - x_k)^2 \\ &= x_{k+1} - \frac{1}{2} \frac{f''(\xi_k^*)}{f'(x_k)}(x^* - x_k)^2, \end{aligned} \tag{12.21}$$

where $\xi_k^* = \xi_k(x^*)$. Thus,

$$|x_{k+1} - x^*| = \frac{1}{2} \frac{|f''(\xi^*)|}{|f'(x_k)|} |x_k - x^*|^2. \tag{12.22}$$

Therefore, if the sequence $\{x_k\}_{k=0}^{\infty}$ generated by Newton's method converges then it does so at least quadratically.

Theorem 12.1. *Let x^* be a simple zero of f (i.e. $f(x^*) = 0$ and $f'(x^*) \neq 0$) and suppose $f \in C^2$ on an interval containing x^*. Then, there is a neighborhood of x^* such that Newton's method converges to x^* for any initial guess in that neighborhood.*

Proof. For $\epsilon > 0$ consider the neighborhood I_ϵ of x^* consisting of all the points x such that $|x - x^*| \leq \epsilon$. We can choose ϵ small enough so that f is C^2 in I_ϵ and $f'(x) \neq 0$ for all $x \in I_\epsilon$, since f' is continuous and $f'(x^*) \neq 0$. Now consider the quantity

$$M(\epsilon) = \frac{1}{2} \frac{\max_{x \in I_\epsilon} |f''(x)|}{\min_{x \in I_\epsilon} |f'(x)|}. \tag{12.23}$$

We can select ϵ sufficiently small so that in addition to satisfying the above conditions we have $\epsilon M(\epsilon) < 1$. This is possible because

$$\lim_{\epsilon \to 0} M(\epsilon) = \frac{1}{2} \frac{|f''(x^*)|}{|f'(x^*)|} < +\infty. \tag{12.24}$$

The condition $\epsilon M(\epsilon) < 1$ allows us to guarantee that x^* is the only zero of f in I_ϵ, as we show now. A Taylor expansion of f around x^* gives

$$\begin{aligned} f(x) &= f(x^*) + f'(x^*)(x - x^*) + \frac{1}{2}f''(\xi)(x - x^*)^3 \\ &= f'(x^*)(x - x^*)\left(1 + \frac{1}{2}(x - x^*)\frac{f''(\xi)}{f'(x^*)}\right), \end{aligned} \tag{12.25}$$

for some ξ between x and x^*. Since for all $x \in I_\epsilon$

$$\frac{1}{2}\left|(x - x^*)\frac{f''(\xi)}{f'(x^*)}\right| = \frac{1}{2}|x - x^*|\frac{|f''(\xi)|}{|f'(x^*)|} \leq \epsilon M(\epsilon) < 1 \tag{12.26}$$

then $f(x) \neq 0$ for all $x \in I_\epsilon$ unless $x = x^*$.

We will now show that Newton's iteration is well defined starting from any initial guess $x_0 \in I_\epsilon$. We prove this by induction. From (12.22) with $k = 0$ it follow that $x_1 \in I_\epsilon$ as

$$|x_1 - x^*| = |x_0 - x|^2 \left|\frac{\frac{1}{2}f''(\xi_0)}{f'(x_0)}\right| \leq \epsilon^2 M(\epsilon) \leq \epsilon. \tag{12.27}$$

Now assume that $x_k \in I_\epsilon$. Then, again from (12.22)

$$|x_{k+1} - x^*| = |x_k - x|^2 \left|\frac{\frac{1}{2}f''(\xi_k)}{f'(x_k)}\right| \leq \epsilon^2 M(\epsilon) < \epsilon \tag{12.28}$$

so $x_{k+1} \in I_\epsilon$.

Finally,

$$\begin{aligned} |x_{k+1} - x^*| &\leq |x_k - x^*|^2 M(\epsilon) = |x_k - x^*|\epsilon M(\epsilon) \\ &\leq |x_{k-1} - x^*|(\epsilon M(\epsilon))^2 \\ &\vdots \\ &\leq |x_0 - x^*|(\epsilon M(\epsilon))^{k+1} \end{aligned}$$

and since $\epsilon M(\epsilon) < 1$ it follows that $x_k \to x^*$ as $k \to \infty$. □

This theorem provides sufficient conditions to guarantee convergence *locally*, i.e. provided the initial guess is in a sufficiently small neighborhood of x^*. However, for some functions Newton's method might converge *globally*, i.e. for any initial guess. But in general it is a good practice to initialize Newton's method with a good initial guess, typically obtained with another method, like for example bisection.

Example 12.3. *Let us consider again the equation* $e^{-x} - 2x = 0$. *We know from Example 12.1 that* $x^* \approx 0.35156$. *If we perform Newton's method with* $x_0 = 0.3$ *we get* x^* *within machine precision in just three iterations. With quadratic convergence, the number of digits of accuracy approximately doubles with each iteration.*

$$
\begin{aligned}
x_0 &= 0.3 \\
x_1 &= 0.3513781686137115, \\
x_2 &= 0.3517336948002153, \\
x_3 &= 0.3517337112491958, \\
x_4 &= 0.3517337112491958.
\end{aligned}
$$

Because of the convexity of f, *Newton's method will converge for any initial guess. This is an example of* global *convergence.*

Example 12.4. *Consider* $f(x) = x^3 - 5x$. *Clearly, one of the roots is* $x^* = 0$. *However, if we apply Newton's method with* $x_0 = 1$ *we get*

$$
x_1 = 1 - \frac{1-5}{3(1)^2 - 5} = -1, \tag{12.29}
$$

$$
x_2 = -1 - \frac{-1+5}{3(-1)^2 - 5} = 1, \tag{12.30}
$$

we get stuck oscillating between $+1$ *and* -1 *and the method fails to converge. This is behavior is called a cycle. On the other hand, if we take* $x_0 = 0.5$, *we obtain a fast, quadratic convergence to* $x^* = 0$:

$$
\begin{aligned}
x_1 &= -0.058823529411764\,72, \\
x_2 &= 8.158603247124252 \times 10^{-5}, \\
x_3 &= -2.1722380838529176 \times 10^{-13}, \\
x_4 &= 0.0.
\end{aligned}
$$

This illustrates a case of local *convergence.*

12.5 The Secant Method

Sometimes it could be computationally expensive or not possible to evaluate exactly the derivative of f. This is in fact often the case for large systems of nonlinear equations. The following method, known as the secant method, approximates the derivative by a backward finite difference, i.e. by the first divided difference of f with respect to x_{k-1} and x_k

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} = f[x_{k-1}, x_k]. \tag{12.31}$$

This approximation corresponds to replacing the tangent line by the secant in Newton's method, that is

$$x_{k+1} = x_k - \frac{f(x_k)}{f[x_{k-1}, x_k]}, \quad k = 1, 2, \ldots \tag{12.32}$$

Note that we need to start the secant iteration (12.32) with two different approximations x_0 and x_1.

Since $f(x^*) = 0$

$$\begin{aligned}
x_{k+1} - x^* &= x_k - x^* - \frac{f(x_k) - f(x^*)}{f[x_{k-1}, x_k]} \\
&= (x_k - x^*)\left(1 - \frac{\frac{f(x_k)-f(x^*)}{x_k - x^*}}{f[x_{k-1}, x_k]}\right) \\
&= (x_k - x^*)\left(1 - \frac{f[x^*, x_k]}{f[x_{k-1}, x_k]}\right) \\
&= (x_k - x^*)\left(\frac{f[x_{k-1}, x_k] - f[x^*, x_k]}{f[x_{k-1}, x_k]}\right) \\
&= (x_k - x^*)(x_{k-1} - x^*)\left(\frac{\frac{f[x_k, x_{k-1}] - f[x^*, x_k]}{x_{k-1} - x^*}}{f[x_{k-1}, x_k]}\right) \\
&= (x_k - x^*)(x_{k-1} - x^*)\frac{f[x_{k-1}, x_k, x^*]}{f[x_{k-1}, x_k]}.
\end{aligned} \tag{12.33}$$

If $x_k \to x^*$ as $k \to \infty$ then

$$\lim_{k\to\infty} \frac{f[x_{k-1}, x_k, x^*]}{f[x_k, x_{k-1}]} = \frac{\frac{1}{2}f''(x^*)}{f'(x^*)} \tag{12.34}$$

and

$$\lim_{k\to\infty} \frac{x_{k+1} - x^*}{x_k - x^*} = 0, \tag{12.35}$$

i.e. the sequence generated by the secant method would converge faster than linear.

Defining $e_k = |x_k - x^*|$, the calculation above suggests

$$e_{k+1} \approx C e_k e_{k-1}, \tag{12.36}$$

where C is a positive constant. Let us try to determine the rate of convergence of the secant method. Starting with the ansatz $e_k \approx A e_{k-1}^p$ or equivalently $e_{k-1} = \left(\frac{1}{A} e_k\right)^{1/p}$ we have

$$e_{k+1} \approx C e_k e_{k-1} \approx C e_k \left(\frac{1}{A} e_k\right)^{\frac{1}{p}}.$$

On the other hand $e_{k+1} = A e_k^p$, therefore

$$A e_k^p \approx C e_k \left(\frac{1}{A} e_k\right)^{\frac{1}{p}}, \tag{12.37}$$

which implies

$$\frac{A^{1+\frac{1}{p}}}{C} \approx e_k^{1-p+\frac{1}{p}}. \tag{12.38}$$

Since the left hand side is a constant we must have $1 - p + \frac{1}{p} = 0$, or equivalently $p^2 - p - 1 = 0$, whose solutions are $p = \frac{1 \pm \sqrt{5}}{2}$. Consequently,

$$p = \frac{1 + \sqrt{5}}{2} = \approx 1.61803 \tag{12.39}$$

gives the rate of convergence of the secant method. It is better than linear, but worse than quadratic. Sufficient conditions for local convergence are as those in Newton's method.

Example 12.5. *Consider again the equation $e^{-x} - 2x$. Starting with $x_0 = 0.3$ and $x_1 = 0.2$, the secant method approximates the solution to machine* *precision with x_6:*

$$
\begin{aligned}
x_0 &= 0.3\\
x_1 &= 0.2\\
x_2 &= 0.3506699785963344,\\
x_3 &= 0.35171205360889224,\\
x_4 &= 0.3517337082511913,\\
x_5 &= 0.3517337112491874,\\
x_6 &= 0.35173371124919584.
\end{aligned}
$$

The number of digits of accuracy almost doubles per iteration. Convergence is slightly slower than in Newton's method as expected from (12.39) *but still fast.*

12.6 Fixed Point Iteration

Newton's method is a particular example of a functional iteration of the form

$$x_{k+1} = g(x_k), \quad k = 0, 1, \ldots \tag{12.40}$$

with

$$g(x) = x - \frac{f(x)}{f'(x)}. \tag{12.41}$$

Clearly, if x^* is a zero of f then x^* is a fixed point of g, i.e.

$$g(x^*) = x^* \tag{12.42}$$

For a given function f, if $g(x) = x - f(x)$ or $g(x) = x + f(x)$, then a solution x^* of $f(x) = 0$ is a fixed point of g and vice-versa. More generally, the same is true if $g(x) = x \pm \phi(x)f(x)$, where ϕ is any function defined a neighborhood of x^*. For example, in Newton's method $\phi = 1/f'$.

We look next at fixed point iterations as a tool for solving $f(x) = 0$. The central idea is to select g such that it shrinks distances between two points and thus, starting with an initial approximation, the error is decreased at every iteration. This desired property of g is made precise in the following definition.

Definition 12.2. *Let g is defined in an interval $[a, b]$. We say that g is a contraction or a contractive map if there is a constant L with $0 \leq L < 1$ such that*

$$|g(x) - g(y)| \leq L|x - y|, \quad \text{for all } x, y \in [a, b]. \tag{12.43}$$

Example 12.6. *The function $g(x) = \frac{1}{4}x^2$ in $[0, 1]$ is a contraction because*

$$\left|\frac{1}{4}x^2 - \frac{1}{4}y^2\right| = \frac{1}{4}\left|(x + y)(x - y)\right| = \frac{1}{4}|x + y|\,|x - y| \leq \frac{1}{2}|x - y| \tag{12.44}$$

for all $x, y \in [0, 1]$.

If x^* is a fixed point of g in $[a, b]$ and g is a contraction, then

$$\begin{aligned} |x_k - x^*| &= |g(x_{k-1}) - g(x^*)| \\ &\leq L|x_{k-1} - x^*| \\ &\leq L^2|x_{k-2} - x^*| \\ &\;\vdots \\ &\leq L^k|x_0 - x^*| \to 0, \text{ as } k \to \infty. \end{aligned} \tag{12.45}$$

Theorem 12.2 (Contraction Mapping Theorem)**.** *If g is a contraction on $[a, b]$ and maps $[a, b]$ into $[a, b]$, then g has a unique fixed point x^* in $[a, b]$ and the fixed point iteration* (12.40) *converges to it for any $x_0 \in [a, b]$. Moreover,*

(a) $|x_k - x^*| \leq L^k|x_0 - x^*|$,

(b) $|x_k - x^*| \leq \dfrac{L^k}{1 - L}|x_1 - x_0|$.

Proof. Since $g : [a, b] \to [a, b]$, the fixed point iteration $x_{k+1} = g(x_k), k = 0, 1, ...$ is well-defined. Proceeding as in (12.45) we have

$$\begin{aligned} |x_{k+1} - x_k| &= |g(x_k) - g(x_{k-1})| \\ &\leq L|x_k - x_{k-1}| \leq \cdots \leq L^k|x_1 - x_0|. \end{aligned} \tag{12.46}$$

Now, for $n \geq m$

$$x_n - x_m = x_n - x_{n-1} + x_{n-1} - x_{n-2} + \ldots + x_{m+1} - x_m \tag{12.47}$$

and so

$$\begin{aligned}
|x_n - x_m| &\leq |x_n - x_{n-1}| + |x_{n-1} - x_{n-2}| + \ldots + |x_{m+1} - x_m| \\
&\leq L^{n-1}|x_1 - x_0| + L^{n-2}|x_1 - x_0| + \ldots + L^m|x_1 - x_0| \\
&\leq L^m|x_1 - x_0| \left(1 + L + L^2 + \ldots L^{n-1-m}\right) \\
&\leq L^m|x_1 - x_0| \sum_{j=0}^{\infty} L^j = \frac{L^m}{1-L}|x_1 - x_0|.
\end{aligned} \tag{12.48}$$

Thus, given $\epsilon > 0$, there is N such that

$$\frac{L^N}{1-L}|x_1 - x_0| \leq \epsilon. \tag{12.49}$$

Therefore, for $n \geq m \geq N$, $|x_n - x_m| \leq \epsilon$, that is $\{x_n\}_{n=0}^{\infty}$ is a Cauchy sequence in $[a, b]$ and so it converges to a point $x^* \in [a.b]$. But

$$|x_k - g(x^*)| = |g(x_{k-1}) - g(x^*)| \leq L|x_{k-1} - x^*|, \tag{12.50}$$

thus $x_k \to g(x^*)$ as $k \to \infty$ i.e. x^* is a fixed point of g. We already proved part (a) in (12.45) and part (b) follows by taking the limit as $n \to \infty$ in (12.48).

Finally, we prove that the fixed point is unique. Suppose that there are two fixed points, $x_1, x_2 \in [a, b]$. Then,

$$|x_1 - x_2| = |g(x_1) - g(x_2)| \leq L|x_1 - x_2|, \tag{12.51}$$

which implies

$$(1 - L)|x_1 - x_2| \leq 0 \tag{12.52}$$

but $0 \leq L < 1$, therefore $|x_1 - x_2| = 0$ and thus $x_1 = x_2$. □

If g is differentiable in (a, b), then by the mean value theorem

$$g(x) - g(y) = g'(\xi)(x - y) \tag{12.53}$$

for some ξ between x and y. If the derivative is bounded, i.e.

$$|g'(x)| \leq L \quad \text{for all } x \in (a, b) \tag{12.54}$$

and $0 \leq L < 1$, then $|g(x) - g(y)| \leq L|x - y|$, i.e. g is contractive in $[a, b]$.

Example 12.7. *Let $g(x) = \frac{1}{6}(x^3 + 3)$ for $x \in [0,1]$. Then, $0 \le g(x) \le 1$ and $|g'(x)| \le \frac{1}{2}$ for all $x \in [0,1]$. Thus, g is contractive in $[0,1]$ and the fixed point iteration will converge to the unique fixed point of g in $[0,1]$.*

Note that if g is differentiable

$$x_{k+1} - x^* = g(x_k) - g(x^*) = g'(\xi_k)(x_k - x^*), \tag{12.55}$$

for some ξ_k between x_k and x^*. Thus, if the fixed point iteration converges to x^*

$$\lim_{k\to\infty} \frac{x_{k+1} - x^*}{x_k - x^*} = g'(x^*) \tag{12.56}$$

and unless $g'(x^*) = 0$, the fixed point iteration would converges only linearly. In Newton's method $g(x) = x - f(x)/f'(x)$, so

$$g'(x) = 1 - \frac{(f'(x))^2 - f''(x)f(x)}{(f'(x))^2} \tag{12.57}$$

and consequently $g(x^*) = 0$, which explains the quadratic convergence of Newton's method.

12.7 Systems of Nonlinear Equations

We now look at the problem of finding numerical approximation to the solution(s) of a nonlinear system of equations $f(x) = 0$, where $f : U \subseteq \mathbb{R}^m \to \mathbb{R}^m$.

The main approach to solve a nonlinear system is fixed point iteration $x_{k+1} = g(x_k)$, $k = 0, 1, \ldots$, where we assume that g is defined on a closed set $B \subseteq \mathbb{R}^m$ and $g : B \to B$.

The map g is a contraction (with respect to some norm,$\|\cdot\|$) if there is a constant L with $0 \le L < 1$ and

$$\|g(x) - g(y)\| \le L\|x - y\|, \quad \text{for all } x, y \in B. \tag{12.58}$$

Then, as we know, by the contraction map principle, g has a unique fixed point and the sequence generated by the fixed point iteration (12.40) converges to it.

If g is C^1 on some convex set $B \subseteq \mathbb{R}^m$, for example a ball, then there is a mean value theorem that stems from the one-variable case as follows.

Consider the line segment $x + t(y - x)$ for $t \in [0, 1]$ with x, y fixed in B and define the one-variable function

$$h(t) = g(x + t(y - x)). \tag{12.59}$$

Then, by the chain rule, $h'(t) = Dg(x+t(y-x))(y-x)$, where Dg stands for the derivative matrix (the Jacobian matrix) of g. Hence, using the definition of h and the fundamental theorem of Calculus we have

$$\begin{aligned} g(y) - g(x) = h(1) - h(0) &= \int_0^1 h'(t)dt \\ &= \int_0^1 Dg(x + t(y - x))(y - x)dt. \end{aligned} \tag{12.60}$$

We can now use this mean value result. Suppose there is $0 \leq L < 1$ such that

$$\|Dg(x)\| \leq L, \quad \text{for all } x \in B, \tag{12.61}$$

for some subordinate norm $\| \cdot \|$. Then,

$$\|g(y) - g(x)\| \leq L\|y - x\| \tag{12.62}$$

and g is a contraction (in that norm).

12.7.1 Newton's Method for Systems

By Taylor's theorem

$$f(x) \approx f(x_0) + Df(x_0)(x - x_0) \tag{12.63}$$

so if we take x_1 as the zero of the right hand side of (12.63) we get

$$x_1 = x_0 - [Df(x_0)]^{-1} f(x_0). \tag{12.64}$$

Continuing this way, Newton's method for the system of equations $f(x) = 0$ can be written as

$$x_{k+1} = x_k - [Df(x_k)]^{-1} f(x_k). \tag{12.65}$$

In the implementation of Newton's method for a large system of equations, we do not compute the inverse matrix. Instead, we solve the linear system $Df(x_k)\Delta x_k = -f(x_k)$ at each iteration and do the update $x_{k+1} = x_k + \Delta x_k$.

To illustrate Newton's method for a system we consider the simplest ($m = 2$) case:

$$\begin{aligned} f_1(x, y) &= 0, \\ f_2(x, y) &= 0. \end{aligned} \tag{12.66}$$

We are labeling the independent variables x and y instead of x_1 and x_2 to avoid using double indices in the iteration. Then,

$$\begin{aligned} x_{k+1} &= x_k + \Delta x_k, \\ y_{k+1} &= y_k + \Delta y_k, \end{aligned} \tag{12.67}$$

where $[\Delta x_k, \Delta y_k]^T$ is the solution of the linear system

$$\begin{bmatrix} \dfrac{\partial f_1}{\partial x}(x_k, y_k) & \dfrac{\partial f_1}{\partial y}(x_k, y_k) \\ \dfrac{\partial f_2}{\partial x}(x_k, y_k) & \dfrac{\partial f_2}{\partial y}(x_k, y_k) \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \\ \Delta y_k \end{bmatrix} = - \begin{bmatrix} f_1(x_k, y_k) \\ \\ f_2(x_k, y_k) \end{bmatrix}. \tag{12.68}$$

Using Kramer's rule we find

$$\begin{aligned} \Delta x_k &= \frac{1}{J_k} \left(f_2(x_k, y_k) \frac{\partial f_1}{\partial y}(x_k, y_k) - f_1(x_k, y_k) \frac{\partial f_2}{\partial y}(x_k, y_k) \right), \\ \Delta y_k &= \frac{1}{J_k} \left(f_1(x_k, y_k) \frac{\partial f_2}{\partial x}(x_k, y_k) - f_2(x_k, y_k) \frac{\partial f_1}{\partial x}(x_k, y_k) \right), \end{aligned} \tag{12.69}$$

where

$$J_k = \frac{\partial f_1}{\partial x}(x_k, y_k) \frac{\partial f_2}{\partial y}(x_k, y_k) - \frac{\partial f_1}{\partial y}(x_k, y_k) \frac{\partial f_2}{\partial x}(x_k, y_k). \tag{12.70}$$

Example 12.8. *Consider the nonlinear system*

$$\begin{aligned} x^2 + y^2 &= 1, \\ xy &= 0. \end{aligned} \tag{12.71}$$

It has a solutions $(1, 0)$ *and* $(0, 1)$*. Letting* $f_1(x, y) = x^2 + y^2 - 1$ *and* $f_2(x, y) = xy$*,* (12.67)*,* (12.69)*, and* (12.70) *give us for* $k = 0, 1, \ldots$

$$\begin{aligned} x_{k+1} &= x_k - \frac{x_k(x_k^2 - y_k^2 - 1)}{2(x_k^2 - y_k^2)}, \\ y_{k+1} &= y_k - \frac{y_k(x_k^2 - y_k^2 + 1)}{2(x_k^2 - y_k^2)}. \end{aligned} \tag{12.72}$$

Starting with $(x_0, y_0) = (0.6, 0.3)$ *the solution* $(1, 0)$ *is computed within machine precision in just 6 iterations:*

$$\begin{aligned}
(x_1, y_1) &= (1.411111111111111, -0.4055555555555555),\\
(x_2, y_2) &= (1.091789018985025, -0.09177367214647353),\\
(x_3, y_3) &= (1.0071173377858802, -0.007117337668119214),\\
(x_4, y_4) &= (1.0000499455377991, -4.9945537799225316 \times 10^{-5}),\\
(x_5, y_5) &= (1.0000000024943076, -2.494307586986392 \times 10^{-9}),\\
(x_6, y_6) &= (1.0, -6.221570734917062 \times 10^{-18}).
\end{aligned}$$

As observed in the other examples of quadratic convergence, the number of digits of accuracy approximately double with each iteration.

12.8 Bibliographic Notes

The main references for this chapter were Chapter 5 of Schwarz's book, Chapter 4 of Gautschi's book [Gau11], and the classical book by Ortega and Rheinboldt [OR70] (for systems of nonlinear equations).

Section 12.1 . There are problems in which one is interested in finding selectively some or all the roots of a polynomial. The bisection method can be combined with a theorem of Sturm to achieve this. This method is described in [Gau11](4.3.2) and [SB02](5.6).

Section 12.2 . The definitions are standard. For the behavior of the order of convergence we followed [Gau11](4.2).

Section 12.3 . Muller's method was originally proposed for computing the zeros of polynomials [Mul56]. The method is presented in detail in [Sch89](5.3.3).

Section 12.4. Newton published his method in *Principia Mathematica* (1687) as a tool to solve Kepler's equation [NCW99] but had already discussed the method in 1969 [Gol77] as a tool to solve the equation $x^3 - 2x - 5 = 0$. Raphson [Rap90] presented the method in a simpler and more systematic way in 1690 and this is why the method is often called Newton-Raphson method. The proof of the theorem of local convergence was adapted from the proof for the secant method presented in [Gau11].

Section 12.5. The derivation of the super-linear rate of convergence of the secant method follows that in [Gau11, Sch89].

Section 12.6. The contraction mapping theorem and its proof can be found in introductory books on analysis and in any standard text on numerical analysis.

Section 12.7. The book by Ortega and Rheinboldt [OR70] has extensive material on systems of nonlinear equations, including existence results, mean value theorems and numerical methods.

Chapter 13

Numerical Methods for ODEs

In this chapter we study numerical methods for ordinary differential equations (ODEs) and systems of ODEs. We first review some basic theory for the ODE initial value problem before proceeding to give an overview of the main numerical methods for this important mathematical problem. The central focus of this chapter are the fundamental concepts of consistency, stability, and convergence.

13.1 The Initial Value Problem for ODEs

We will focus on numerical methods to approximate the solution of the following the first order, initial value problem (IVP):

$$\frac{dy(t)}{dt} = f(t, y(t)), \quad t_0 < t \leq T, \tag{13.1}$$

$$y(t_0) = \alpha. \tag{13.2}$$

Here, f is a given function of the independent variable t and the unknown function y. Geometrically, it represents the slope of the solution y. Often, t represents time but not necessarily. Equation (13.2), where α is a given constant, is called the initial condition. The problem is therefore to find a function $y(t)$ for t in some interval $[t_0, T]$ such that it is equal to α at $t = t_0$ and satisfies the ODE (13.1). Without loss of generality we will often take $t_0 = 0$, unless otherwise noted.

In the IVP (13.1)-(13.2), y and f may be vector-valued, i.e $y \in \mathbb{R}^d$ and $f(t, y) \in \mathbb{R}^d$, $d > 1$, in which case we have an IVP for a $d \times d$ system of first order ODEs.

The time derivative is also frequently denoted with a dot (especially in physics) or an apostrophe

$$\frac{dy}{dt} = \dot{y} = y'. \tag{13.3}$$

We will not write the dependence of y on t in the ODE; we will simply write $y' = f(t, y)$.

Example 13.1.

$$y' = t \sin y, \quad 0 < t \leq 2\pi, \tag{13.4}$$

$$y(0) = \alpha. \tag{13.5}$$

Example 13.2.

$$\begin{aligned} y_1' &= y_1 y_2 - y_1^2, \\ y_2' &= -y_2 + t^2 \cos y_1, \quad 0 < t \leq T, \end{aligned} \tag{13.6}$$

$$y_1(0) = \alpha_1, \quad y_2(0) = \alpha_2. \tag{13.7}$$

These two are examples of first order ODEs. Higher order ODEs can be written as first order systems by introducing new variables as we illustrate in the next two examples.

Example 13.3. *The Harmonic Oscillator.*

$$y'' + k^2 y = 0. \tag{13.8}$$

If we define

$$y_1 = y, \tag{13.9}$$

$$y_2 = y', \tag{13.10}$$

we get

$$\begin{aligned} y_1' &= y_2, \\ y_2' &= -k^2 y_1. \end{aligned} \tag{13.11}$$

Example 13.4.

$$y''' + 2yy'' + \cos y' + e^t = 0. \tag{13.12}$$

Introducing the variables

$$y_1 = y, \tag{13.13}$$
$$y_2 = y', \tag{13.14}$$
$$y_3 = y'', \tag{13.15}$$

we obtain the first order system:

$$\begin{aligned} y_1' &= y_2, \\ y_2' &= y_3, \\ y_3' &= -2y_1y_3 - \cos y_2 - e^t. \end{aligned} \tag{13.16}$$

If f does not depend explicitly on t we call the ODE (or the system of ODEs) **autonomous**. We can turn a non-autonomous system into an autonomous one by introducing t as a new variable.

Example 13.5. *Consider the ODE*

$$y' = \sin t - y^2. \tag{13.17}$$

If we define

$$y_1 = y, \tag{13.18}$$
$$y_2 = t, \tag{13.19}$$

we can write this ODE as the autonomous system

$$\begin{aligned} y_1' &= \sin y_2 - y_1^2, \\ y_2' &= 1. \end{aligned} \tag{13.20}$$

Continuity of f guarantees *local* existence of solutions but not uniqueness. A sufficient condition for uniqueness is given by the following definition.

Definition 13.1. *A function f defined on a domain $D \subset \mathbb{R} \times \mathbb{R}^d$ and with values in $\mathbb{R}^d$ is Lipschitz in y, if there is $L \geq 0$ such that*

$$\|f(t, y) - f(t, w)\| \leq L\|y - w\| \tag{13.21}$$

for all (t, y) and (t, w) in D and some norm $\|\cdot\|$ defined in $\mathbb{R}^d$. L is called the Lipschitz constant.

Note that if f is differentiable and D is convex (D contains the line segment joining any two points in D), the Lipschitz condition is equivalent to boundedness of f_y, i.e. if there is $L \geq 0$ such that

$$\left\| \frac{\partial f}{\partial y}(t,y) \right\| \leq L \tag{13.22}$$

for all $(t,y) \in D$. For a system, f_y is derivative matrix of f with respect to y (see Section 12.7). It is usually easier to check (13.22) than to use directly the Lipschitz condition (13.21).

We now state a fundamental theorem of local existence and uniqueness of solutions of the IVP (13.1)-(13.2).

Theorem 13.1 (Local Existence and Uniqueness)**.** *Let*

$$D = \{(t,y) : t_0 \leq t \leq T, \; \|y - \alpha\| \leq b\}. \tag{13.23}$$

If f is continuous in D and Lipschitz in y, with $\|f\|$ bounded by M, and $T \leq t_0 + b/M$, the IVP (13.1)-(13.2) has a unique solution for each $\alpha \in \mathbb{R}^d$.

We emphasize the local nature of this result. However, if f and f_y are continuous on an open set $D \subset \mathbb{R} \times \mathbb{R}^d$ then for every point $(t_0, \alpha) \in D$ there is a unique solution that can be continued up to the boundary of D.

Example 13.6.

$$\begin{aligned} y' &= y^{1/2}, \quad 0 < t, \\ y(0) &= 0. \end{aligned} \tag{13.24}$$

The partial derivative

$$\frac{\partial f}{\partial y} = \frac{1}{2}\, y^{-1/2} \tag{13.25}$$

is not continuous around 0*. While f is continuous, it is not Lipschitz in y. Clearly, $y \equiv 0$ is a solution of this initial value problem but so is $y(t) = \frac{1}{4}t^2$. There is no uniqueness of solution for this IVP.*

Example 13.7.

$$\begin{aligned} y' &= \frac{1}{2} y^2, \quad 0 < t \leq 3, \\ y(0) &= 1. \end{aligned} \tag{13.26}$$

We can integrate to obtain

$$y(t) = \frac{2}{2-t}, \tag{13.27}$$

which becomes unbounded as $\rightarrow 2$. *There is a unique solution only for* $t \in [0, 2)$ *and it cannot be continued past this interval. Note that*

$$\frac{\partial f}{\partial y}(t, y(t)) = y(t) \tag{13.28}$$

becomes unbounded at $t = 2$.

Integrating (13.1) from t_0 to t and using the initial condition (13.2), the IVP (13.1)-(13.2) can be reformulated as

$$y(t) = \alpha + \int_{t_0}^{t} f(s, y(s))ds. \tag{13.29}$$

This is an integral equation for the unknown function y. In particular, if f does not depend on y the problem is reduced to the approximation of the definite integral

$$\int_{t_0}^{t} f(s)ds, \tag{13.30}$$

for which a numerical quadrature can be applied.

The numerical methods we will study in this chapter deal with the more general and important case when f depends on the unknown y. These methods produce an approximation of the exact solution of the IVP (assuming uniqueness) at a set of discrete points

$$0 = t_0 < t_1 < \ldots < t_N \leq T. \tag{13.31}$$

For simplicity in the presentation, we will assume these points are equispaced,

$$t_n = n\Delta t, \quad n = 0, 1, \ldots, N \text{ and } \Delta t = T/N, \tag{13.32}$$

but they do not have to be. Δt is called the *step size*.

We will write a numerical method for an IVP as an algorithm to go from one discrete time, t_n, to the next one, t_{n+1}. With that in mind, it is convenient to integrate (13.1) from t_n to t_{n+1}:

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t))dt. \tag{13.33}$$

This equation provides a useful framework for the construction of some numerical methods employing quadratures.

13.2 A First Look at Numerical Methods

Let us denote by y^n the approximation[1] produced by the numerical method of the exact solution at t_n, i.e.

$$y^n \approx y(t_n). \tag{13.34}$$

Starting from (13.33), if we approximate the integral using only f evaluated at the lower integration limit

$$\int_{t_n}^{tn+1} f(t, y(t))dt \approx f(t_n, y(t_n))(t_{n+1} - t_n) = f(t_n, y(t_n))\,\Delta t \tag{13.35}$$

and replace $f(t_n, y(t_n))$ by $f(t_n, y^n)$, we obtain the so called **forward Euler method**:

$$y^0 = \alpha, \tag{13.36}$$
$$y^{n+1} = y^n + \Delta t f(t_n, y^n), \quad n = 0, 1, \ldots N-1. \tag{13.37}$$

This provides an *explicit* formula to advance from one time step to the next. The approximation y^{n+1} at the future step only depends on the approximation y^n at the current step. The forward Euler method is an example of an *explicit* **one-step method**.

Example 13.8. *Consider the initial value problem:*

$$y' = -\frac{1}{5}y - e^{-t/5}\sin t, \quad 0 < t \le 2\pi, \tag{13.38}$$
$$y(0) = 1. \tag{13.39}$$

To use the forward Euler method for this problem we start with $y^0 = 1$ and proceed with the iteration (13.37) with $f(t_n, y^n) = -\frac{1}{5}y^n - e^{-t_n/5}\sin t_n$. Figure 13.1 shows the forward Euler approximation with $\Delta t = 2\pi/20$ and the exact solution, $y(t) = e^{-t/5}\cos t$.

Note we just compute an approximation y^n of the solution at the discrete points $t_n = n\Delta t$, for $n = 0, 1, \ldots, N$. However, the numerical approximation is often plotted using a continuous curve that passes through all the points (t_n, y^n), $n = 0, 1, \ldots, N$ (i.e. an interpolant).

[1] We use a superindex for the time approximation, instead of the most commonly employed subindex notation, to facilitate the transition to numerical methods for PDEs.

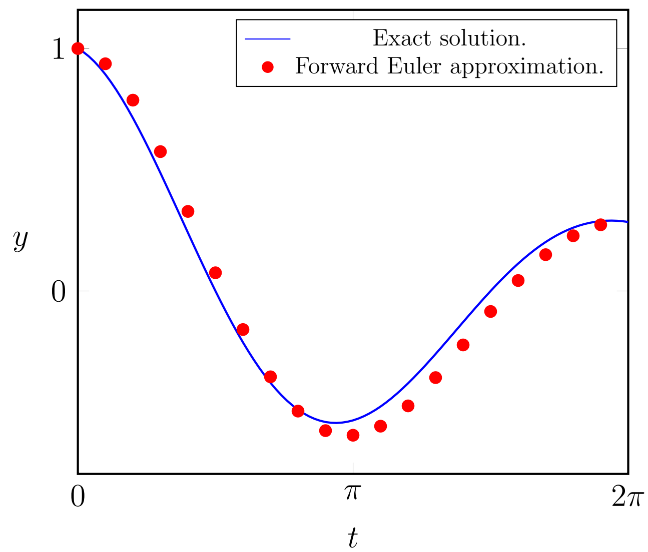

Figure 13.1: Forward Euler approximation with $\Delta t = 2\pi/20$ and exact solution of the IVP (13.38)-(13.39).

If we approximate the integral in (13.33) employing only the upper limit of integration and replace $f(t_{n+1}, y(t_{n+1}))$ by $f(t_{n+1}, y^{n+1})$ we obtain the **backward Euler method**:

$$y^0 = \alpha, \tag{13.40}$$

$$y^{n+1} = y^n + \Delta t f(t_{n+1}, y^{n+1}), \quad n = 0, 1, \dots N-1. \tag{13.41}$$

Note that now y^{n+1} is defined *implicitly* in (13.41). Thus, to update the approximation we need to solve this equation for y^{n+1}, for each $n = 0, \dots, N-1$. If f is nonlinear, we would generally need to employ a numerical method to solve $y - F(y) = 0$, where $F(y) = \Delta t f(t_{n+1}, y) + y^n$. This is equivalent to finding a fixed point of F. By the contraction mapping theorem (Theorem 12.2), we can guarantee a unique solution y^{n+1} if $\Delta t L < 1$ (where L is the Lipschitz constant of f). In practice, an approximation to y^{n+1} is obtained by performing a limited number of fixed point iterations $y^{(k+1)} = F(y^{(k)})$, $k = 0, 1, \dots, K$, or by a few iterations of Newton's method for $y - F(y) = 0$ with y^n as initial guess. The backward Euler method is an *implicit* **one-step method**.

We can employ more accurate quadratures as the basis for our numerical methods. For example, if we use the trapezoidal rule

$$\int_{t_n}^{tn+1} f(t, y(t))dt \approx \frac{\Delta t}{2}\left[f(t_n, y(t_n)) + f(t_{n+1}, y(t_{n+1}))\right] \tag{13.42}$$

and proceed as before, we get the trapezoidal rule method:

$$y^0 = \alpha, \tag{13.43}$$

$$y^{n+1} = y^n + \frac{\Delta t}{2}\left[f(t_n, y^n) + f(t_{n+1}, y^{n+1})\right], \quad n = 0, 1, \ldots N-1. \tag{13.44}$$

Like the backward Euler method, this is an implicit one-step method.

We will see later an important class of one-step methods, known as **Runge-Kutta** (RK) methods, which use intermediate approximations to the derivative (i.e. to f) and a corresponding quadrature. For example, if we employ the midpoint rule quadrature and the approximation

$$f\left(t_{n+1/2}, y(t_{n+1/2})\right) \approx f\left(t_{n+1/2}, y^n + \frac{\Delta t}{2} f(t_n, y^n)\right), \tag{13.45}$$

where $t_{n+1/2} = t_n + \Delta t/2$, we obtain the explicit midpoint Runge-Kutta method

$$y^{n+1} = y^n + \Delta t f\left(t_{n+1/2}, y^n + \frac{\Delta t}{2} f(t_n, y^n)\right). \tag{13.46}$$

Another possibility is to approximate the integrand f in (13.33) by an interpolating polynomial of f evaluated at m previous approximations y^n, y^{n-1}, ..., $y^{n-(m-1)}$. To simplify the notation, let us write

$$f^n = f(t_n, y^n), \quad f^{n-1} = f(t_{n-1}, y^{n-1}), \quad \text{etc.} \tag{13.47}$$

For example, if we replace f in $[t_n, t_{n+1}]$ by the linear polynomial p_1 interpolating (t_n, f^n) and (t_{n-1}, f^{n-1}),

$$p_1(t) = \frac{(t - t_{n-1})}{\Delta t} f^n - \frac{(t - t_n)}{\Delta t} f^{n-1} \tag{13.48}$$

we get

$$\int_{t_n}^{tn+1} f(t, y(t))dt \approx \int_{t_n}^{tn+1} p_1(t)dt = \frac{\Delta t}{2}[3f^n - f^{n-1}] \tag{13.49}$$

and the corresponding numerical method is

$$y^{n+1} = y^n + \frac{\Delta t}{2}\left[3f^n - f^{n-1}\right], \quad n = 1, 2, \ldots N-1. \tag{13.50}$$

This is a two-step method because to determine y^{n+1} we need the approximations at the previous two steps, y^n and y^{n-1}. Numerical methods that require approximations of more than one step to determine the approximation at the future step are called **multistep methods**. Note that to start using (13.50), i.e. $n = 1$, we need y^0 and y^1. For y^0 we use the initial condition, $y^0 = \alpha$, and for y^1 we can employ a one-step method of *comparable accuracy*. All multistep methods require this *initialization process* where approximations to $y^1, \ldots, y^{m-1}$ have to be generated with one-step methods before we can apply the multistep formula.

Numerical methods can also be constructed by approximating y' using finite differences or interpolation. For example, the central difference approximation

$$y'(t_n) \approx \frac{y(t_n + \Delta t) - y(t_n - \Delta t)}{2\Delta t} \approx \frac{y^{n+1} - y^{n-1}}{2\Delta t} \tag{13.51}$$

produces the two-step method

$$y^{n+1} = y^{n-1} + 2\Delta t f^n. \tag{13.52}$$

If we approximate $y'(t_{n+1})$ by the derivative of the polynomial interpolating y^{n+1} and some previous approximations we obtain a class of methods known as **backward differentiation formula (BDF)** methods. For instance, let $p_2 \in \mathbb{P}_2$ be the polynomial that interpolates (t_{n-1}, y^{n-1}), (t_n, y^n), and (t_{n+1}, y^{n+1}). Then

$$y'(t_{n+1}) \approx p_2'(t_{n+1}) = \frac{3y^{n+1} - 4y^n + y^{n-1}}{2\Delta t}, \tag{13.53}$$

which gives the BDF method

$$\frac{3y^{n+1} - 4y^n + y^{n-1}}{2\Delta t} = f^{n+1}, \quad n = 1, 2, \ldots N-1. \tag{13.54}$$

Note that this is an *implicit* **multistep method**.

13.3 One-Step and Multistep Methods

As we have seen, there are two broad classes of methods for the IVP (13.1)-(13.2): one-step methods and multistep methods.

Explicit one-step methods can be written in the general form

$$y^{n+1} = y^n + \Delta t\, \Phi(t_n, y^n, \Delta t) \tag{13.55}$$

for some continuous function Φ. For example $\Phi(t, y, \Delta t) = f(t, y)$ for the forward Euler method and $\Phi(t, y, \Delta t) = f\left(t + \frac{\Delta t}{2}, y + \frac{\Delta t}{2} f(t, y)\right)$ for the midpoint RK method. For an implicit one-step method Φ is also a function of y^{n+1}. Φ is call the increment function.

A general, m-step ($m > 1$) *linear* multistep method has the form

$$\begin{aligned} a_m y^{n+1} + a_{m-1} y^n + \ldots + a_0 y^{n-(m-1)} &= \\ \Delta t \left[b_m f^{n+1} + b_{m-1} f^n + \ldots + b_0 f^{n-(m-1)} \right], \end{aligned} \tag{13.56}$$

for some coefficients $a_0, a_1, \ldots, a_m$ and $b_0, b_1, \ldots, b_m$ with $a_m \neq 0$. If $b_m \neq 0$ the multistep is implicit otherwise it is explicit. This class of methods are called linear because the right hand size in (13.56) is a linear function of the values $f^j = f(t_j, y^j)$ for $j = n - (m-1), \ldots, n+1$. There are also nonlinear multistep methods, where the hand side is a nonlinear function of f, and which are useful for some specialized IVP's. We will limit the discussion here to the more widely used *linear* multistep methods and simply call them multistep methods.

Shifting the index by $m-1$, we can also write an m-step ($m > 1$) method as

$$\sum_{j=0}^{m} a_j y^{n+j} = \Delta t \sum_{j=0}^{m} b_j f^{n+j}. \tag{13.57}$$

13.4 Local and Global Error

At each time step in the numerical approximation of an IVP there is an error associated with evolving the solution from t_n to t_{n+1} with the numerical method instead of using the ODE (or the integral equation). There is also an error due to employing y^n instead of $y(t_n)$ as the starting point. After several time steps, these *local errors* accumulate in the *global error* of the approximation. Let us make the definition of these errors more precise.

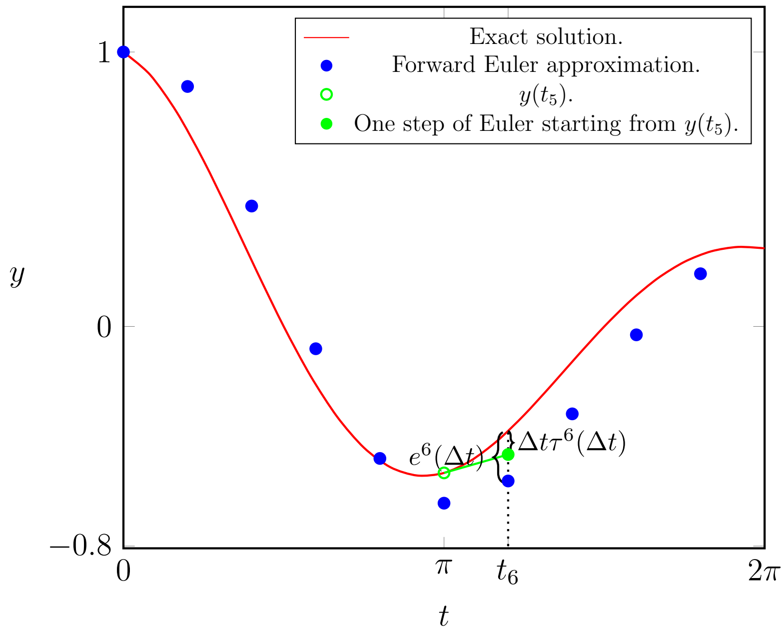

Figure 13.2: Global and local discretization error of the forward Euler method at t_6 with $\Delta t = 2\pi/10$ for the IVP (13.38)-(13.39).

Definition 13.2. *The local discretization or local truncation error $\tau^{n+1}(\Delta t)$ at t_{n+1} is given by*

$$\tau^{n+1}(\Delta t) = \frac{y(t_{n+1}) - \tilde{y}^{n+1}}{\Delta t}, \tag{13.58}$$

where $\tilde{y}^{n+1}$ is computed by doing one step of the numerical method starting with the exact value $y(t_n)$ for a one-step method and with $y(t_n), y(t_{n-1}), \ldots,$ $y(t_{n-m+1})$ for an m-step method.

Definition 13.3. *The global error $e^n(\Delta t)$ at t_n is given by*

$$e^n(\Delta t) = y(t_n) - y^n, \tag{13.59}$$

where $y(t_n)$ and y^n are the exact solution of the IVP and the numerical approximation at t_n, respectively.

Figure 13.2 shows the global error and the local discretization error times Δt at $t_6 = 6\Delta t$ for the forward Euler method applied to the IVP (13.38)-(13.39) with $\Delta t = 2\pi/10$. Note that the $\Delta t\tau^6$ is the local error made by taking only one step of the numerical method starting from the exact initial condition $y(t_5)$ whereas $e^6(\Delta t)$ is the global error of the approximation after six time steps starting from $y^0 = \alpha = 1$.

For an explicit one-step method the local truncation error is simply

$$\tau^{n+1}(\Delta t) = \frac{y(t_{n+1}) - [y(t_n) + \Delta t\, \Phi(t_n, y(t_n), \Delta t)]}{\Delta t}. \tag{13.60}$$

That is,

$$\tau^{n+1}(\Delta t) = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - \Phi(t_n, y(t_n), \Delta t) \tag{13.61}$$

For an explicit multistep method ($b_m = 0$),

$$\tau^{n+m}(\Delta t) = \frac{y(t_{n+m}) - \tilde{y}^{n+m}}{\Delta t}, \tag{13.62}$$

where

$$a_m \tilde{y}^{n+m} = -\sum_{j=0}^{m-1} a_j y(t_{n+j}) + \Delta t \sum_{j=0}^{m} b_j f(t_{n+j}, y(t_{n+j})). \tag{13.63}$$

Substituting (13.63) into (13.62) we get

$$\tau^{n+m}(\Delta t) = \frac{1}{\Delta t} \sum_{j=0}^{m} a_j y(t_{n+j}) - \sum_{j=0}^{m} b_j f(t_{n+j}, y(t_{n+j})), \tag{13.64}$$

where we assumed, without loss of generality, that $a_m = 1$. Since $y' = f(t, y)$, we also have

$$\tau^{n+m}(\Delta t) = \frac{1}{\Delta t} \sum_{j=0}^{m} a_j y(t_{n+j}) - \sum_{j=0}^{m} b_j y'(t_{n+j}). \tag{13.65}$$

For implicit methods we can also use (13.64) for the local truncation error because it is (13.62) up to a multiplicative factor. Indeed, let

$$\tilde{\tau}^{n+m}(\Delta t) = \frac{1}{\Delta t} \sum_{j=0}^{m} a_j y(t_{n+j}) - \sum_{j=0}^{m} b_j f(t_{n+j}, y(t_{n+j})). \tag{13.66}$$

Then,

$$\sum_{j=0}^{m} a_j y(t_{n+j}) = \Delta t \sum_{j=0}^{m} b_j f(t_{n+j}, y(t_{n+j})) + \Delta t \; \tilde{\tau}^{n+m}(\Delta t). \tag{13.67}$$

On the other hand $\tilde{y}^{n+m}$ in the definition of the local error is computed using

$$a_m \tilde{y}^{n+m} + \sum_{j=0}^{m-1} a_j y(t_{n+j}) = \Delta t \left[b_m f(t_{n+m}, \tilde{y}_{n+m}) + \sum_{j=0}^{m-1} b_j f(t_{n+j}, y(t_{n+j})) \right]. \tag{13.68}$$

Subtracting (13.68) to (13.67) and using $a_m = 1$ we get

$$\begin{aligned} y(t_{n+m}) - \tilde{y}^{n+m} &= \Delta t \, b_m \left[f(t_{n+m}, y(t_{n+m})) - f(t_{n+m}, \tilde{y}^{n+m}) \right] \\ &\quad + \Delta t \; \tilde{\tau}^{n+k}(\Delta t). \end{aligned} \tag{13.69}$$

Assuming f is a scalar C^1 function, from the mean value theorem we have

$$f(t_{n+m}, y(t_{n+m})) - f(t_{n+m}, \tilde{y}^{n+m}) = \frac{\partial f}{\partial y}(t_{n+m}, \eta) \left[y(t_{n+m}) - \tilde{y}^{n+m} \right],$$

for some η between $y(t_{n+m})$ and $\tilde{y}^{n+m}$. Substituting this into (13.69) and solving for $y(t_{n+m}) - \tilde{y}_{n+m}$ we get

$$\tau^{n+m}(\Delta t) = \left[1 - \Delta t \, b_m \frac{\partial f}{\partial y}(t_{n+m}, \eta) \right]^{-1} \tilde{\tau}^{n+m}(\Delta t). \tag{13.70}$$

If f is a vector valued function (a system of ODEs), the partial derivative in (13.70) is a derivative matrix. A similar argument can be made for an implicit one-step method if the increment function Φ is Lipschitz in y and we use absolute values in the errors. Thus, (13.61) and (13.64) can be used as the definition of the local truncation error for one-step and multi-step methods, respectively. With this definition, *we can view the local truncation error as a measure of how well the exact solution of the IVP satisfies the numerical method formula.*

Example 13.9. *The local truncation error for the forward Euler method is*

$$\tau^{n+1}(\Delta t) = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - f(t_n, y(t_n)). \tag{13.71}$$

Taylor expanding the exact solution around t_n we have

$$y(t_{n+1}) = y(t_n) + \Delta t\, y'(t_n) + \frac{1}{2}(\Delta t)^2 y''(\eta_n) \tag{13.72}$$

for some η_n between t_n and t_{n+1}. Using $y' = f$ and substituting (13.72) into (13.71) we get

$$\tau^{n+1}(\Delta t) = \frac{1}{2} y''(\eta_n)\Delta t. \tag{13.73}$$

Thus, assuming the exact solution is C^2, the local truncation error of the forward Euler method is $O(\Delta t)$.

To simplify notation we will henceforth write $O(\Delta t)^k$ instead of $O((\Delta t)^k)$.

Example 13.10. *For the explicit midpoint Runge-Kutta method we have*

$$\tau^{n+1}(\Delta t) = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - f\left(t_{n+1/2}, y(t_n) + \frac{\Delta t}{2} f(t_n, y(t_n))\right). \tag{13.74}$$

Taylor expanding f around $(t_n, y(t_n))$ we obtain

$$\begin{aligned} f\left(t_{n+1/2}, y(t_n) + \frac{\Delta t}{2} f(t_n, y(t_n))\right) &= f(t_n, y(t_n)) \\ &+ \frac{\Delta t}{2}\frac{\partial f}{\partial t}(t_n, y(t_n)) \\ &+ \frac{\Delta t}{2} f(t_n, y(t_n)) \frac{\partial f}{\partial y}(t_n, y(t_n)) \\ &+ O(\Delta t)^2. \end{aligned} \tag{13.75}$$

But $y' = f$, $y'' = f'$ and

$$f' = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y} y' = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y} f. \tag{13.76}$$

Therefore

$$f\left(t_{n+1/2}, y(t_n) + \frac{\Delta t}{2} f(t_n, y(t_n))\right) = y'(t_n) + \frac{1}{2}\Delta t\, y''(t_n) + O(\Delta t)^2. \tag{13.77}$$

On the other hand

$$y(t_{n+1}) = y(t_n) + \Delta t\, y'(t_n) + \frac{1}{2}(\Delta t)^2 y''(t_n) + O(\Delta t)^3. \tag{13.78}$$

Substituting (13.77) and (13.78) into (13.74) we get

$$\tau^{n+1}(\Delta t) = O(\Delta t)^2. \tag{13.79}$$

In the previous two examples the methods are one-step. We now obtain the local truncation error for a particular multistep method.

Example 13.11. *Let us consider the 2-step Adams-Bashforth method (13.50). We have*

$$\tau^{n+2}(\Delta t) = \frac{y(t_{n+2}) - y(t_{n+1})}{\Delta t} - \frac{1}{2}\left[3f(t_{n+1}, y(t_{n+1})) - f(t_n, y(t_n))\right] \quad (13.80)$$

and using $y' = f$

$$\tau^{n+2} = \frac{y(t_{n+2}) - y(t_{n+1})}{\Delta t} - \frac{1}{2}\left[3y'(t_{n+1}) - y'(t_n)\right]. \quad (13.81)$$

Taylor expanding $y(t_{n+2})$ and $y'(t_n)$ around t_{n+1} we have

$$y(t_{n+2}) = y(t_{n+1}) + y'(t_{n+1})\Delta t + \frac{1}{2}y''(t_{n+1})(\Delta t)^2 + O(\Delta t)^3, \quad (13.82)$$

$$y'(t_n) = y'(t_{n+1}) - y''(t_{n+1})\Delta t + O(\Delta t)^2. \quad (13.83)$$

Substituting these expressions into (13.81) we get

$$\begin{aligned} \tau^{n+2}(\Delta t) &= y'(t_{n+1}) + \frac{1}{2}y''(t_{n+1})\Delta t \\ &\quad - \frac{1}{2}\left[2y'(t_{n+1}) - \Delta t y''(t_{n+1})\right] + O(\Delta t)^2 \\ &= O(\Delta t)^2. \end{aligned} \quad (13.84)$$

13.5 Order of a Method and Consistency

As we have seen, if the exact solution of the IVP $y' = f(t, y)$, $y(0) = \alpha$ is sufficiently smooth, the local truncation error can be expressed as $O(\Delta t)^p$, for some positive integer p and *sufficiently small* Δt.

Definition 13.4. *A numerical method for the initial value problem (13.1)-(13.2) is said to be of order p if its local truncation error is $O(\Delta t)^p$.*

Euler's method is order 1 or first order. The midpoint Runge-Kutta method and the 2-step Adams-Bashforth method are order 2 or second order.

As mentioned above, the local truncation error can be viewed as a measure of how well the exact solution of the IVP satisfies the numerical method formula. Thus, a natural requirement is that the numerical method formula approaches $y' = f(t, y)$ as $\Delta t \to 0$ and not some other equation. This motivates the following definition.

Definition 13.5. *We say that a numerical method is consistent (with the ODE of the IVP) if*

$$\lim_{\Delta t \to 0} \left(\max_{1 \le n \le N} |\tau^n(\Delta t)| \right) = 0. \tag{13.85}$$

Equivalently, if the method is at least of order 1.

For one-step methods, we have

$$\tau^{n+1}(\Delta t) = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - \Phi(t_n, y(t_n), \Delta t). \tag{13.86}$$

Since $[y(t_{n+1}) - y(t_n)]/\Delta t$ converges to $y'(t_n)$ as $\Delta t \to 0$ and $y' = f(t, y)$, a one-step method is consistent with the ODE $y' = f(t, y)$ if and only if[2]

$$\Phi(t, y, 0) = f(t, y). \tag{13.87}$$

To find a consistency condition for a multistep method, we expand $y(t_{n+j})$ and $y'(t_{n+j})$ around t_n

$$y(t_{n+j}) = y(t_n) + (j\Delta t)y'(t_n) + \frac{1}{2!}(j\Delta t)^2 y''(t_n) + \ldots \tag{13.88}$$

$$y'(t_{n+j}) = y'(t_n) + (j\Delta t)y''(t_n) + \frac{1}{2!}(j\Delta t)^2 y'''(t_n) + \ldots \tag{13.89}$$

and substituting in the definition of the local error (13.64) we get that a multistep method is consistent if and only if

$$a_0 + a_1 + \ldots + a_m = 0, \tag{13.90}$$

$$a_1 + 2a_2 + \ldots m a_m = b_0 + b_1 + \ldots + b_m. \tag{13.91}$$

All the methods that we have seen so far are consistent (with $y' = f(t, y)$).

13.6 Convergence

A basic requirement of the approximations generated by a numerical method is that they get better and better as we take smaller step sizes. That is, we want the approximations to approach the exact solution at each fixed $t = n\Delta t$ as $\Delta t \to 0$.

[2] We assume Φ is continuous as stated in the definition of one-step methods.

Definition 13.6. *A numerical method for the IVP (13.1)-(13.2) is convergent if the global error converges to zero as $\Delta t \to 0$ with $t = n\Delta t$ fixed i.e.*

$$\lim_{\substack{\Delta t \to 0 \\ n\Delta t = t}} [y(n\Delta t) - y^n] = 0. \tag{13.92}$$

Note that for a multistep method the initialization values $y^1, \ldots, y^{m-1}$ must converge to $y(0) = \alpha$ as $\Delta t \to 0$.

If we consider a one-step method and the definition (13.61) of the local truncation error, the exact solution satisfies

$$y(t_{n+1}) = y(t_n) + \Delta t\, \Phi(t_n, y(t_n), \Delta t) + \Delta t\ \tau^{n+1}(\Delta t) \tag{13.93}$$

while the approximation is given by

$$y^{n+1} = y^n + \Delta t\, \Phi(t_n, y^n, \Delta t). \tag{13.94}$$

Subtracting (13.94) from (13.93) we get a difference equation for the global error

$$e^{n+1}(\Delta t) = e^n(\Delta t) + \Delta t\, [\Phi(t_n, y(t_n), \Delta t) - \Phi(t_n, y^n, \Delta t)] + \Delta t\ \tau^{n+1}(\Delta t). \tag{13.95}$$

The growth of the global error as we take more and more time steps is linked not only to the local truncation error but also to the increment function Φ. To have a controlled error growth, we need an additional assumption on Φ, namely that it is Lipschitz in y, i.e. there is $L \geq 0$ such that

$$|\Phi(t, y, \Delta t) - \Phi(t, w, \Delta t)| \leq L|y - w| \tag{13.96}$$

for all $t \in [0, T]$ and y and w in the relevant domain of existence of the solution. Recall that for a consistent one-step method $\Phi(t, y, 0) = f(t, y)$ and we assume $f(t, y)$ is Lipschitz in y to guarantee existence and uniqueness of the IVP. Thus, the Lipschitz assumption on Φ is somewhat natural.

Taking absolute values (or norms in the vector case) in (13.95), using the triangle inequality and (13.96) we obtain

$$|e^{n+1}(\Delta t)| \leq (1 + \Delta t L)|e^n(\Delta t)| + \Delta t\ |\tau^{n+1}(\Delta t)|. \tag{13.97}$$

For a method of order p, $|\tau^{n+1}(\Delta t)| \leq C(\Delta t)^p$, for sufficiently small Δt. Therefore,

$$
\begin{aligned}
|e^{n+1}(\Delta t)| &\leq (1+\Delta t L)|e^n(\Delta t)| + C(\Delta t)^{p+1} \\
&\leq (1+\Delta t L)\left[(1+\Delta t L)|e^{n-1}(\Delta t)| + C(\Delta t)^{p+1}\right] + C(\Delta t)^{p+1} \\
&\leq \dots \\
&\leq (1+\Delta t L)^{n+1}|e^0(\Delta t)| + C(\Delta t)^{p+1}\sum_{j=0}^{n}(1+\Delta t L)^j
\end{aligned}
\tag{13.98}
$$

and summing up the geometric sum we get

$$
|e^{n+1}(\Delta t)| \leq (1+\Delta t L)^{n+1}|e^0(\Delta t)| + \left[\frac{(1+\Delta t L)^{n+1}-1}{\Delta t L}\right] C(\Delta t)^{p+1}. \tag{13.99}
$$

Now $1 + t \leq e^t$ for all real t and consequently $(1 + \Delta t L)^n \leq e^{n\Delta t L} = e^{tL}$. Since $e^0(\Delta t) = 0$,

$$
|e^n(\Delta t)| \leq \left[\frac{e^{tL}-1}{\Delta t L}\right] C(\Delta t)^{p+1} < \frac{C}{L} e^{tL}\ (\Delta t)^p. \tag{13.100}
$$

Therefore, the global error goes to zero like $(\Delta t)^p$ as $\Delta t \to 0$, keeping $t = n\Delta t$ fixed. We have thus established the following important result.

Theorem 13.2. *A consistent ($p \geq 1$) one-step method with a Lipschitz in y increment function Φ is convergent.*

The Lipschitz condition on Φ allowed us to bound the growth of the local truncation error as more and more time steps are taken. This controlled error growth, which is called *numerical stability*, was achieved through the bound

$$
(1 + \Delta t L)^n \leq \text{constant}. \tag{13.101}
$$

Example 13.12. *The forward Euler method is order 1 and hence consistent. Since $\Phi = f$ and we are assuming that f is Lipschitz in y, by the previous theorem the forward Euler method is convergent.*

Example 13.13. *Prove that the midpoint Runge-Kutta method is convergent (assuming f is Lipschitz in y).*

The increment function in this case is

$$\Phi(t, y, \Delta t) = f\left(t + \frac{\Delta t}{2}, y + \frac{\Delta t}{2} f(t, y)\right). \tag{13.102}$$

Therefore

$$\begin{aligned} |\Phi(t, y, \Delta t) - \Phi(t, w, \Delta t)| &= \left| f\left(t + \frac{\Delta t}{2}, y + \frac{\Delta t}{2} f(t, y)\right) \right. \\ &\qquad \left. - f\left(t + \frac{\Delta t}{2}, w + \frac{\Delta t}{2} f(t, w)\right) \right| \\ &\leq L \left| y + \frac{\Delta t}{2} f(t, y) - w - \frac{\Delta t}{2} f(t, w) \right| \\ &\leq L\,|y - w| + \frac{\Delta t}{2} L\,|f(t, y) - f(t, w)| \\ &\leq \left(1 + \frac{\Delta t}{2} L\right) L\,|y - w| \leq \tilde{L}\,|y - w|. \end{aligned} \tag{13.103}$$

where $\tilde{L} = (1 + \frac{\Delta t_0}{2} L)L$ and $\Delta t \leq \Delta t_0$, i.e. for sufficiently small Δt. This proves that Φ is Lipschitz in y and since the midpoint Runge-Kutta method is of order 2, it is consistent and therefore convergent.

The exact solution of the IVP at t_{n+1} is determined uniquely from its value at t_n. In contrast, multistep methods use not only y^n but also $y^{n-1}, \ldots, y^{n-(m-1)}$ to produce y^{n+1}. The use of more than one step introduces some peculiarities to the theory of stability and convergence of multistep methods. We will cover these topics separately after we take a look at the most widely used class of one-step methods: the Runge-Kutta methods.

13.7 Runge-Kutta Methods

Runge-Kutta (RK) methods are based on replacing the integral in

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t))dt \tag{13.104}$$

with a quadrature formula and using accurate enough intermediate approximations for the integrand f (the derivative of y). To illustrate their derivation, we consider the simplest case with two quadrature nodes, t_n and another point $t_n + c\Delta t$ with $c \in (0, 1]$, so that the method has the form

$$\begin{aligned} K_1 &= f(t_n, y^n), && (13.105)\\ K_2 &= f(t_n + c\Delta t, y_n + a\Delta t K_1), && (13.106)\\ y^{n+1} &= y_n + \Delta t\left[b_1 K_1 + b_2 K_2\right]. && (13.107) \end{aligned}$$

The constants c, a and the quadrature weights b_1 and b_2 are going to be determined so that the method has the highest order possible. Note that K_1 and K_2 are approximations to the derivative of y at t_n and $t_n + c\Delta t$, respectively. This is a two-stage method. In the first stage, K_1 is computed and in the second stage K_2 is obtained using K_1. The last step, Eq. (13.107), employs the selected quadrature to update the approximation.

Recall the definition of the local truncation error (13.61). We have

$$\tau^{n+1}(\Delta t) = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - \left[b_1 f(t_n, y(t_n)) + b_2 K_2(t_n, y(t_n))\right], \quad (13.108)$$

where

$$K_2(t_n, y(t_n)) = f\left(t_n + c\Delta t, y(t_n) + a\Delta t f(t_n, y(t_n))\right). \quad (13.109)$$

We first Taylor expand $y(t_{n+1})$ around t_n:

$$\begin{aligned} y(t_{n+1}) &= y + \Delta t y' + \frac{1}{2}(\Delta t)^2 y'' + O(\Delta t)^3 \\ &= y + \Delta t f + \frac{1}{2}(\Delta_t)^2 (f_t + f f_y) + (\Delta t)^3, \end{aligned} \quad (13.110)$$

where, on the right hand side, all instances of y and its derivatives, and f and its derivatives, are evaluated at t_n and $(t_n, y(t_n))$, respectively. We also need to Taylor expand the right hand side of (13.109) around $(t_n, y(t_n))$:

$$K_2(t_n, y(t_n)) = f + c\Delta t f_t + a\Delta t f f_y + O(\Delta t)^2. \quad (13.111)$$

Substituting (13.110) and (13.111) into (13.108) we get

$$\begin{aligned} \tau^{n+1}(\Delta t) = {} & (1 - b_1 - b_2) f + \Delta t \left(\frac{1}{2} - b_2 c\right) f_t + \Delta t \left(\frac{1}{2} - b_2 a\right) f_y \\ & + O(\Delta t)^2. \end{aligned} \quad (13.112)$$

Thus, to have second order we need

$$b_1 + b_2 = 1, \tag{13.113}$$
$$cb_2 = 1/2, \tag{13.114}$$
$$ab_2 = 1/2. \tag{13.115}$$

It is also clear that a higher order is unattainable with the four parameters (the $O(\Delta t)^3$ in (13.110) involves some partial derivatives of f that cannot be matched with those in the $O(\Delta t)^2$ term of $b_1 K1 + b_2 K_2$). This system of three equations in four unknowns has an infinite number of solutions. For any value $b_2 \neq 0$ there correspond one solution. For example, with $b_2 = 1/2$ we get $b_1 = 1/2$ (trapezoidal rule quadrature), $c = 1$, and $a = 1$, which corresponds to the RK method known as the *improved Euler method* or *Heun method*:

$$K_1 = f(t_n, y^n), \tag{13.116}$$
$$K_2 = f(t_n + \Delta t, y^n + \Delta t K_1), \tag{13.117}$$
$$y^{n+1} = y^n + \Delta t \left[\frac{1}{2} K_1 + \frac{1}{2} K_2 \right]. \tag{13.118}$$

If we take $b_2 = 1$ wet get $b_1 = 0$, $c = 1/2$, $a = 1/2$ we obtain the midpoint RK method (13.46) which can now be written as

$$K_1 = f(t_n, y^n), \tag{13.119}$$
$$K_2 = f\left(t_n + \frac{\Delta t}{2}, y^n + \frac{\Delta t}{2} K_1 \right), \tag{13.120}$$
$$y^{n+1} = y^n + \Delta t K_2. \tag{13.121}$$

Obtaining the order of an RK method using Taylor expansions becomes a long, tedious process because the number of terms in the derivatives of f grows rapidly ($y' = f, y'' = f_t + f_y f, y''' = f_{tt} + 2 f_{ty} f + f_{yy} f^2 + f_y f_t + f_y^2 f$, etc.). There is a beautiful alternative approach based on graph theory to obtain the order of an RK method due to J.C. Butcher but we will not discuss it here.

One of the most popular RK method is the following 4-stage (and fourth order) explicit RK, known as the *classical fourth order RK*:

$$
\begin{aligned}
K_1 &= f(t_n, y^n), \\
K_2 &= f(t_n + \frac{1}{2}\Delta t, y^n + \frac{1}{2}\Delta t K_1), \\
K_3 &= f(t_n + \frac{1}{2}\Delta t, y^n + \frac{1}{2}\Delta t K_2), \\
K_4 &= f(t_n + \Delta t, y^n + \Delta t K_3), \\
y^{n+1} &= y^n + \frac{\Delta t}{6}\left[K_1 + 2K_2 + 2K_3 + K_4\right].
\end{aligned}
\tag{13.122}
$$

A general s-stage RK method can be written as

$$
\begin{aligned}
K_1 &= f\left(t_n + c_1\Delta t,\, y^n + \Delta t\sum_{j=1}^{s} a_{1j}K_j\right), \\
K_2 &= f\left(t_n + c_2\Delta t,\, y^n + \Delta t\sum_{j=1}^{s} a_{2j}K_j\right), \\
&\;\vdots \\
K_s &= f\left(t_n + c_s\Delta t,\, y^n + \Delta t\sum_{j=1}^{s} a_{sj}K_j\right), \\
y^{n+1} &= y^n + \Delta t\sum_{j=1}^{s} b_j K_j.
\end{aligned}
\tag{13.123}
$$

RK methods are determined by the constants $c_1, \ldots, c_s$ that specify the quadrature nodes, the coefficients $a_{1j}, \ldots, a_{sj}$ for $j = 1, \ldots, s$ used to obtain approximations of the solution at the intermediate quadrature points, and the quadrature coefficients $b_1, \ldots, b_s$. Consistent RK methods need to satisfy the condition

$$
\sum_{j=1}^{s} b_j = 1. \tag{13.124}
$$

Additionally, the following simplifying condition is assumed

$$
\sum_{j=1}^{s} a_{ij} = c_i, \quad i = 1, \ldots, s. \tag{13.125}
$$

This condition arises by requiring that the method preserves the non-autonomous to autonomous transformation ($t' = 1$) illustrated in Example 13.5.

To define an RK method it is enough to specify the coefficients c_j, a_{ij} and b_j for $i, j = 1, \ldots, s$. These coefficients are often displayed in a table, called the Butcher tableau (after J.C. Butcher) as shown in Table 13.1.

Table 13.1: Butcher tableau for a general RK method.

$$
\begin{array}{c|ccc}
c_1 & a_{11} & \ldots & a_{1s} \\
\vdots & \vdots & \vdots & \vdots \\
c_s & a_{s1} & \ldots & a_{ss} \\
\hline
 & b_1 & \ldots & b_s
\end{array}
$$

For an *explicit RK method*, the matrix of coefficients $A = (a_{ij})$ is lower triangular with zeros on the diagonal, i.e. $a_{ij} = 0$ for $i \leq j$. *The zeros of A are usually not displayed in the Butcher tableau.*

Example 13.14. *Tables 13.2-13.4 show the Butcher tableaux of some explicit RK methods.*

Table 13.2: Improved Euler.

$$
\begin{array}{c|cc}
0 & & \\
1 & 1 & \\
\hline
 & \frac{1}{2} & \frac{1}{2}
\end{array}
$$

Table 13.3: Midpoint RK.

$$
\begin{array}{c|cc}
0 & & \\
\frac{1}{2} & \frac{1}{2} & \\
\hline
 & 0 & 1
\end{array}
$$

Implicit RK methods are useful for some initial values problems with disparate time scales as we will see later. To reduce the computational work needed to solve for the unknown $K_1, \ldots, K_s$ (each K is vector-valued for a

Table 13.4: Classical fourth order RK.

$$
\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & & \\
1 & 0 & 0 & 1 & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}
$$

system of ODEs) in an implicit RK method, two particular types of implicit RK methods are usually employed. The first type is the *diagonally implicit* RK method or DIRK which has $a_{ij} = 0$ for $i < j$ and at least one a_{ii} is nonzero. The second type has also $a_{ij} = 0$ for $i < j$ but with the additional condition that $a_{ii} = \gamma$ for all $i = 1, \ldots, s$ and γ is a constant. The corresponding methods are called *singly diagonally implicit* RK or SDIRK.

Example 13.15. *Tables 13.5-13.8 show some examples of DIRK and SDIRK methods.*

Table 13.5: Backward Euler.

$$
\begin{array}{c|c}
1 & 1 \\
\hline
 & 1
\end{array}
$$

Table 13.6: Implicit mid-point rule RK.

$$
\begin{array}{c|c}
\frac{1}{2} & \frac{1}{2} \\
\hline
 & 1
\end{array}
$$

13.8 Implementation for Systems

As mentioned in the introduction, the IVP could be for a first order system of ODEs, i.e. for vector-valued y and f. The implementation of a numerical

Table 13.7: Hammer and Hollingworth DIRK.

$$
\begin{array}{c|cc}
0 & 0 & 0 \\
\frac{2}{3} & \frac{1}{3} & \frac{1}{3} \\
\hline
 & \frac{1}{4} & \frac{3}{4}
\end{array}
$$

Table 13.8: Two-stage, order 3 SDIRK ($\gamma = \frac{3\pm\sqrt{3}}{6}$).

$$
\begin{array}{c|cc}
\gamma & \gamma & 0 \\
1-\gamma & 1-2\gamma & \gamma \\
\hline
 & \frac{1}{2} & \frac{1}{2}
\end{array}
$$

method for an ODE system requires the appropriate updating of all the components of the approximate solution as the following example illustrates.

Consider the IVP

$$y_1' = f_1(t, y_1, y_2), \tag{13.126}$$

$$y_2' = f_2(t, y_1, y_2), \tag{13.127}$$

$$y_1(0) = \alpha_1, \ y_2(0) = \alpha_2 \tag{13.128}$$

and suppose we would like to get an approximation for the solution of this first order ODE system for $t \in [0, T]$ using the improved Euler method. We can write this in the form

$$K_{11} = f_1(t_n, y_1^n, y_2^n), \tag{13.129}$$

$$K_{12} = f_2(t_n, y_1^n, y_2^n), \tag{13.130}$$

$$K_{21} = f_1(t_n + \Delta t, y_1^n + \Delta t K_{11}, y_2^n + \Delta t K_{12}), \tag{13.131}$$

$$K_{22} = f_2(t_n + \Delta t, y_1^n + \Delta t K_{11}, y_2^n + \Delta t K_{12}), \tag{13.132}$$

$$y_1^{n+1} = y_1^n + \Delta t \left[\frac{1}{2} K_{11} + \frac{1}{2} K_{21} \right], \tag{13.133}$$

$$y_2^{n+1} = y_2^n + \Delta t \left[\frac{1}{2} K_{12} + \frac{1}{2} K_{22} \right]. \tag{13.134}$$

The implementation is now straightforward; it just requires one procedure to evaluate $f_1(t, y_1, y_2)$ and $f_2(t, y_1, y_2)$ and this is called twice per time step:

first with arguments (parameters) (t_n, y_1^n, y_2^n) for the first stage and then with arguments $(t_n + \Delta t, y_1^n + \Delta t K_{11}, y_2^n + \Delta t K_{12})$ for the second stage.

Note that for an explicit (m-step) multistep method we only need to evaluate f once per time step because we store the other $m - 1$ previous values which in turn get successively updated as the time stepping proceeds.

13.9 Adaptive Stepping

So far we have considered a fixed Δt throughout the entire computation of an approximation to the IVP of an ODE or of a system of ODEs. We can vary Δt as we march up in t to maintain the approximation within a given error bound. The idea is to obtain an estimate of the error using two different methods, one of order p and one of order $p + 1$, and employ this estimate to decide whether the size of Δt is appropriate or not at the given time step.

Let y^{n+1} and w^{n+1} be the numerical approximations updated from y^n using the method of order p, and $p + 1$, respectively. Then, we estimate the error at t_{n+1} by

$$e^{n+1}(\Delta t) \approx w^{n+1} - y^{n+1}. \tag{13.135}$$

If $|w^{n+1} - y^{n+1}| \leq \delta$, where δ is a prescribed tolerance, then we maintain the same Δt and use w^{n+1} as initial condition for the next time step. If $|w^{n+1} - y^{n+1}| > \delta$, we decrease Δt (e.g. we set it to $\Delta t/2$), recompute y^{n+1} and w^{n+1}, obtain the new estimate of the error (13.135), etc.

One-step methods allow for straightforward use of variable Δt. Variable step, multistep methods can also be derived but are not used much in practice due to more limited stability properties.

13.10 Embedded Methods

For computational efficiency, adaptive stepping as described above is implemented reusing as much as possible evaluations of f, the derivative of y, because this is the most expensive part of RK methods. So the idea is to embed, with minimal additional f evaluations, an RK method inside another. The following example illustrates this.

Consider the improved Euler method (second order) and the Euler method (first order). We can embed them as follows

$$K_1 = f(t_n, y^n), \tag{13.136}$$
$$K_2 = f\left(t_n + \Delta t, y^n + \Delta t K_1\right), \tag{13.137}$$
$$w^{n+1} = y^n + \Delta t \left[\frac{1}{2}K_1 + \frac{1}{2}K_2\right], \tag{13.138}$$
$$y^{n+1} = y^n + \Delta t K_1. \tag{13.139}$$

Note that the approximation of the derivative K_1 is used for both methods. The computation of the higher order method (13.138) only costs an additional evaluation of f.

13.11 Multistep Methods

Multistep methods use approximations from more than one step to obtain the approximation at the next time step. *Linear* multistep methods can be written in the general form

$$\sum_{j=0}^{m} a_j y^{n+j} = \Delta t \sum_{j=0}^{m} b_j f^{n+j}. \tag{13.140}$$

where $m \geq 2$ is the number of previous steps the method employs.

Multistep methods only require one evaluation of f per time step because the other previously computed values of f are stored. Thus, multistep methods have generally lower computational cost per time step than one-step methods of the same order. The trade-off is reduced numerical stability as we will see later.

We used in Section 13.2 interpolation and finite differences to construct some examples of multistep methods. It is possible to build also multistep methods by choosing the coefficients $a_0, \ldots, a_m$ and $b_0, \ldots, b_m$ so as to achieve a desired maximal order for a given $m \geq 2$ and/or to have certain stability properties.

Two classes of multistep methods, both derived from interpolation, are among the most commonly used multistep methods. These are the explicit and implicit Adams methods.

13.11.1 Adams Methods

We constructed in Section 13.2 the two-step Adams-Barshforth method

$$y^{n+1} = y^n + \frac{\Delta t}{2}\left[3f^n - f^{n-1}\right], \quad n = 1, 2, \ldots N-1,$$

where $f^n = f(t_n, y^n)$ and $f^{n-1} = f(t_{n-1}, y^{n-1})$. An m-step explicit Adams method, also called Adams-Bashforth, can be derived by starting with the integral formulation of the IVP,

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t))dt, \tag{13.141}$$

and replacing the integrand with the interpolating polynomial $p \in \mathbb{P}_{m-1}$ of (t_j, f^j) for $j = n-m+1, \ldots, n$. Recall that $f^j = f(t_j, y^j)$. If we represent p in Lagrange form we have

$$p(t) = \sum_{j=n-m+1}^{n} l_j(t) f^j, \tag{13.142}$$

where

$$l_j(t) = \prod_{\substack{k=n-m+1 \\ k \neq j}}^{n} \frac{(t - t_k)}{(t_j - t_k)}, \quad \text{for } j = n-m+1, \ldots, n. \tag{13.143}$$

Thus, the m-step explicit Adams method has the form

$$y^{n+1} = y^n + \Delta t\left[b_{m-1}f^n + b_{m-2}f^{n-1} + \ldots b_0 f^{n-m+1}\right], \tag{13.144}$$

where

$$b_{j-(n-m+1)} = \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} l_j(t)dt, \quad \text{for } j = n-m+1, \ldots, n. \tag{13.145}$$

Here are the first three explicit Adams methods, 2-step, 3-step, and 4-step, respectively:

$$y^{n+1} = y^n + \frac{\Delta t}{2}\left[3f^n - f^{n-1}\right], \tag{13.146}$$

$$y^{n+1} = y^n + \frac{\Delta t}{12}\left[23f^n - 16f^{n-1} + 5f^{n-2}\right], \tag{13.147}$$

$$y^{n+1} = y^n + \frac{\Delta t}{24}\left[55f^n - 59f^{n-1} + 37f^{n-2} - 9f^{n-3}\right]. \tag{13.148}$$

The implicit Adams methods, also called Adams-Moulton methods, are derived by including (t_{n+1}, f^{n+1}) in the interpolation. That is, $p \in \mathbb{P}_m$ is now the polynomial interpolating (t_j, f^j) for $j = n - m + 1, \ldots, n + 1$. Here are the first three implicit Adams methods:

$$y^{n+1} = y^n + \frac{\Delta t}{12}\left[5f^{n+1} + 8f^n - f^{n-1}\right], \tag{13.149}$$

$$y^{n+1} = y^n + \frac{\Delta t}{24}\left[9f^{n+1} + 19f^n - 5f^{n-1} + f^{n-2}\right], \tag{13.150}$$

$$\begin{aligned} y^{n+1} = y^n + \frac{\Delta t}{720}\big[&251f^{n+1} \\ &+ 646f^n - 264f^{n-1} + 106f^{n-2} - 19f^{n-3}\big]. \end{aligned} \tag{13.151}$$

13.11.2 D-Stability and Dahlquist Equivalence Theorem

Recall that we can write a general multistep method as

$$\sum_{j=0}^{m} a_j y^{n+j} = \Delta t \sum_{j=0}^{m} b_j f^{n+j}.$$

The coefficients a_j's and b_j's define two *characteristic polynomials* of the multistep method:

$$\rho(z) = a_m z^m + a_{m-1} z^{m-1} + \ldots + a_0, \tag{13.152}$$

$$\sigma(z) = b_m z^m + b_{m-1} z^{m-1} + \ldots + b_0. \tag{13.153}$$

Consistency, given by the conditions (13.90)-(13.91), can be equivalently expressed as

$$\rho(1) = 0, \tag{13.154}$$

$$\rho'(1) = \sigma(1). \tag{13.155}$$

Numerical stability is related to the notion of a uniform bound (independent of n) of the amplification of the local error in the limit as $n \to \infty$ and $\Delta t \to 0$ [see bound (13.101) for one-step methods]. Thus, it is natural to consider the equation:

$$a_m y^{n+m} + a_{m-1} y^{n+m-1} + \ldots + a_0 y^n = 0. \tag{13.156}$$

This is a homogeneous linear difference equation. Since $a_m \neq 0$, we can easily solve for y^{n+m} in terms of the previous values m values. So given the initial values $y^0, y^1, \ldots, y^{m-1}$, there is a unique solution of (13.156). Let us look for a solution of the form $y^n = c\xi^n$, where c is a constant and the n in ξ^n is a power not a superindex. Plugging this ansatz in (13.156) we obtain

$$c\xi^n \left[a_m\xi^m + a_{m-1}\xi^{m-1} + \ldots + a_0\right] = 0. \tag{13.157}$$

If $c\xi^n = 0$ we get the trivial solution $y^n \equiv 0$, for all n. Otherwise, ξ has to be a root of the polynomial ρ.

If ρ has m distinct roots, $\xi_1, \xi_2, \ldots, \xi_m$, the general solution of (13.156) is

$$y^n = c_1\xi_1^n + c_2\xi_2^n + \ldots + c_m\xi_m^n, \tag{13.158}$$

where $c_1, c_2, \ldots, c_m$ are determined uniquely from the m initial values y^0, y^1, $\ldots$, y^{m-1}.

If the roots are not all distinct, the solution of (13.156) changes as follows: If for example $\xi_1 = \xi_2$ is a double root, i.e. a root of multiplicity 2 $[\rho(\xi_1) = 0$, and $\rho'(\xi_1) = 0$ but $\rho''(\xi_1) \neq 0]$ then $y^n = n\xi_1^n$ is also a solution of (13.156). Let's check this is indeed the case. Substituting $y^n = n\xi_1^n$ in (13.156) we get

$$\begin{aligned} &a_m(n+m)\xi_1^{n+m} + a_{m-1}(n+m-1)\xi_1^{n+m-1} + \ldots + a_0 n\xi_1^n \\ &\quad = \xi_1^n\left[a_m(n+m)\xi_1^m + a_{m-1}(n+m-1)\xi_1^{m-1} + \ldots + a_0 n\right] \\ &\quad = \xi_1^n\left[n\rho(\xi_1) + \xi_1\rho'(\xi_1)\right] = 0. \end{aligned} \tag{13.159}$$

Thus, when there is one double root, the general solution of (13.156) is

$$y^n = c_1\xi_1^n + c_2 n\xi_1^n + c_3\xi_3^n + \ldots + c_m\xi_m^n. \tag{13.160}$$

If there is a triple root, say $\xi_1 = \xi_2 = \xi_3$, the general solution of (13.156) is given by

$$y^n = c_1\xi_1^n + c_2 n\xi_1^n + c_3 n(n-1)\xi_1^n + \ldots + c_m\xi_m^n, \tag{13.161}$$

and so on and so forth.

We need the solution y^n of (13.156) to remain bounded as $n \to \infty$ for otherwise it will not converge to $y(t) = \alpha$, the solution of $y' = 0, y(0) = \alpha$. Thus, we need that all the roots $\xi_1, \xi_2, \ldots, \xi_m$ of ρ satisfy:

(a) $|\xi_j| \leq 1$, for all $j = 1, 2, \ldots, m$.

(b) If ξ_k is a root of multiplicity greater than one then $|\xi_k| < 1$.

(a) and (b) are known as the root condition.

Since the exact solution $y(t)$ is bounded, the global error is bounded as $\Delta t \to 0$ ($n \to \infty$) if and only if the numerical approximation y^n is bounded as $\Delta t \to 0$ ($n \to \infty$). This motivates the following central concept in the theory of multistep methods.

Definition 13.7. *A multistep method is D-stable (or zero-stable) if the zeros of ρ satisfy the root condition.*

Example 13.16. *All the m-step ($m > 1$) methods in the Adams family have*

$$\rho(\xi) = \xi^m - \xi^{m-1}. \tag{13.162}$$

The roots of ρ are 1 (with multiplicity one) and 0. Hence, the Adams methods are all D-stable.

As we have seen, D-stability is necessary for convergence of a multistep method. It is truly remarkable that D-stability, together with consistency, is also sufficient for convergence. In preparation for this fundamental result, let us go back to the general linear multistep method. Without loss of generality we take $a_m = 1$ and write a general multistep method as

$$y^{n+m} + a_{m-1}y^{n+m-1} + \ldots + a_0 y^n = c_{n+m}, \tag{13.163}$$

where

$$c_{n+m} = \Delta t \sum_{j=0}^{m} b_j f^{n+j}. \tag{13.164}$$

For c_{n+m} given, (13.163) is a inhomogeneous linear difference equation. We will show next that we can express the solution of (13.163) in terms of m particular solutions of the homogeneous equation ($c_{n+m} = 0$), the m initial values, and the right hand side in a sort of *Discrete Duhamel's principle.* For a multistep method, c^{n+m} actually depends on the solution itself so we proceed formally to find the aforementioned solution representation.

Let y_k^n, for $k = 0, 1, \ldots, m-1$, be the solution of the homogeneous equation with initial values $y_k^n = \delta_{n,k}$, $n = 0, 1, \ldots, m-1$, and let w_k^n for $k = m, m+1, ...$ be the solution of the "unit impulse equation"

$$w_k^{n+m} + a_{m-1}w_k^{n+m-1} + \ldots + a_0 w_k^n = \delta_{n,k-m}, \tag{13.165}$$

with initial values $w_k^0 = w_k^1 = \ldots = w_k^{m-1} = 0$. Then, the solution of (13.163) with initial values $y^0 = \alpha_0, y^1 = \alpha_1, \ldots, y^{m-1} = \alpha_{m-1}$ can be written as

$$y^n = \sum_{k=0}^{m-1} \alpha_k y_k^n + \sum_{k=m}^{n} c_k w_k^n, \quad n = 0, 1, \ldots. \tag{13.166}$$

The first sum enforces the initial conditions and is a solution of the homogeneous equation (13.156). Since $w_k^n = 0$ for $n < k$ we can extend the second sum to ∞. Let

$$z^n = \sum_{k=m}^{n} c_k w_k^n = \sum_{k=m}^{\infty} c_k w_k^n. \tag{13.167}$$

Then

$$\begin{aligned}\sum_{j=0}^{m} a_j z^{n+j} = \sum_{j=0}^{m} a_j \sum_{k=m}^{\infty} c_k w_k^{n+j} &= \sum_{k=m}^{\infty} c_k \sum_{j=0}^{m} a_j w_k^{n+j} \\ &= \sum_{k=m}^{\infty} c_k\, \delta_{n,k-m} = c_{n+m},\end{aligned} \tag{13.168}$$

i.e. z^n is a solution of (13.163). Finally, we can interpret (13.165) as a homogeneous problem with m "delayed" (by $k - m + 1$ steps) initial values $0, 0, \ldots, 0, 1$. Hence, $w_k^n = y_{m-1}^{n+m-k-1}$ and we arrive at the following representation of the solution of (13.163)

$$y^n = \sum_{k=0}^{m-1} \alpha_k y_k^n + \sum_{k=m}^{n} c_k y_{m-1}^{n+m-1-k}, \quad n = 0, 1, \ldots \tag{13.169}$$

where $y_{m-1}^n = 0$ for $n < 0$.

Theorem 13.3. *(Dahlquist Equivalence Theorem) A multistep method is convergent if and only if it is consistent and D-stable.*

Proof. Necessity of D-stability. If a multistep method is convergent for all IVP's $y' = f(t, y)$, $y(0) = \alpha$ with f continuous and uniformly Lipschitz in y, then so is for $y' = 0$, $y(0) = 0$ whose solution is $y(t) = 0$. In this case y^n satisfies (13.156). One of the solutions of this equation is $y^n = \Delta t \xi^n$, where ξ is a root of the characteristic polynomial ρ (13.152). Note that $y^n = \Delta t \xi^n$

satisfies the convergence requirement that $y^k \to 0$ for $k = 0, 1, \ldots, m-1$ as $\Delta t \to 0$. If $|\xi| > 1$, for fixed $t = n\Delta t$ $(0 < t \le T)$, $|y^n| = \Delta t|\xi|^n = t|\xi|^n/n$ does not converge to zero as $n \to \infty$. Similarly, if ξ is a root of multiplicity greater than 1 and $|\xi| = 1$, $y^n = \Delta t n \xi^n$ is a solution and $|y^n| = \Delta t n|\xi|^n = t|\xi|^n = t$ does not converge to zero as $n \to \infty$.

Necessity of consistency. Consider the particular IVP $y' = 0$, $y(0) = 1$, whose solution is $y(t) = 1$. Again, y^n satisfies (13.156). Setting $y^0 = y^1 = \ldots = y^{m-1} = 1$ we can obtain y^n for $n \ge m$ from (13.156). If the method converges, then $y^n \to 1$ as $n \to \infty$. Using this in (13.156) implies that $a_m + a_{m-1} + \ldots + a_0 = 0$ or equivalently $\rho(1) = 0$. Now, consider the initial value, $y' = 1$, $y(0) = 0$. Then, the multistep method satisfies

$$a_m y^{n+m} + a_{m-1} y^{n+m-1} + \ldots + a_0 y^n = \Delta t(b_m + b_{m-1} + \ldots + b_0). \quad (13.170)$$

We are now going to find a solution of this equation of the form $y^n = An\Delta t$ for a suitable constant A. First, note that $y^k = Ak\Delta t$ converges to zero as $\Delta t \to 0$ for $k = 0, 1, \ldots, m-1$, as required. Substituting $y^n = An\Delta t$ into (13.170) we get

$$A\Delta t\left[a_m(n+m) + a_{m-1}(n+m-1) + \ldots + a_0 n\right] = \Delta t(b_m + b_{m-1} + \ldots + b_0)$$

and splitting the left hand side,

$$\begin{aligned} A\Delta t n\left[a_m + a_{m-1} + \ldots + a_0\right] + A\Delta t\left[m a_m + (m-1)a_{m-1} + \ldots + a_1\right] \\ = \Delta t(b_m + b_{m-1} + \ldots + b_0). \end{aligned} \quad (13.171)$$

Using $\rho(1) = 0$ this simplifies to

$$A\Delta t\left[m a_m + (m-1)a_{m-1} + \ldots + a_1\right] = \Delta t(b_m + b_{m-1} + \ldots + b_0), \quad (13.172)$$

i.e. $A\rho'(1) = \sigma(1)$. Since D-stability is necessary for convergence $\rho'(1) \ne 0$ and consequently $A = \sigma(1)/\rho'(1)$ and $y^n = \frac{\sigma(1)}{\rho'(1)} n\Delta t$ is a solution of (13.170). For fixed $t = n\Delta t$, y^n should converge to t as $n \to \infty$. Therefore, we must have $\sigma(1) = \rho'(1)$, which together with $\rho(1) = 0$, implies consistency.

Sufficiency of consistency and D-stability. From the definition of the local truncation error (13.64)

$$\sum_{j=0}^{m} a_j y(t_{n+j}) = \Delta t \sum_{j=0}^{m} b_j f(t_{n+j}, y(t_{n+j})) + \Delta t\, \tau^{n+m}(\Delta t). \quad (13.173)$$

Subtracting (13.140) to this equation we get

$$\sum_{j=0}^{m} a_j e^{n+j}(\Delta t) = c_{n+m}, \quad n = 0, 1, \ldots, N-m, \tag{13.174}$$

where $e^j(\Delta t) = y(t_j) - y^j$ is the global error at t_j and

$$c_{n+m} = \Delta t \sum_{j=0}^{m} b_j [f(t_{n+j}, y(t_{n+j})) - f^{n+j}] + \Delta t \; \tau^{n+m}(\Delta t). \tag{13.175}$$

Then, using (13.169) we can represent the solution of (13.174) as

$$e^n(\Delta t) = \sum_{k=0}^{m-1} e^k(\Delta t) y_k^n + \sum_{k=0}^{n-m} c_{k+m} y_{m-1}^{n-1-k}, \quad n = 0, 1, \ldots, N. \tag{13.176}$$

Since the method is D-stable, the solutions of the homogeneous linear difference equation, y_k^n, $k = 0, 1, \ldots m-1$, are bounded, i.e. there is M such that $|y_k^n| \leq M$, $k = 0, 1, \ldots m-1$ and all n. Then,

$$|e^n(\Delta t)| \leq mM \max_{0 \leq k \leq m-1} |e^k(\Delta t)| + M \sum_{k=0}^{n-m} |c_{k+m}|, \quad n = 0, 1, \ldots, N. \tag{13.177}$$

Moreover, using the Lipschitz continuity of f and the bound of the local truncation error

$$|c_{k+m}| \leq \Delta t \left[Lb \sum_{j=0}^{m} |e^{k+j}(\Delta t)| + C(\Delta t)^p \right], \tag{13.178}$$

where L is the Lipschitz constant and $b = \max_j |b_j|$. Therefore,

$$\begin{aligned} |e^n(\Delta t)| \leq{} & mM \max_{0 \leq k \leq m-1} |e^k(\Delta t)| \\ & + (n-m+1) M \Delta t \left[(m+1) Lb \max_{0 \leq k \leq n} |e^k(\Delta t)| + C(\Delta t)^p \right], \end{aligned} \tag{13.179}$$

for $n = 0, 1, \ldots, N$. Let $E^n = \max_{0 \leq k \leq n} |e^k(\Delta t)|$ (we omit the dependance of E^n on Δt to simplify the notation). Then, we can write (13.179) as

$$|e^n(\Delta t)| \leq mME^{m-1} + (n-m+1) M \Delta t \left[(m+1) LbE^n + C(\Delta t)^p \right]. \tag{13.180}$$

Since $E^n = |e_{k'}(\Delta t)|$ for some $0 \leq k' \leq n$, we can replace the left hand side of (13.180) by E^n and because $m > 1$ it follows that

$$E^n \leq \tilde{C} n \Delta t E^n + mME^{m-1} + MCn(\Delta t)^{p+1}, \tag{13.181}$$

where $\tilde{C} = (m+1)MLb$. Therefore,

$$\left(1 - \tilde{C} n \Delta t\right) E^n \leq mME^{m-1} + MCn(\Delta t)^{p+1}. \tag{13.182}$$

If we restrict the integration up to $T_1 = 1/(2\tilde{C})$, i.e. $\tilde{C} n \Delta t \leq 1/2$, we have

$$E^n \leq 2M\left[mE^{m-1} + T_1 C(\Delta t)^p\right], \quad 0 \leq n\Delta t \leq T_1 \tag{13.183}$$

and going back to the definition of E^n we obtain

$$|e^n(\Delta t)| \leq 2M\left[mE^{m-1} + T_1 C(\Delta t)^p\right], \quad 0 \leq n\Delta t \leq T_1. \tag{13.184}$$

The term E^{m-1} depends only on the m initialization values of the multistep method. For a consistent method $p \geq 1$ and $E^{m-1} \to 0$ as $\Delta t \to 0$. Hence (13.184) implies convergence on the interval $[0, T_1]$, where $T_1 = 1/(2\tilde{C})$. We can repeat the argument on the interval $[T_1, 2T_1]$, using the estimate of the error (13.184) for the first m values $e^{k_1-(m-1)}, e^{k_1-(m-2)}, \ldots, e^{k_1}$, where $k_1 = [T_1/\Delta t]$, and obtain convergence in $[T_1, 2T_1]$. Continuing with this process a finite number of times, $J = [T/T_1]$, we can prove convergence on the intervals

$$[0, T_1], [T_1, 2T_1], \ldots, [(J-1)T_1, T]. \tag{13.185}$$

The pointwise error bound on each of these interval depends on the error bound of the previous interval as follows

$$E_j \leq 2M\left[mE_{j-1} + T_1 C(\Delta t)^p\right], \quad j = 1, \ldots, J, \tag{13.186}$$

where E_j is the (pointwise) error bound on $[(j-1)T_1, jT_1]$ and $E_0 = E^{m-1}$. Defining $A = 2Mm$ and $B = 2MT_1 C(\Delta t)^p$, we get for the error bound of the last interval

$$\begin{aligned}
E_J &\leq AE_{J-1} + B \leq A\left[AE_{J-2} + B\right] + B \\
&= A^2 E_{J-2} + AB + B \\
&\leq A^2\left[AE_{J-3} + B\right] + AB + B \\
&= A^3 E_{J-3} + (A^2 + A + 1)B \\
&\;\vdots \\
&\leq A^J E_0 + (A^{J-1} + A^{J-2} + \ldots + 1)B.
\end{aligned} \tag{13.187}$$

Therefore, we obtain the error bound

$$|e^n(\Delta t)| \leq (2Mm)^J E^{m-1} + S(\Delta t)^p, \tag{13.188}$$

where $S = [(2Mm)^{J-1} + (2Mm)^{J-2} + \ldots + 1](2MT_1C)$, which establishes the convergence of a consistent, D-stable multistep method, and shows the dependence of the global error on the initialization error and on the truncation error. □

13.12 A-Stability

So far we have discussed numerical stability in the sense of a bounded growth of the error, or equivalently of the boundedness of the numerical approximation in the limit as $\Delta t \to 0$ ($n \to \infty$). There is another type of numerical stability which gives us some guidance on the actual size of Δt one can take for a stable computation using a given ODE numerical method. This type of stability is called linear stability, absolute stability, or A-stability. It is based on the behavior of the given numerical method for the simple linear problem:

$$y' = \lambda y, \tag{13.189}$$
$$y(0) = 1, \tag{13.190}$$

where λ is a complex number. The exact solution is $y(t) = e^{\lambda t}$.

Let us look at the forward Euler method applied to this model problem:

$$\begin{aligned} y^{n+1} &= y^n + \Delta t \lambda y^n = (1 + \Delta t\lambda) y^n \\ &= (1 + \Delta t\lambda)(1 + \Delta t\lambda) y^{n-1} = (1 + \Delta t\lambda)^2 y^{n-1} \\ &= \ldots = (1 + \Delta t\lambda)^{n+1} y^0 = (1 + \Delta t\lambda)^{n+1}. \end{aligned} \tag{13.191}$$

Thus, $y^n = (1 + \Delta t\lambda)^n$. Evidently, in order for this numerical approximation to remain bounded as $n \to \infty$ (long time behavior) we need

$$|1 + \Delta t\lambda| \leq 1. \tag{13.192}$$

This puts a constraint on the size of Δt we can take for a stable computation with the forward Euler method. For example, if $\lambda \in \mathbb{R}$ and $\lambda < 0$, we need to take $\Delta t \leq 2/|\lambda|$. Denoting $z = \Delta t\lambda$, the set

$$\mathcal{S} = \{z \in \mathbb{C} : |1 + z| \leq 1\}, \tag{13.193}$$

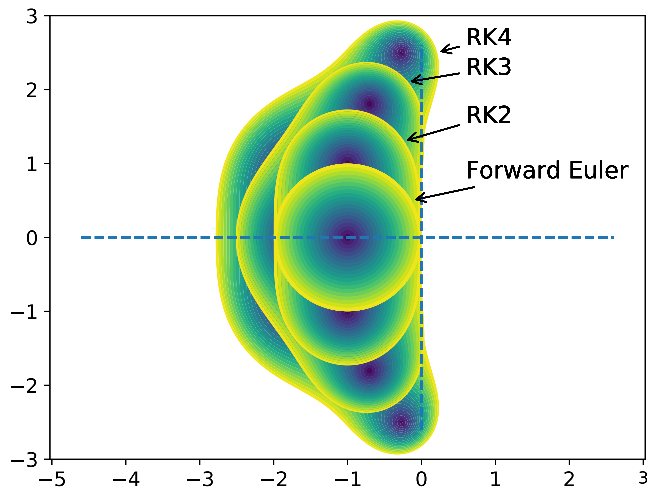

Figure 13.3: A-Stability regions for explicit RK methods of order 1–4.

i.e. the unit disk centered at -1 is the region of A-stability of the forward Euler method.

Runge-Kutta methods applied to the linear problem (13.189) produce a solution of the form

$$y^{n+1} = R(\Delta t\lambda)y^n, \tag{13.194}$$

where R is a rational function, i.e. $R(z) = \frac{P(z)}{Q(z)}$, where P ands Q are polynomials. In particular, when the RK method is explicit R is just a polynomial. For an RK method, the region of A-stability is given by the set

$$\mathcal{S} = \{z \in \mathbb{C} : |R(z)| \leq 1\}. \tag{13.195}$$

R is called the *stability function* of the RK method. Figure 13.3 shows the A-stability regions for explicit RK methods of order 1 (Euler) through 4. Note that as the order increases so does the A-stability region.

Example 13.17. *The improved Euler method. We have*

$$y^{n+1} = y^n + \frac{\Delta t}{2}\left[\lambda y^n + \lambda(y^n + \Delta t\lambda y^n)\right], \tag{13.196}$$

that is

$$y^{n+1} = \left[1 + \Delta t\lambda + \frac{1}{2}(\Delta t\lambda)^2\right] y^n. \tag{13.197}$$

The stability function is therefore $R(z) = 1 + z + \frac{z^2}{2}$. *Observe that*

$$R(z) = e^z + O(z^3). \tag{13.198}$$

That is, $R(\Delta t\lambda)$ *approximates* $e^{\Delta t\lambda}$ *to third order in* Δt, *as it should, because the method is second order. The A-stability region, i.e. the set of all complex numbers* z *such that* $|R(z)| \leq 1$ *is the RK2 region shown in Fig. 13.3).*

Example 13.18. *The backward Euler method. In this case we have,*

$$y^{n+1} = y^n + \Delta t\lambda y^{n+1}, \tag{13.199}$$

and solving for y^{n+1} *we obtain*

$$y^{n+1} = \left[\frac{1}{1 - \Delta t\lambda}\right] y^n. \tag{13.200}$$

So its stability function is $R(z) = 1/(1 - z)$ *and its A-stability region is therefore the set of complex numbers* z *such that* $|1 - z| \geq 1$, *i.e. the exterior of the unit disk centered at 1 as shown in Fig. 13.4(a).*

Example 13.19. *The implicit trapezoidal rule method. We have*

$$y^{n+1} = y^n + \frac{\Delta t}{2}\left(\lambda y^n + \lambda y^{n+1}\right) \tag{13.201}$$

and solving for y^{n+1} *we get*

$$y^{n+1} = \left[\frac{1 + \frac{\Delta t}{2}\lambda}{1 - \frac{\Delta t}{2}\lambda}\right] y^n. \tag{13.202}$$

Thus, the region of A-stability of the (implicit) trapezoidal rule method is the set complex numbers z *such that*

$$\left|\frac{1 + \frac{z}{2}}{1 - \frac{z}{2}}\right| \leq 1 \tag{13.203}$$

and this is the entire left half complex plane, $Re\{z\} \leq 0$ *[Fig. 13.4(b)]*

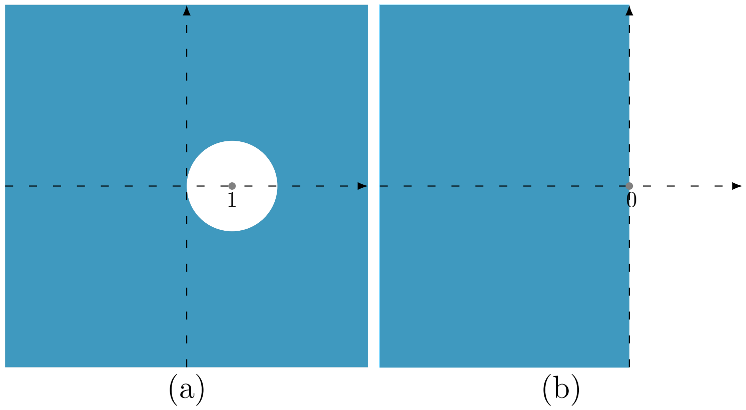

Figure 13.4: Region of A-stability for (a) backward Euler and (b) the trapezoidal rule method.

Definition 13.8. *A method is called A-stable if its linear stability region contains the left half complex plane.*

The trapezoidal rule method and the backward Euler method are both A-stable.

Let us consider now A-stability for linear multistep methods. When we apply an m-step ($m > 1$) method to the linear ODE (13.189) we get

$$\sum_{j=0}^{m} a_j y^{n+j} - \Delta t \lambda \sum_{j=0}^{m} b_j y^{n+j} = 0. \tag{13.204}$$

This is a constant coefficients, linear difference equation. We look for solutions of this equation in the form $y^n = c\,\xi^n$ as we have done earlier. Substituting into (13.204) we have

$$c\,\xi^n \sum_{j=0}^{m} (a_j - \Delta t \lambda b_j)\xi^j = 0. \tag{13.205}$$

If $c\,\xi^n = 0$ we get the trivial solution $y^n \equiv 0$, otherwise ξ must be root of the polynomial

$$\pi(\xi, z) = (a_m - z b_m)\xi^m + (a_{m-1} - z b_{m-1})\xi^{m-1} + \ldots + (a_0 - z b_0), \tag{13.206}$$

where $z = \Delta t \lambda$. We can write $\pi(\xi, z)$ in terms of the characteristic polynomials of the multistep method ρ and σ, (13.152) and (13.153), respectively, as

$$\pi(\xi, z) = \rho(\xi) - z\sigma(\xi). \tag{13.207}$$

Hence, for the numerical approximation y^n to remain bounded as $n \to \infty$ we need that all the roots of the polynomial π satisfy the root condition.

Definition 13.9. *The region of A-stability of a linear multistep method is the set*

$$\mathcal{S} = \{z \in \mathbb{C} : \textit{all the roots of } \pi(\xi, z) \textit{ satisfy the root condition}\}. \tag{13.208}$$

Recall that consistency for a multistep method translates into the following conditions $\rho(1) = 0$ and $\rho'(1) = \sigma(1)$. The first condition implies that $\pi(1, 0) = 0$. Because the zeros of a polynomial depend continuously on its coefficients, it follows that π has a root $\xi_1(z)$ for z in the neighborhood of zero. Such root is called the principal root of $\pi(\xi, z)$ and it can be shown that $\xi_1(z) = e^z + O(z^{p+1})$ for a method of order p. Thus, it carries the expected approximation to the exact solution e^z. The other roots of $\pi(\xi, z)$ are called parasitic roots.

Example 13.20. *Consider the 2-step method*

$$y^{n+1} + 4y^n - 5y^{n-1} = \Delta t \left[\, 4f^n + 2f^{n-1}\right]. \tag{13.209}$$

Then

$$\rho(\xi) = \xi^2 + 4\xi - 5 = (\xi - 1)(\xi + 5), \tag{13.210}$$

$$\sigma(\xi) = 4\xi + 2. \tag{13.211}$$

Thus, $\rho(1) = 0$ and $\rho'(1) = \sigma(1)$ and the method is consistent. However, the roots of ρ are 1 and -5 and hence the method is not D-stable. Therefore, by Dahlquist Equivalence Theorem, it is not convergent. Note that

$$\pi(\xi, z) = \xi^2 + 4(1 - z)\xi - (5 + 2z) \tag{13.212}$$

has roots

$$\xi_{\pm} = -2 + 2z \pm 3\sqrt{1 - \frac{2}{3}z + \frac{4}{9}z^2} \tag{13.213}$$

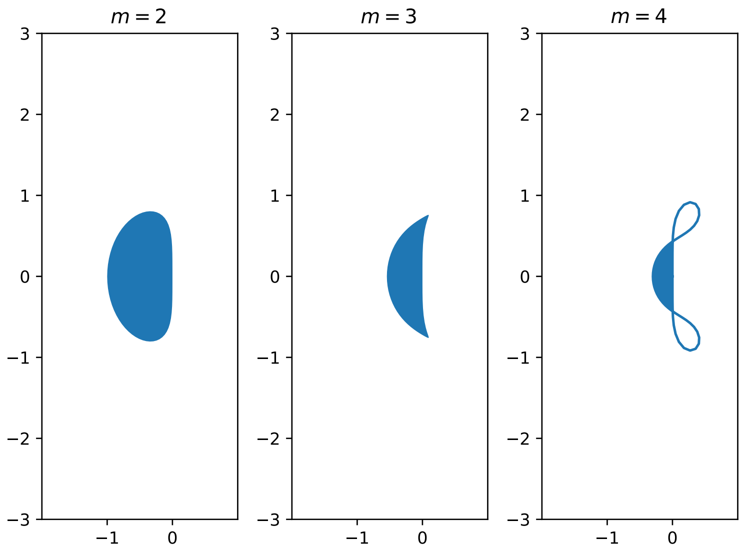

Figure 13.5: A-Stability regions (shown shaded) for the m-step Adams-Bashforth method for $m = 2, 3, 4$.

and for small $|z|$

$$\sqrt{1 - \frac{2}{3}z + \frac{4}{9}z^2} = 1 - \frac{1}{3}z + \frac{1}{6}z^2 + \frac{1}{18}z^3 + O(z^4) \tag{13.214}$$

we have

$$\xi_+ = 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3 + O(z^4) = e^z + O(z^4) \quad \textit{principal root.} \tag{13.215}$$

$$\xi_- = -5 + 3z + O(z^2) \quad \textit{parasitic root.} \tag{13.216}$$

Note that this 2-step, explicit method is third order. However, it is completely useless!

Figure 13.5 displays the A-stability region for the (explicit) m-step Adams-Bashforth (AB) method for $m = 2, 3, 4$. Note that the A-stability region of the AB methods decreases as m increases and that these stability regions are significantly smaller than those of the explicit RK counterparts (Fig. 13.5).

The (implicit) Adams-Moulton methods have a relatively larger stability region than the Adam-Bashforth methods as Fig. 13.6 shows. Note the

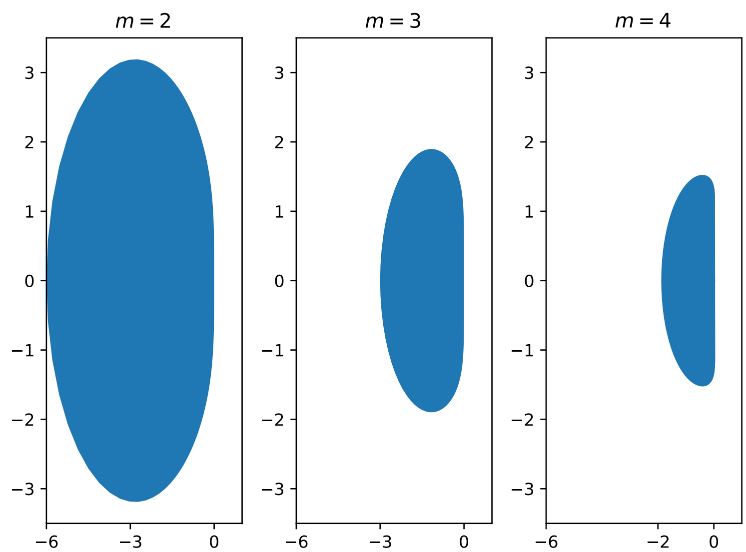

Figure 13.6: A-Stability regions (shown shaded) for the Adams-Moulton method of step $m = 2, 3, 4$.

change in the axis scale in this figure with respect to that used in Fig. 13.5. Implicit RK methods have a much larger stability region but are computationally more expensive than the multistep methods. Moreover, there is an A-stability barrier for the latter: it is not possible to have A-stable multistep methods of more than two steps ($m > 2$).

13.13 Numerically Stiff ODEs and L-Stability

In applications we often have systems of ODEs that have two or more disparate time scales. For example, a process that evolves very fast in a slowly varying environment or a reaction of several chemicals with vastly different reaction rates. This type of problems are called *numerically stiff* and, as we will see, explicit numerical methods fail miserably when applied to them. In fact, numerically stiff systems are often defined as those systems for which explicit numerical methods fail.

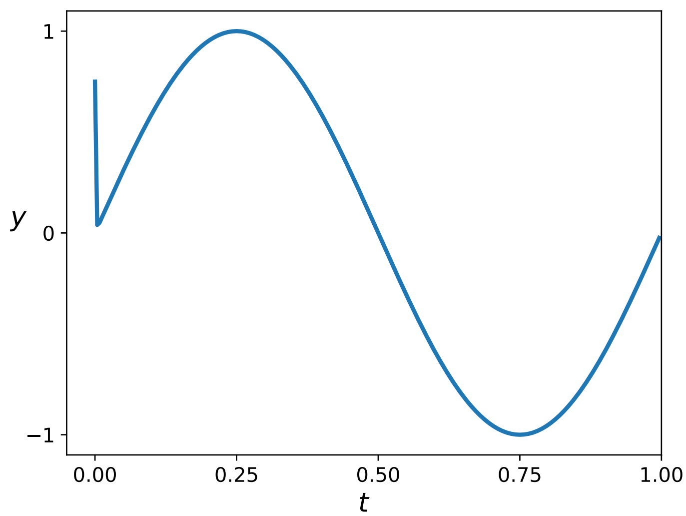

Figure 13.7: The exact solution (13.217) of the IVP (13.218)-(13.219) with $\alpha = 0.75$ and $\lambda = -1000$.

Consider the function

$$y(t) = \alpha e^{\lambda t} + \sin 2\pi t, \tag{13.217}$$

where $\alpha, \lambda \in \mathbb{R}$ and λ is negative with large absolute value. Thus, for $\alpha \neq 0$, y has two components: an exponentially decaying, transient part and a slowly (order one time scale) varying sinusoidal part. It is easy to verify that y in (13.217) is the solution of the IVP

$$y'(t) = \lambda\,[y(t) - \sin 2\pi t] + 2\pi \cos 2\pi t, \quad 0 < t \leq 1 \tag{13.218}$$

$$y(0) = \alpha. \tag{13.219}$$

For concreteness, let us take $\lambda = -1000$. Figure 13.7 shows $y(t)$ for $\alpha = 0.75$. Clearly, $y(t)$ quickly approaches the steady part, $\sin 2\pi t$. This will be the case for any other non-zero initial value α.

The explicit (forward) Euler method applied to (13.218)-(13.219) requires $\Delta t < 2/1000 = 1/500$ for A-stability. Figure 13.8 presents the approximation for $t \in [0, 0.25]$ obtained with the forward Euler method for $\Delta t = 1/512$, close to the boundary of $\mathcal{S}$. Observe that the Euler approximation approaches

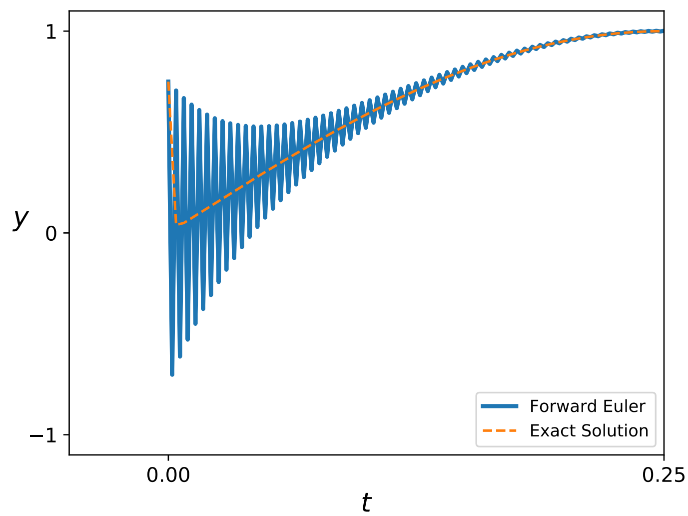

Figure 13.8: Forward Euler approximation and exact solution of (13.218)-(13.219) with $\alpha = 0.75$ and $\lambda = -1000$ for $t \in [0, 0.25]$. $\Delta t = 1/512$.

the steady solution but with an oscillatory behavior. The method fails to adequately capture the fast transient and the smooth evolution of the exact solution, despite the small Δt. The accuracy is clearly not $O(\Delta t)$!

We now consider the implicit (backward) Euler method to solve (13.218)-(13.219) again with $\lambda = -1000$ and $\Delta t = 1/512$. Figure 13.9 compares the backward Euler approximation with the exact solution and shows this method produces a smooth and accurate approximation. The backward Euler method is A-stable so for $\lambda < 0$ there is no stability restriction for Δt. This is advantageous when are interested in reaching quickly the steady state by taking a large Δt and do not care too much about the ultra fast transient.

The backward Euler method is only first order accurate. It is tempting to replace it with a second order A-stable method, like the trapezoidal rule method. After all, the latter has about the same computational cost as the former but its order is higher. Well, as Fig. 13.10 demonstrates, the trapezoidal rule method is actually a poor choice for this stiff problem; the first order, backward Euler method turns out to be more accurate than the second order trapezoidal rule method in the stiff regime $|\lambda \Delta t|$ large (we used $\Delta t = 0.05$).

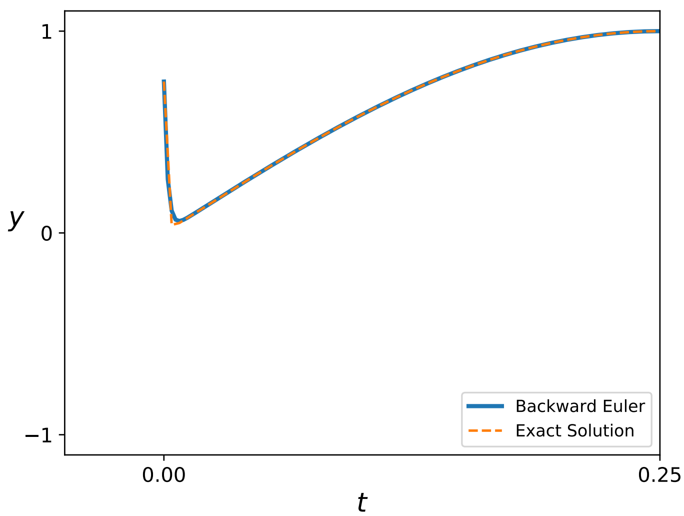

Figure 13.9: Backward Euler approximation and exact solution of (13.218)-(13.219) with $\alpha = 0.75$ and $\lambda = -1000$ for $t \in [0, 0.25]$. $\Delta t = 1/512$.

This behavior can be explained in terms of the stability function R. Recall that for the trapezoidal rule method

$$R(z) = \frac{1 + \frac{1}{2}z}{1 - \frac{1}{2}z}, \tag{13.220}$$

with $z = \lambda\Delta t$. In this example, $z = -50$ and $R(-50) = -24/26 \approx -0.923$, which is close to -1. Hence, the trapezoidal rule approximation decays very slowly toward the solution's steady state part and does so with oscillations because of the negative sign in $R(z)$. In contrast, for the backward Euler method $R(-50) = 1/51 \approx 0.0196$ and the decay is fast and nonoscillatory.

Note that if we take the initial condition $\alpha = 0$ the numerical stiffness of the initial value problem (13.218)-(13.219) disappears and the trapezoidal rule method approximates more accurately the exact solution $y(t) = \sin 2\pi$ than the backward Euler method for the same Δt, as expected.

As we have just seen, the behavior $|R(z)| \to 0$ as $|z| \to \infty$ is desirable for some stiff problems where the solution or some components of the solution have fast decay. This motivates the following definition.

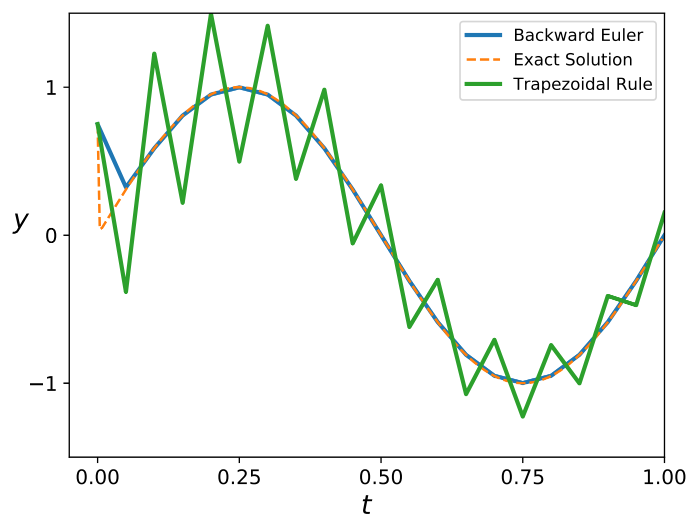

Figure 13.10: Trapezoidal rule approximation compared with the backward Euler approximation and the exact solution of (13.218)-(13.219) with $\alpha = 0.75$ and $\lambda = -1000$ for $t \in [0, 1]$. $\Delta t = 0.05$.

Definition 13.10. *A one-step method is L-stable if it is A-stable and*

$$\lim_{|z|\to\infty} |R(z)| = 0. \tag{13.221}$$

Example 13.21. *The backward Euler method is A-stable and has stability function $R(z) = 1/(1-z)$. Therefore, it is L-stable. The trapezoidal rule method, while A-stable is not L-stable for $|R(z)| \to 1$ as $|z| \to \infty$.*

Let us consider now the linear system

$$y' = Ay + f(t), \tag{13.222}$$

where $y, f \in \mathbb{R}^d$ and A is an $d \times d$ matrix. If A has distinct eigenvalues $\lambda_1, \lambda_2, \ldots \lambda_d$, with corresponding eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_d$, and all the eigenvalues have a negative real part, the general solution consists of a transient part and a steady state part, just as in the scalar example that motivated this section. Specifically, the general solution of $y' = Ay + f(t)$ can be written as

$$y(t) = \sum_{k=1}^{d} a_k e^{\lambda_k t}\mathbf{v}_k + s(t), \tag{13.223}$$

where the a_k's are constants determined by the initial condition and $s(t)$ represents the steady state. Let λ_p and λ_q be the eigenvalues with the largest and the smallest absolute value of the real part, i.e. $|\mathrm{Re}\{\lambda_p\}| = \max_j |\mathrm{Re}\{\lambda_j\}|$ and $|\mathrm{Re}\{\lambda_q\}| = \min_j |\mathrm{Re}\{\lambda_j\}|$. For an explicit method, $|\mathrm{Re}\{\lambda_p\}|$ limits the size of Δt to be in the A-stability region while $|\mathrm{Re}\{\lambda_q\}|$ dictates how long we need to time-step to reach the steady state; the smaller $|\mathrm{Re}\{\lambda_q\}|$ the longer we need to compute. Hence, the ratio of the fastest to slowest time scale

$$S_r = \frac{|\mathrm{Re}\{\lambda_p\}|}{|\mathrm{Re}\{\lambda_q\}|} \tag{13.224}$$

is a measure of the numerical stiffness for this linear system.

Often, numerically stiff ODE systems are nonlinear. We can get some estimate of their degree of stiffness by linearization. The idea is that small perturbations in the solution are governed by a linear system and if this is numerically stiff, then so is the nonlinear system.

Consider the autonomous nonlinear system $y' = f(y)$ and write

$$y(t) = y(t^*) + \epsilon w(t), \tag{13.225}$$

for small ϵ. Here, $y(t^*)$ is just a given state, for example $y(t_n)$. Now, Taylor expand $f(y)$ around $y(t^*)$, retaining only up the $O(\epsilon)$ term,

$$f(y(t)) \approx f(y(t^*)) + \epsilon \frac{\partial f}{\partial y}(y(t^*))w(t). \tag{13.226}$$

substituting (13.225) and (13.226) into $y' = f(y)$, we find the perturbation $w(t)$ approximately satisfies the linear ODE system

$$w'(t) = \frac{\partial f}{\partial y}(y(t^*))w(t) + \frac{1}{\epsilon} f(y(t^*)). \tag{13.227}$$

Then, at least *locally* (i.e. in a neighborhood of t^*) the variation of the solution is approximately governed by (13.227). Thus, one approximate indicator of numerical stiffness could be the stiffness ratio S_r of the Jacobian matrix $\frac{\partial f}{\partial y}(y(t^*))$. However, if the Jacobian varies significantly in the time interval of interest, S_r might not a good stiffness indicator. In practice, numerical stiffness is often assessed by using two error estimators. One for an explicit method and the other for a lower order approximation but that outperforms the explicit method in the stiff limit. If the error estimate for the lower order method is smaller than that of the explicit method repeatedly over several time-steps, it is viewed as an indication that the explicit method is inadequate, the IVP is considered stiff, and the explicit method is replaced by a suitable implicit one.

Example 13.22. *The van der Pol system*

$$y_1' = y_2 - y_1^3 + 2\mu y_1, \tag{13.228}$$
$$y_2' = -y_1, \tag{13.229}$$

is a simple model of an RLC electric circuit; y_1 and y_2 are related to the current and voltage, respectively, and the parameter μ controls the resistance. This ODE system has only one equilibrium point, $(0,0)$. *Let's look at the Jacobian evaluated at* $(0,0)$

$$\frac{\partial f}{\partial y}(0,0) = \begin{bmatrix} 2\mu & 1 \\ -1 & 0 \end{bmatrix}. \tag{13.230}$$

The eigenvalues of this matrix are

$$\lambda = \mu \pm \sqrt{\mu^2 - 1}. \tag{13.231}$$

For moderate values of $|\mu|$ *the system could be integrated with an explicit method. However, for very negative values of* μ *it becomes numerically stiff. For example, if* $\mu = -100$ *the corresponding stiffness ratio is*

$$S_r = \left| \frac{\mu - \sqrt{\mu^2 - 1}}{\mu + \sqrt{\mu^2 - 1}} \right| \approx 4 \times 10^4. \tag{13.232}$$

13.14 Bibliographic Notes

Numerical methods for ODEs is a much broader topic than presented here. The main references are the two volumes "Solving Differential Equations" by Hairer, Nørsett, and Wanner [HNW93] and Hairer and Wanner [HNW96], the classical texts by Henrici [Hen62] and Lambert [Lam91], and the book by Butcher [But08]. Other excellent, specialized books on numerical methods for both ordinary and partial differential equations are the ones by Leveque [LeV07] and by Iserles [Ise08].

Section 13.1 . The theory of the ODE IVP can be found in most differential equations books. For example Coddington and Levison [CL84](Chapters 1 and 2), and Sideris [Sid13](Chapter 3). Also, Volume I of "Solving Differential Equations" [HNW93] has three chapters (I.7, I.8, I.9) on existence theory with historical notes.

Section 13.2 . Euler proposed his method in 1768 [Eul68] (Caput VII, p. 494). RK methods were first introduced by Runge in 1895 [Run95] with subsequent contributions by Heun [Heu00] and Kutta [Kut01] as indicated by Butcher [But08](p. 93). The idea of multistep methods was first proposed by Bashforth and Adams [BA83], see [But08](p. 105) for a short historical account.

Section 13.3 . The division of numerical methods for the IVP into Runge-Kutta and multistep methods is a standard one in most texts. Here we follow instead Henrici's [Hen62] use of one-step and multistep. Stoer and Bulirsch [SB02][7.2.7] employ the general form of a nonlinear multistep method for the discussion of the main theory of multistep methods. Lambert and Shaw [LS66] provide early examples of a class of nonlinear multistep methods.

Section 13.4. For the definition of the local truncation error (LTE) we followed Hairer, Nørsett, and Wanner [HNW93], except that we chose to divide by Δt to make it consistent with the standard definition of LTE for finite differences for partial differential equations as we will see in the next chapter. Leveque [LeV07] makes a similar choice. The discussion of the LTE for implicit linear multistep methods follows that in [HNW93][III.2].

Section 13.5. The local truncation error and the definition of order of accuracy for a multistep method can be equivalently given in terms of the linear operator $L(y, t, \Delta t) = \sum_{j=0}^{m} a_j y(t_{n+j}) - \sum_{j=0}^{m} b_j y'(t_{n+j})$, see for example [Gau11, HNW93].

Section 13.6. The proof of Theorem 13.2 follows that in [Sch89] with some minor variations.

Section 13.7. We only presented a brief introduction of RK methods. The book by Butcher [But08] is an excellent reference on this topic, including the use of trees for order conditions, implicit RK methods, stability, and implementation details. Chapter II of [HNW93] is also a comprehensive reference for RK methods. The DIRK and SDIRK examples are from this text.

Section 13.8. The method of lines often employed in IVP for partial differential equations (see Section 14.2), which consists in discretizing in the space variables but keeping time continuous, leads to a large, first order system of ODE to which the methods seen in this chapter may be applied.

Section 13.9 and *Section 13.10.* Here we follow the exposition of Leveque's text [LeV07][5.7.1].

Section 13.11. We only discussed Adams methods but there are several other classes of multistep methods. For example, the Nyström methods are derived, like the Adams methods, using interpolation but starting from the integral equation from t_{n-1} to t_{n+1}. There are also the briefly mentioned BDF methods, extrapolation methods, second derivative methods for stiff ODEs, among others (see Chapter III of [HNW93] and Chapter V of [HNW93]). We chose D-stable to refer to methods that satisfy the root condition, in honor of Dahlquist [HNW93] but zero-stable is more commonly used. The proof of Dahlquist theorem [Dah56] follows that in [IK94] with minor variations. Another proof can be found in Henrici's classical text [Hen62].

Section 13.12. This is a standard topic in numerical ODE texts, where it is commonly found as absolute stability. Here we have followed [HNW93][IV.3, also for L-stability] and [V.1]. Example 13.20 is from Dahlquist [Dah56], see also [HNW93][III.3].

Section 13.13. The standard reference for stiff ODEs is the book by Hairer and Wanner [HNW96]. Our presentation follows that in Leveque's book [LeV07]. For the van der Pol equation and for applications in circuit theory see for example Chapter 12 of [HSD04].

Chapter 14

Numerical Methods for PDE's

This chapter provides a brief introduction to the vast topic of numerical methods for partial differential equations (PDEs). We focus the discussion on *finite difference methods*. Other important classes of numerical methods for PDEs, not treated here, are the finite element method and spectral methods.

We introduce the main concepts (truncation error, consistency, stability, and convergence) through one example, the heat equation in one spatial dimension. We then look at the method of lines, which connects the solution of initial value problems (IVPs) for PDEs with that of IVPs for large systems of ODEs. This is followed by two examples of implicit methods, the extension to higher dimensional problems, and wave propagation.

14.1 Key Concepts through One Example

Consider a thin rod of length L, with an initial temperature distribution f and whose left and right endpoints are keep at fixed temperatures u_L and u_R, respectively. Assuming the rod is homogeneous, the temperature $u(t,x)$ at a later time t and at a point x in the rod satisfies the heat equation problem:

$$u_t(t,x) = D\,u_{xx}(t,x), \quad 0 < x < L, \quad 0 < t \leq T, \tag{14.1}$$

$$u(0,x) = f(x), \quad 0 < x < L, \tag{14.2}$$

$$u(t,0) = u_L, \quad u(t,L) = u_R, \tag{14.3}$$

where $D > 0$ is the rod's diffusion coefficient and T defines the endpoint of the time interval of interest. This is an IVP with Dirichlet boundary conditions because we are specifying the value of the solution at the boundary, Eq. (14.3). For simplicity, we are going to take $u_R = u_L = 0$ and $L = \pi$. This linear problem can be solved analytically, using the method of separation of variables and Fourier (sine) series. Having a representation of the exact solution will be very helpful in the discussion of the fundamental aspects of the numerical approximations.

Assuming,

$$u(t, x) = \phi(t)\psi(x) \tag{14.4}$$

and substituting into the heat equation (14.1) we get $\phi'\psi = D\phi\psi''$, and rearranging

$$\frac{\phi'}{D\phi} = \frac{\psi''}{\psi}. \tag{14.5}$$

The expression on the left hand side of (14.5) is a function of t only while that on the right hand side is a function of x only. Therefore, they must both be equal to a constant. This constant has to be negative since $D > 0$ and the temperature cannot grow exponentially in time. We write this constant as $-\lambda^2$ and get from (14.5) the following two linear ODEs

$$\psi'' + \lambda^2\psi = 0, \tag{14.6}$$

$$\phi' + \lambda^2 D\phi = 0. \tag{14.7}$$

The first equation, (14.6), is that of a harmonic oscillator whose general solution is

$$\psi(x) = a\cos\lambda x + b\sin\lambda x. \tag{14.8}$$

The boundary condition at $x = 0$ implies $a = 0$, while the boundary condition at $x = \pi$ gives that λ has to be an integer, which we can assume to be positive since $b\sin(-\lambda x) = -b\sin(\lambda x)$ and we can absorb the negative sign in b. So we set $\lambda = k$ for all $k \in \mathbb{Z}^+$. On the other hand, the solutions of (14.7) are a constant times $\exp(-k^2 Dt)$. Thus, for every $k \in \mathbb{Z}^+$ and constant b_k, $b_k \exp(-k^2 Dt)\sin kx$ is a solution of the heat equation which vanishes at the boundary. We find the general solution by superposition:

$$u(t, x) = \sum_{k=1}^{\infty} b_k e^{-k^2 Dt}\sin kx. \tag{14.9}$$

The coefficients b_k are determined from the initial condition (14.2):

$$f(x) = \sum_{k=1}^{\infty} b_k \sin kx. \tag{14.10}$$

In other words, the b_k's are the sine Fourier coefficients of the initial temperature f, i.e.

$$b_k = \frac{2}{\pi} \int_0^{\pi} f(x) \sin kx dx, \quad k = 1, 2, \ldots \tag{14.11}$$

In general, we cannot evaluate these coefficients exactly but we can obtain an accurate approximation of them by using the composite trapezoidal rule and computing the corresponding discrete sine coefficients efficiently with the DST (or the FFT by extending f as an odd function, Section 3.12). Naturally, in any practical use of the solution's representation formula (14.9) we would also have to decide on where to truncate the series. Note that the solution is a superposition of harmonic modes whose amplitudes decay exponentially for $t > 0$, that is

$$u(t, x) = \sum_{k=1}^{\infty} \hat{u}_k(t) \sin kx, \tag{14.12}$$

where $\hat{u}_k(t) = b_k e^{-k^2 D t}$ is the amplitude of each harmonic mode (each $\sin kx$). Thus, even for a merely continuous initial condition, an accurate approximation can be obtained by truncating the series (14.12) after just a moderate number of terms.

Suppose the initial temperature is the function

$$f(x) = \begin{cases} x & 0 \le x \le \frac{\pi}{3}, \\ \frac{\pi}{3} & \frac{\pi}{3} < x \le \frac{2\pi}{3} \\ \pi - x & \frac{2\pi}{3} < x \le \pi, \end{cases} \tag{14.13}$$

shown in Fig. 14.1. For this piece-wise linear initial condition, we can evaluate the Fourier coefficients (14.11) exactly using integration by parts. Figure 14.2 shows snapshots of the solution (the series was truncated after 100 terms) for this initial condition. Note that even though the initial temperature f is just continuous, the solution at any $t > 0$ is smooth and decays monotonically in time.

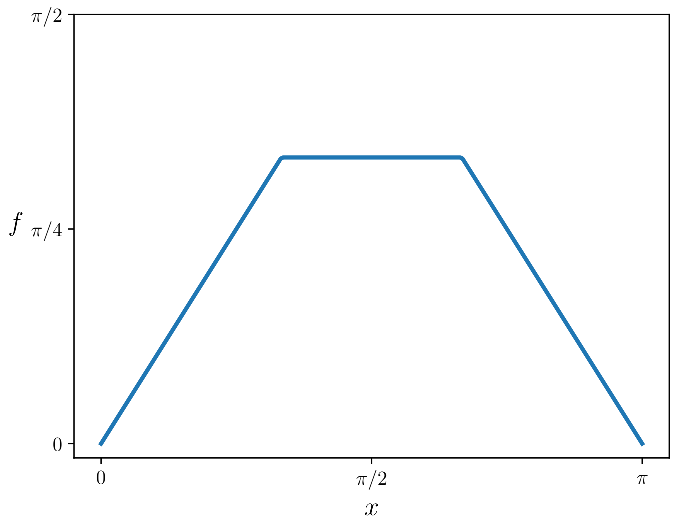

Figure 14.1: Initial temperature (14.13), $u(0, x) = f(x)$.

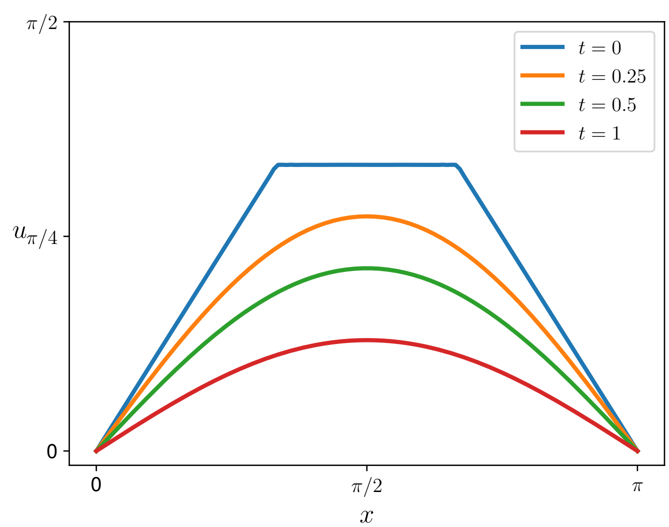

Figure 14.2: Exact solution of the heat equation with $D = 1$ for initial condition (14.13) and with homogenous Dirichlet boundary conditions.

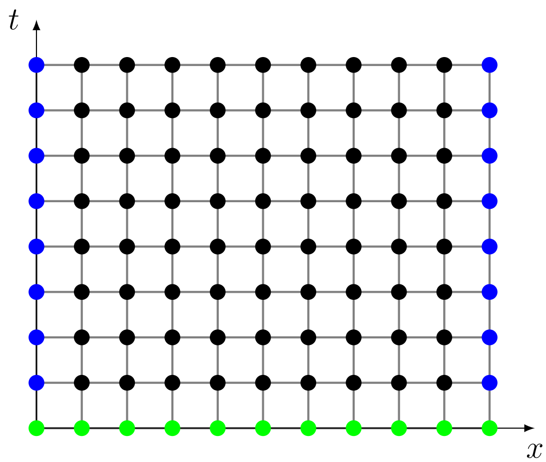

Figure 14.3: Grid in the xt-plane. The interior nodes (where an approximation to the solution is sought), the boundary points, and initial value nodes are marked with black, blue, and green dots, respectively.

While the preceding method based on a Fourier expansion yields an exact representation of the solution, ultimately approximations have to be made to obtain the Fourier coefficients of the initial condition and to truncate the series, as mentioned earlier. The method is also quite limited in its applicability. Finite difference methods offer a much broader applicability and are widely used in both linear and nonlinear PDE problems.

In finite difference methods, we start by laying out a grid on the computational space. In our example, the computational space is the rectangle $[0, \pi] \times [0, T]$ in the xt plane. For simplicity, we employ a uniform grid, i.e one created by a uniform partition of $[0, \pi]$ and $[0, T]$ as shown in Fig. 14.3. We select positive integers M and N so that our grid or mesh is defined by the nodes

$$(t_n, x_j) = (n\Delta t, j\Delta x), \quad n = 0, 1, \ldots, N, \quad j = 0, 1, \ldots M, \tag{14.14}$$

where $\Delta t = T/N$ is the temporal mesh size or time step size and $\Delta x = \pi/M$ is the spatial mesh size. We look for an approximation

$$u_j^n \approx u(t_n, x_j) \tag{14.15}$$

of the solution at the interior nodes (t_n, x_j), $n = 1, 2, \ldots, N$, $j = 1, 2, \ldots, M-1$. To this end, we approximate the derivatives with finite differences (Chapter 6). For example, if we use forward in time and centered in space finite differences we get

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = D\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2}, \tag{14.16}$$

for $n = 0, 2, \ldots, N-1$ and $j = 1, 2, \ldots, M-1$, with boundary conditions $u_0^n = 0$, $u_M^n = 0$, and initial condition $u_j^0 = f(x_j)$, $j = 0, 1, \ldots, M$. We can solve explicitly for u_j^{n+1} and march up in time, starting from the initial condition; for $n = 0, 1, \ldots, N-1$,

$$u_j^{n+1} = u_j^n + \alpha \left[u_{j+1}^n - 2u_j^n + u_{j-1}^n\right], \quad \text{for } j = 1, 2, \ldots, M-1, \tag{14.17}$$

where

$$\alpha = D\Delta t/(\Delta x)^2. \tag{14.18}$$

Note that the boundary conditions are used for $j = 1$ and $j = M - 1$. This is an explicit, one-step finite difference scheme and is straightforward to implement. The resulting approximation, however, depends crucially on whether $\alpha \leq 1/2$ or $\alpha > 1/2$. As Fig. 14.4(a)-(c) shows, for $\alpha = 0.55$ the numerical approximation does not vary smoothly; it has oscillations whose amplitude grows with n and has no resemblance with the exact solution. Clearly, the approximation for $\alpha = 0.55$ is numerically *unstable* in the sense that u_j^n is not bounded as $n \to \infty$. In contrast, for $\alpha = 0.50$ [Fig. 14.4(d)] the numerical approximation has the expected smooth and monotone behavior and approximates well the exact solution.

The following simple estimate offers some clue on why there is a marked difference in the numerical approximation depending on whether $\alpha \leq 1/2$.

From (14.17) we can rewrite the finite difference scheme as

$$u_j^{n+1} = \alpha u_{j+1}^n + (1 - 2\alpha)u_j^n + \alpha u_{j-1}^n \quad \text{for } j = 1, 2, \ldots, M-1. \tag{14.19}$$

Note that (since $D > 0$) $\alpha > 0$ and if $\alpha \leq 1/2$ then $1 - 2\alpha \geq 0$. Taking the absolute value in (14.19) and using the triangle inequality we get

$$|u_j^{n+1}| \leq \alpha|u_{j+1}^n| + (1 - 2\alpha)|u_j^n| + \alpha|u_{j-1}^n|. \tag{14.20}$$

Denoting

$$\|u^n\|_\infty = \max_{1\leq j\leq M-1} |u_j^n|, \tag{14.21}$$

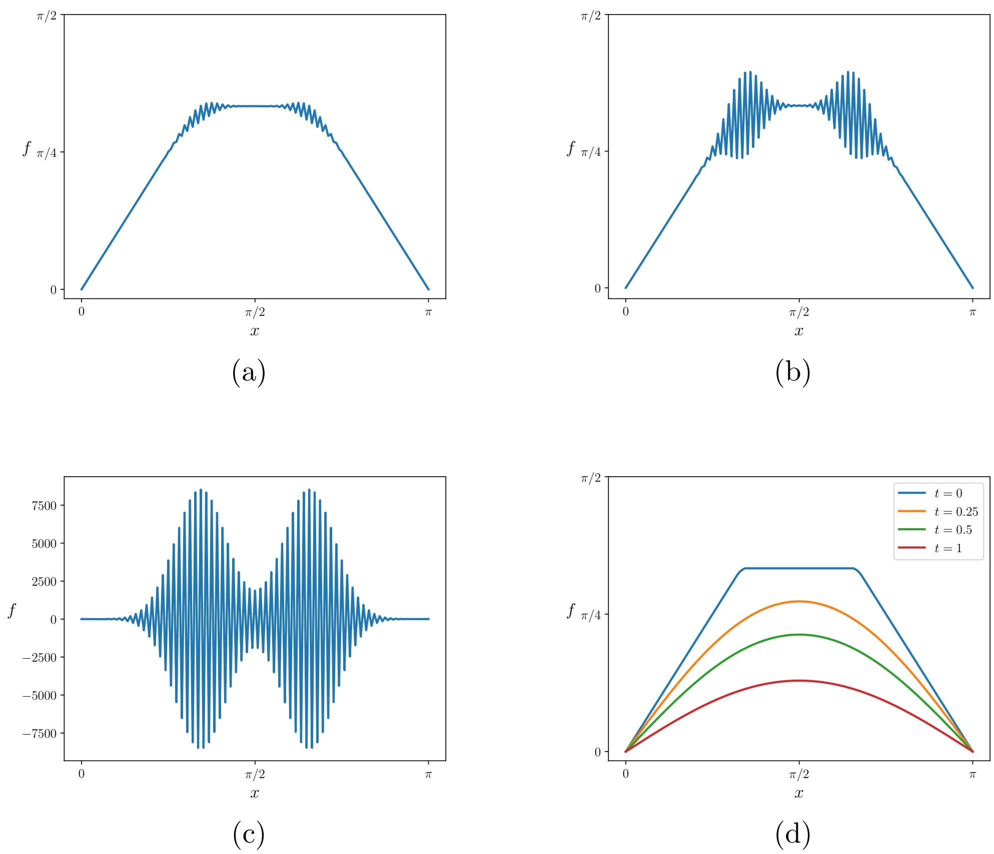

Figure 14.4: Numerical approximation of the heat equation with the forward in time-centered in space finite difference scheme for $\alpha = 0.55$ after (a) 30 time steps, (b) 40 time steps, and (c) 100 time steps and for $\alpha = 0.5$ (d) plotted at different times. In all the computations $\Delta x = \pi/128$.

and taking the maximum over j in (14.20) we obtain

$$\begin{aligned}\|u^{n+1}\|_\infty &\le \alpha\|u^n\|_\infty + (1-2\alpha)\|u^n\|_\infty + \alpha\|u^n\|_\infty \\ &= [\alpha + (1-2\alpha)+\alpha]\,\|u^n\|_\infty = \|u^n\|_\infty,\end{aligned} \tag{14.22}$$

and consequently,

$$\|u^{n+1}\|_\infty \le \|u^n\|_\infty \tag{14.23}$$

for all integers $n \ge 0$. Numerical schemes with this property are called *monotone*; the size of numerical approximation (in some norm) does not increase from one time step to the next. Using (14.23) repeatedly all the way down to $n = 0$ we have

$$\|u^n\|_\infty \le \|u^{n-1}\|_\infty \le \|u^{n-2}\|_\infty \le \ldots \le \|u^0\|_\infty \tag{14.24}$$

and thus

$$\|u^n\|_\infty \le \|u^0\|_\infty \tag{14.25}$$

for all integers $n \ge 0$. Since the initial condition $\|u^0\|_\infty = \|f\|_\infty \le$ constant we have that the numerical approximation remains bounded as $n \to \infty$. Thus, numerical method for $\alpha \le 1/2$ is *stable*.

14.1.1 von Neumann Analysis of Numerical Stability

The heat equation is a linear PDE with constant coefficients. This allowed us to use a Fourier (sine) series to arrive at the exact solution (14.9). Specifically, we separated the problem into an x and a t dependance. For the former, we found that for each $k \in \mathbb{Z}^+$, $\sin kx$ is the solution of

$$\frac{d^2}{dx^2}\psi = -k^2\psi$$

that vanishes at 0 and π, i.e., $\sin kx$ is an eigenfunction of the second derivative operator on the space of $C^2[0,\pi]$ functions vanishing at the boundary. We can do a similar separation of variables to represent the solution of the finite difference scheme

$$u_j^{n+1} = u_j^n + \alpha\left[u_{j+1}^n - 2u_j^n + u_{j-1}^n\right], \tag{14.26}$$

with the boundary conditions $u_0^n = u_M^n = 0$ for all n. To this effect, we note that the vector whose components are $\sin(kj\Delta x)$, $j = 1, \ldots, M-1$, with $\Delta x = \pi/M$, is an eigenvector of the centered, finite difference operator

$$\delta^2 u_j := u_{j-1} - 2u_j + u_{j+1} \tag{14.27}$$

with vanishing boundary conditions (at $j = 0, M$), i.e. of the $(M-1) \times (M-1)$ matrix

$$\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & & & 1 \\ & & & 1 & -2 \end{bmatrix}. \tag{14.28}$$

To prove this and to simplify the algebra we employ the complex exponential

$$e^{ikj\Delta x} = \cos(kj\Delta x) + i\sin(kj\Delta x), \quad j = 1, 2, \ldots, M-1. \tag{14.29}$$

We have

$$\begin{aligned} \delta^2 e^{ikj\Delta x} &= \left[e^{-ik\Delta x} - 2 + e^{ik\Delta x}\right] e^{ikj\Delta x} \\ &= -2\left[1 - \cos(k\Delta x)\right] e^{ikj\Delta x}. \end{aligned} \tag{14.30}$$

Taking the imaginary part, we obtain

$$\delta^2 \sin(kj\Delta x) = -2\left[1 - \cos(k\Delta x)\right] \sin(kj\Delta x). \tag{14.31}$$

This result and (14.12) suggest to look for solutions of the finite difference scheme (14.27) of the form

$$u_j^n = \hat{u}_k^n \sin(kj\Delta x), \quad \text{for } k \in \mathbb{Z}^+. \tag{14.32}$$

Substituting this in (14.27), and cancelling the common factor $\sin(kj\Delta x)$, we obtain

$$\hat{u}_k^{n+1} - \left[1 - 2\alpha(1 - \cos(k\Delta x))\right] \hat{u}_k^n = 0. \tag{14.33}$$

For each $k \in \mathbb{Z}^+$, this is a constant coefficient, linear difference equation (in the super-index) whose solutions are of the form

$$\hat{u}_k^n = b_k \xi^n, \tag{14.34}$$

where b_k is a constant determined by the initial condition $b_k = \hat{u}_k^0$, n in ξ^n is a power, and

$$\xi = 1 - 2\alpha\left[1 - \cos(k\Delta x)\right]. \tag{14.35}$$

The function ξ is called the amplification factor of the finite difference scheme because it determines how the amplitude of a Fourier mode grows or decays each time step, the discrete counterpart to $e^{-Dk^2\Delta t}$. Note that ξ depends on $k\Delta x$, henceforth we will emphasize this dependence by writing $\xi(k\Delta x)$. Using linearity of the finite difference scheme (14.27) we can write its solution as

$$u_j^n = \sum_{k=1}^{\infty} b_k\, \xi^n(k\Delta x)\sin(kj\Delta x), \quad j = 1, 2, \ldots, M-1. \tag{14.36}$$

Since $u_j^0 = f(j\Delta x)$ it follows that the coefficients b_k are the sine coefficients of the initial condition f and are thus given by (14.11). Therefore,

$$|u_j^n| \leq \sum_{k=1}^{\infty} |b_k||\xi(k\Delta x)|^n. \tag{14.37}$$

If $|\xi(k\Delta x)| \leq 1$ for all possible values of $k\Delta x$, then

$$|u_j^n| \leq \sum_{k=1}^{\infty} |b_k| = \text{constant}, \tag{14.38}$$

where we have assumed that the initial condition has an absolutely convergent sine series. That is, the numerical approximation is guaranteed to be bounded as $n \to \infty$ if $|\xi(k\Delta x)| \leq 1$. On the other hand if for some k^*, $|\xi(k^*\Delta x)| > 1$, then the corresponding Fourier mode, $b_{k^*}\xi^n\sin(k^*j\Delta x)$, will grow without a bound as $n \to \infty$ if the initial condition has a nonzero b_{k^*}. Setting $\theta = k\Delta x$, we conclude that the finite difference scheme (14.17) is numerically stable *if and only if*

$$|\xi(\theta)| \leq 1, \quad \forall\, \theta \in [0, \pi], \tag{14.39}$$

and using (14.35) this condition translates into

$$-1 \leq 1 - 2\alpha(1 - \cos\theta) \leq 1, \quad \forall\, \theta \in [0, \pi]. \tag{14.40}$$

Since $\alpha > 0$ the second inequality is always satisfied. From the first inequality, noting that the maximum of $1 - \cos\theta$ occurs for $\theta = \pi$, we obtain that the scheme (14.17) is numerically stable *if and only if*

$$\alpha \leq 1/2. \tag{14.41}$$

This is the same condition we found earlier using a maximum norm estimate. However, the Fourier analysis for the finite difference scheme, which is commonly called *von Neumann analysis*, offers additional information on what happens if $\alpha > 1/2$. If I $|\xi(k\Delta x)| > 1$ for some k, the corresponding Fourier mode will not be bounded as $n \to \infty$. The mode that becomes most unstable is the one for which $|\xi|$ is the largest, i.e. when $k\Delta x \approx \pi$ or equivalently $k \approx \pi/\Delta x$. This is precisely the highest wave number ($k = M - 1$ in this case) mode we can resolve with a mesh of size Δx. Going back to our numerical experiment in Fig. 14.4(a)-(c) we see that the oscillations in the numerical approximation with $\alpha > 1/2$ have a wavelength of approximately $2\Delta x$. Moreover, the oscillations appear first in a localized region around the points where the underlying exact solution is less regular. *The short wavelength of the oscillations, its initial localized appearance, and fast amplitude growth as n increases are a telltale of numerical instability.*

It is important to note that due to the linearity of the finite difference scheme and its constant coefficients, we only need to examine the behavior of individual Fourier modes of the numerical approximation. This is the basis of the von Neumann analysis: to examine how the finite difference scheme evolves a (complex) Fourier mode $\xi^n e^{ikj\Delta x}$. The focus of this analysis is on stability at the interior nodes, not at the boundary, so the problem need not have periodic or homogeneous boundary conditions. For non-periodic boundary conditions, the stability of the numerical scheme at the boundary has to be considered separately.

For some finite difference schemes, ξ might also be a function of Δt. In this case the stability condition for the amplification factor has the milder form

$$|\xi(k\Delta x, \Delta t)| \leq 1 + C\Delta t, \tag{14.42}$$

where C is a constant or equivalently, $|\xi|^2 \leq 1 + \tilde{C}\Delta t$ for some constant $\tilde{C}$. The condition for $|\xi|^2$ is generally easier to check than (14.42) because it avoids the square root when ξ is complex.

14.1.2 Order of a Method and Consistency

It is instructive to compare the representations (14.9) and (14.36) of the exact solution and of the solution of the forward in time and centered in space finite difference scheme, respectively. It is clear that the amplification factor $\xi(k\Delta x)$ should be an approximation of $e^{-k^2 D\Delta t}$ for sufficiently small Δt and Δx. Keeping $\alpha = D\Delta t/(\Delta x)^2$ fixed, we have $k^2 D\Delta t = \alpha(k\Delta x)^2 = \alpha\theta^2$ and Taylor expanding

$$\xi(\theta) = 1 - 2\alpha\left[\frac{1}{2}\theta^2 - \frac{1}{24}\theta^4 + \dots\right] \tag{14.43}$$

$$e^{-\alpha\theta^2} = 1 - \alpha\theta^2 + \frac{1}{2}\alpha^2\theta^4 + \dots, \tag{14.44}$$

from which it follows that

$$\xi(k\Delta x) = e^{-k^2 D\Delta t} + O(\Delta t)^2. \tag{14.45}$$

This is reminiscent of the approximation of $e^{\lambda\Delta t}$ by the stability function for a first order one-step method (Section 13.12) and which gives the local truncation error of said method (up to the factor $y(t_n)/\Delta t$). Indeed, (14.45) is a consequence of the fact that the finite difference scheme (14.16) provides a $O(\Delta t)$ approximation to the time derivative and an $O(\Delta x)^2$ approximation to the spatial second derivative.

Definition 14.1. *The local discretization or truncation error* $\tau_j^{n+1}(\Delta t, \Delta x)$ *at* (t_{n+1}, x_j) *is given by*

$$\tau_j^{n+1}(\Delta t, \Delta x) = \frac{u(t_{n+1}, x_j) - \tilde{u}_j^{n+1}}{\Delta t}, \tag{14.46}$$

where $\tilde{u}_j^{n+1}$ *is computed by doing one step of the numerical method starting with the exact solution of the PDE IVP at time* t_n *for a one-step method or at times* $t_{n-(m-1)}$, …, t_{n-1}, t_n *for an* m*-step* $(m > 1)$ *method.*

The local discretization error of the finite difference scheme (14.16) at a point (x_j, t_{n+1}) is thus given by

$$\begin{aligned}\tau_j^{n+1}(\Delta t, \Delta x) = &\frac{u(t_{n+1}, x_j) - u(t_n, x_j)}{\Delta t}\\ &- D\frac{u(t_n, x_{j+1}) - 2u(t_n, x_j) + u(t_n, x_{j-1})}{(\Delta x)^2},\end{aligned} \tag{14.47}$$

where $u(t,x)$ is the exact solution of the PDE IVP[1]. As in the ODE case, the local truncation error can be interpreted as a measure of how well the exact solution of the PDE satisfies the finite difference scheme *locally*.

Assuming the exact solution has enough continuous derivatives, we can Taylor expand the right hand side of (14.47) around (t_n, x_j) to find

$$\begin{aligned}\tau_j^{n+1}(\Delta t, \Delta x) = u_t - Du_{xx} + \frac{1}{2}u_{tt}\Delta t - \frac{D}{12}u_{xxxx}(\Delta x)^2 \\ + O(\Delta t)^2 + O(\Delta x)^4,\end{aligned} \tag{14.48}$$

where all the derivatives on the right hand side are evaluated at (t_n, x_j). Since u is the exact solution, we have that

$$\tau_j^{n+1}(\Delta t, \Delta x) = O(\Delta t) + O(\Delta x)^2 \tag{14.49}$$

and we say that the finite difference method is of *order* 1 in time and of order 2 in space.

Definition 14.2. *A finite difference scheme is consistent with the PDE it is approximating at a fixed point (t_{n+1}, x_j) if*

$$\tau_j^{n+1}(\Delta t, \Delta x) \to 0, \quad \textit{as } \Delta t, \Delta x \to 0. \tag{14.50}$$

Consistency means that the exact solution of the PDE satisfies increasingly better the finite difference scheme as $\Delta t, \Delta x \to 0$. This is a necessary requirement for the finite difference scheme to approximate the PDE in question and not another equation. However, as we have seen, consistency is not sufficient to guarantee the finite difference approximation will get better as the mesh is refined. We also need stability ($\alpha \leq 1/2$ in this particular case).

14.1.3 Convergence

At a fixed point (t,x), we want u_j^n to be an accurate approximation of $u(t,x)$ and to improve as $\Delta t, \Delta x \to 0$, keeping $t = n\Delta t$, $x = j\Delta x$ fixed.

[1] Note that the finite difference operators, the forward in time and the standard second difference in space, can be defined at any point (x,t), not necessarily a grid point. Thus, the local truncation error is well-defined at each (t,x).

Definition 14.3. *The* global error *of the finite difference approximation at point* (t_n, x_j) *is given*

$$e_j^n(\Delta t, \Delta x) = u(t_n, x_j) - u_j^n, \tag{14.51}$$

where $u(t_n, x_j)$ *and* u_j^n *are the exact solution and the numerical approximation at* (t_n, x_j)*, respectively.*

Because of the linearity of the finite difference scheme it is easy to derive an equation for the global error and using both stability and consistency prove *convergence* of the numerical approximation to the exact solution, i.e $e_j^n(\Delta t, \Delta x) \to 0$ as $\Delta t, \Delta x \to 0$, keeping $t = n\Delta t, x = j\Delta x$ fixed.

Using (14.47) it follows that the exact solution satisfies

$$\begin{aligned} u(t_{n+1}, x_j) = \alpha u(t_n, x_{j+1}) + (1-2\alpha)u(t_n, x_j) + \alpha u(t_n, x_{j-1}) \\ + \Delta t\ \tau_j^{n+1}(\Delta t, \Delta x) \end{aligned} \tag{14.52}$$

and subtracting (14.19) from this equation we obtain

$$e_j^{n+1} = \alpha e_{j+1}^n + (1-2\alpha)e_j^n + \alpha e_{j-1}^n + \Delta t\ \tau_j^{n+1}(\Delta t, \Delta x), \tag{14.53}$$

where we have written e_j^n instead of $e_j^n(\Delta t, \Delta x)$ for short. Taking the absolute value, using the triangle inequality, and the *stability condition* $\alpha \le 1/2$ we have

$$|e_j^{n+1}| \le \alpha|e_{j+1}^n| + (1-2\alpha)|e_j^n| + \alpha|e_{j-1}^n| + \Delta t\ |\tau_j^{n+1}(\Delta t, \Delta x)|. \tag{14.54}$$

Now, taking the maximum over j, and using that (14.49) implies there exist constants C_1 and C_2 such that $|\tau_j^n(\Delta t, \Delta x)| \le C_1\Delta t + C_2(\Delta x)^2$ for sufficiently small Δt and Δx, we obtain

$$\|e^{n+1}\|_\infty \le \|e^n\|_\infty + C_1(\Delta t)^2 + C_2\Delta t(\Delta x)^2, \tag{14.55}$$

where again $\|e^{n+1}\|_\infty = \max_j |e_j^{n+1}|$, etc. Applying repeatedly this inequality it follows that

$$\begin{aligned} \|e^n\|_\infty &\le \|e^{n-1}\|_\infty + C_1(\Delta t)^2 + C_2\Delta t(\Delta x)^2 \\ &\le \|e^{n-2}\|_\infty + 2\left[C_1(\Delta t)^2 + C_2\Delta t(\Delta x)^2\right] \\ &\dots \\ &\le \|e^0\|_\infty + n\left[C_1(\Delta t)^2 + C_2\Delta t(\Delta x)^2\right]. \end{aligned} \tag{14.56}$$

But $n\Delta t \leq T$ and $\|e^0\|_\infty = 0$ (u_j^0 coincides with the initial condition), therefore

$$\|e^n\|_\infty \leq T\left[C_1\Delta t + C_2(\Delta x)^2\right], \tag{14.57}$$

for all n. The fact that the terms in the rectangular brackets go to zero as $\Delta t, \Delta x \to 0$ is a restatement of *consistency* and from this the convergence of the numerical approximation follows.

14.1.4 The Lax-Richmyer Equivalence Theorem

We just seen in one example the importance of *consistency and stability* in the notion of *convergence*. It is clear that both consistency and stability are necessary for convergence; for without consistency we would not be solving the correct problem in the limit as $\Delta t, \Delta x \to 0$ and without stability the numerical approximation (and therefore the global error) would not remain bounded as $n \to \infty$. For the case of well-posed, linear PDE IVP's consistency and stability are also sufficient for the convergence of a finite difference scheme. This is the content of the following fundamental theorem in the theory of finite difference methods.

Theorem 14.1. *The Lax-Richmyer Equivalence Theorem. A consistent finite difference scheme for a well-posed, linear initial value PDE problem is convergent if and only if it is stable.*

A rigorous proof of this result requires advanced functional analysis tools and will not be presented here.

14.2 The Method of Lines

One approach to construct numerical methods for PDE IVPs is to discretize in space but leave time dependance continuous. This produces a large ODE system to which, *in principle*, one can apply a suitable ODE method. This construction is known as the *method of lines* because times varies along the lines defined by the spatial nodes as Fig. 14.5 suggests. We emphasize "in principle" in the previous statement because a blind application of this technique, without an understanding of the underlying PDE to be solved, can have disastrous consequences.

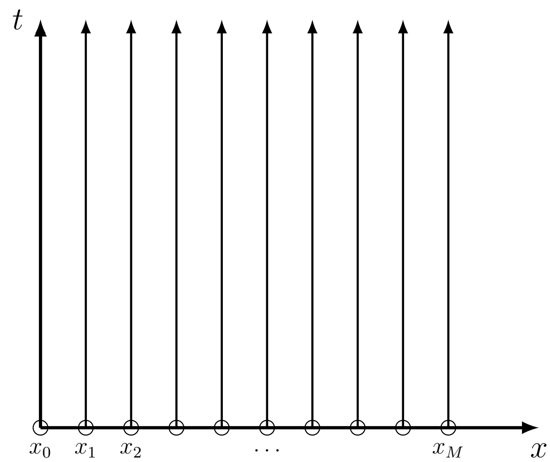

Figure 14.5: Method of lines. Space is discretized and time is left continuous.

To illustrate this approach let us consider again the one-dimensional heat equation and discretize the second derivative with respect to x using the standard, centered, second finite difference but leaving time continuous:

$$\frac{du_j(t)}{dt} = D\,\frac{u_{j-1}(t) - 2u_j(t) + u_{j+1}(t)}{(\Delta x)^2}, \quad j = 1, \ldots, M-1, \tag{14.58}$$

$$u_j(0) = f(x_j), \quad j = 1, \ldots, M-1, \tag{14.59}$$

where $u_0(t) = 0$ and $u_M(t) = 0$ and we are interested in solving this ODE system for $0 < t \leq T$. If we apply the forward Euler method to this ODE system we get the forward in time-centered in space scheme (14.16) we have analyzed in detail. But, as we will see in Section 14.6 with the simple equation $u_t + au_x = 0$, it is crucial to have an understanding of the PDE to be solved before applying the method of lines. The issue of numerical stability is also much more subtle than that for ODE methods.

14.3 The Backward Euler and Crank-Nicolson Methods

The forward in time-centered in space scheme (14.16) has a somewhat restrictive stability constraint, $\alpha \leq 1/2$, i.e.

$$\Delta t \leq \frac{1}{2D}(\Delta x)^2. \tag{14.60}$$

This is a quadratic stability constraint in Δx. For example, with $D = 5$, even for a modest spatial resolution of $\Delta t = 0.01$ we require a fairly small Δt, less than 10^{-5} and the constraint becomes more severe for D large. The computational cost associated with such small time-steps can be significant in higher spatial dimensions.

As we saw in Chapter 13 implicit methods offer larger A-stability regions than the corresponding explicit ones. In particular, the backward Euler method is A-stable. Let's use the method of lines and apply the backward Euler method to (14.58). We obtain

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = D\frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{(\Delta x)^2}, \quad j = 1, \ldots, M-1, \tag{14.61}$$

with the initial condition $u_j^0 = f(x_j)$, $j = 1, \ldots, M-1$, and the boundary conditions $u_0^{n+1} = 0$, $u_M^{n+1} = 0$. This is an *implicit* scheme. At each time step n, to update the numerical approximation for the future time step $n+1$ we need to solve this linear system for the $M-1$ unknowns, $u_1^{n+1}, \ldots, u_{M-1}^{n+1}$. Using again $\alpha = D\Delta t/(\Delta x)^2$ we can write (14.61) as

$$-\alpha u_{j-1}^{n+1} + (1+2\alpha)u_j^{n+1} - \alpha u_{j+1}^{n+1} = u_j^n, \quad j = 1, \ldots, M-1. \tag{14.62}$$

This is of course a tridiagonal linear system. It is also diagonally dominant and hence nonsingular. In fact it is also positive definite. Thus there is a unique solution and we can find it efficiently in $O(M)$ operations with the tridiagonal solver, Algorithm 9.5.

Let us look at the stability of (14.61) via von Neumann analysis. As before, we look at how the finite difference scheme evolves a Fourier mode $u_j^n = \xi^n e^{ikj\Delta x}$. Plugging this into (14.61) we get

$$\xi^{n+1}e^{ikj\Delta x} = \xi^n e^{ikj\Delta x} + \alpha\left[\xi^{n+1}e^{ik(j-1)\Delta x} - 2\xi^{n+1}e^{ikj\Delta x} + \xi^{n+1}e^{ik(j+1)\Delta x}\right].$$

Cancelling out the common factor $\xi^n e^{ikj\Delta x}$ and using that $\cos\theta = \frac{1}{2}(e^{i\theta}+e^{-i\theta})$ we obtain

$$\xi(k\Delta x) = \frac{1}{1+2\alpha(1-\cos(k\Delta x))}. \tag{14.63}$$

Since $\alpha > 0$ and $\cos\theta \leq 1$ we have that

$$|\xi(k\Delta x)| \leq 1 \tag{14.64}$$

for all $k \in \mathbb{Z}$, regardless of the value of α. Because there is no restriction on Δt to satisfy (14.64) we say that the backward in time-centered in space scheme (14.61) is *unconditionally stable*. It is easy to see that scheme (14.61) is first order in time and second order in space.

If we now use the method of lines again and the trapezoidal rule for the ODE system (14.58) we get the following second order in time and second order in space scheme:

$$\frac{u_j^{n+1}-u_j^n}{\Delta t} = \frac{D}{2}\left[\frac{u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1}}{(\Delta x)^2} + \frac{u_{j+1}^{n}-2u_j^{n}+u_{j-1}^{n}}{(\Delta x)^2}\right], \tag{14.65}$$

for $j = 1, \ldots, M-1$ with $u_j^0 = f(x_j)$, $j = 1, \ldots, M-1$, and $u_0^{n+1} = 0$, $u_M^{n+1} = 0$. This implicit method is known as Crank-Nicolson. As in the backward Euler method, we have a tridiagonal (diagonally dominant) linear system to solve for u_j^{n+1}, $j = 1, \ldots, M-1$ at each time step which can be done with the tridiagonal solver.

Let's do von Neumann analysis for the Crank-Nicolson method. Substituting a Fourier mode $u_j^n = \xi^n e^{ikj\Delta x}$ in (14.65) and cancelling the common term we get that the amplification factor is given by

$$\xi(k\Delta x) = \frac{1-\alpha(1-\cos(k\Delta x))}{1+\alpha(1-\cos(k\Delta x))} \tag{14.66}$$

and consequently

$$|\xi(k\Delta x)| \leq 1 \tag{14.67}$$

for all $k \in \mathbb{Z}$, independent of the value of α, that is, the Crank-Nicolson method is also unconditionally stable. However, note that $|\xi(k\Delta x)| \to 1$ as $\alpha \to \infty$ (recall that the trapezoidal rule method is not L-stable, i.e. not accurate in the stiff limit). Thus, for large α, ξ is not an accurate approximation of $e^{-k^2 D\Delta t}$, particularly for the high wavenumber modes (large $|k|$. As a result, the Crank-Nicolson method is not a good choice for problems with non-smooth data and large α.

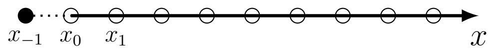

Figure 14.6: Neumann boundary condition at $x_0 = 0$. A "ghost point" (•), $x_{-1} = -\Delta x$ is introduced to implement the boundary condition.

14.4 Neumann Boundary Conditions

We look now briefly at how to apply a Neumann boundary condition with finite differences. Consider again the heat equation in the interval $[0, \pi]$ and suppose there is a homogeneous Neumann boundary condition on $x = 0$, $u_x(t, 0) = 0$, which enforces a no-heat flux across $x = 0$, and a homogenous Dirichlet boundary condition at $x = \pi$. Note that now the value of the solution at $x = 0$ is unknown (we only know its derivative). Thus, for each n we need to find the M values $u_0^n, u_1^n, \ldots, u_{M-1}^n$. For concreteness, let's consider the forward in time-centered in space scheme. As before

$$u_j^{n+1} = u_j^n + \alpha \left[u_{j+1}^n - 2u_j^n + u_{j-1}^n \right], \quad \text{for } j = 1, 2, \ldots, M-1, \tag{14.68}$$

with $\alpha = D\Delta t/(\Delta x)^2$. But now we also need an equation to update u_0^n. If we take $j = 0$ at (14.68) we get

$$u_0^{n+1} = u_0^n + \alpha \left[u_1^n - 2u_0^n + u_{-1}^n \right]. \tag{14.69}$$

However, this equation involves u_{-1}^n, an approximation corresponding to the point $x_{-1} = -\Delta x$, outside of the domain as Fig. 14.6 shows. We can eliminate this so-called *ghost point* by using the Neumann boundary condition:

$$0 = u_x(t_n, 0) \approx \frac{u_1^n - u_{-1}^n}{2\Delta x}, \tag{14.70}$$

where we are approximating the spatial derivative at $x = 0$ with the centered difference. With this approximation $u_{-1}^n = u_1^n$ and substituting this in (14.69) we obtain

$$u_0^{n+1} = u_0^n + 2\alpha \left[u_1^n - u_0^n \right]. \tag{14.71}$$

This equation together with (14.68) gives us the complete scheme.

14.5 Higher Dimensions and the ADI Method

We are going to consider now the heat equation in a rectangular domain $\Omega = [0, L_x] \times [0, L_y]$ as an example of an initial value problem in more than one spatial dimension. The problem is to find $u(t, x, y)$ for $0 < t \leq T$ and $(x, y) \in \Omega$ such that

$$u_t(t, x, y) = D\,\nabla^2 u(t, x, y), \quad (x, y) \in \Omega, \quad 0 < t \leq T, \tag{14.72}$$

$$u(0, x, y) = f(x, y), \quad (x, y) \in \Omega, \tag{14.73}$$

$$u(t, x, y) = g(x, y), \quad (x, y) \in \partial\Omega. \tag{14.74}$$

In (14.72), $\nabla^2 u = u_{xx} + u_{yy}$ is the Laplacian of u, also denoted[2] Δu, and $\partial\Omega$ in (14.74) denotes the boundary of Ω.

As in the one-dimensional case, we start by discretizing the domain. For simplicity, we are going to use a uniform grid. To this effect we choose positive integers M_x and M_y to partition $[0, L_x]$ and $[0, L_y]$, respectively and generate the *grid points or nodes*

$$(x_l, y_m) = (l\Delta x, m\Delta y), \quad l = 0, 1, \ldots, M_x, \quad m = 0, 1, \ldots, M_y, \tag{14.75}$$

where $\Delta x = L_x/M_x$ and $\Delta y = L_y/M_y$ are the grid sizes in the x and y direction, respectively. Also for simplicity we discretize time uniformly, $t_n = n\Delta t$, $n = 0, 1, \ldots, N$ with $\Delta t = T/N$, but variable time stepping can be useful in many problems. We seek for a numerical approximation $u_{l,m}^n$ of $u(t_n, x_l, y_m)$ at the interior nodes $l = 1, \ldots, M_x - 1$, $m = 1, M_y - 1$ and for $n = 1, \ldots, N$.

We now approximate the Laplacian of u at the interior nodes using centered finite differences for the second derivatives:

$$\nabla^2 u(t_n, x_l, y_m) \approx \frac{u_{l-1,m}^n - 2u_{l,m}^n + u_{l+1,m}^n}{(\Delta x)^2} + \frac{u_{l,m-1}^n - 2u_{l,m}^n + u_{l,m+1}^n}{(\Delta y)^2} \tag{14.76}$$

As in the 1D case, it is useful to introduce the following notation for the centered finite differences:

$$\delta_x^2 u_{l,m}^n = u_{l-1,m}^n - 2u_{l,m}^n + u_{l+1,m}^n, \tag{14.77}$$

$$\delta_y^2 u_{l,m}^n = u_{l,m-1}^n - 2u_{l,m}^n + u_{l,m+1}^n. \tag{14.78}$$

[2] We prefer not to use Δu for the Laplacian when discussing numerical methods to avoid confusion with the common notation employed for numerical increments or variations, such as $\Delta u = u(x + h) - u(x)$.

Assuming u is smooth enough,

$$\begin{aligned}\nabla^2 u(t_n, x_l, y_m) = \frac{1}{(\Delta x)^2}\delta_x^2 u(t_n, x_l, y_m) + \frac{1}{(\Delta y)^2}\delta_y^2 u(t_n, x_l, y_m) \\ + O(\Delta x)^2 + O(\Delta y)^2.\end{aligned} \tag{14.79}$$

The finite difference $\frac{1}{(\Delta x)^2}\delta_x^2 u_{l,m}^n + \frac{1}{(\Delta y)^2}\delta_y^2 u_{l,m}^n$ is called the *5-point* discrete Laplacian because it uses 5 grid points to approximate $\nabla^2 u(t_n, x_l, y_m)$.

The explicit, forward in time (forward Euler) can be written as

$$u_{l,m}^{n+1} = u_{l,m}^n + \alpha_x \delta_x^2 u_{l,m}^n + \alpha_y \delta_y^2 u_{l,m}^n, \tag{14.80}$$

for $n = 0, 1, \dots, N-1$ and $l = 1, \dots M_x - 1$, $m = 1, \dots, M_y - 1$. Here $\alpha_x = D\Delta t/(\Delta x)^2$ and $\alpha_y = D\Delta t/(\Delta y)^2$. As in the one-dimensional case, this scheme has a quadratic stability constraint. Unless the diffusion coefficient D is very small, it is better to employ an implicit method. For example, the Crank-Nicolson method can be written as

$$u_{l,m}^{n+1} = u_{l,m}^n + \frac{1}{2}\left[\alpha_x \delta_x^2 u_{l,m}^{n+1} + \alpha_y \delta_y^2 u_{l,m}^{n+1} + \alpha_x \delta_x^2 u_{l,m}^n + \alpha_y \delta_y^2 u_{l,m}^n\right], \tag{14.81}$$

for $n = 0, 1, \dots, N-1$ and $l = 1, \dots M_x-1$, $m = 1, \dots, M_y-1$. This is a linear system of $(M_x-1)\times(M_y-1)$ equations in the same number of unknowns. The structure of the matrix of coefficients, depends on how we label the unknowns. The most common labeling is the so-called *lexicographical order*, bottom to top and left to right: $u_{1,1}^{n+1}, u_{1,2}^{n+1}, \dots, u_{1,M_y-1}^{n+1}, u_{2,1}^{n+1}, u_{2,2}^{n+1}, \dots, u_{2,M_y-1}^{n+1}, \dots$, etc. (see Section 9.6). The result is a block tridiagonal, linear system which is symmetric and positive definite. This system could be solved, for example, with the (preconditioned) conjugate gradient method but it is more efficient to employ the following approach which splits the differentiation in each direction:

$$u_{l,m}^{*} = u_{l,m}^n + \frac{1}{2}\left[\alpha_x \delta_x^2 u_{l,m}^{*} + \alpha_y \delta_y^2 u_{l,m}^n\right], \tag{14.82}$$

$$u_{l,m}^{n+1} = u_{l,m}^{*} + \frac{1}{2}\left[\alpha_x \delta_x^2 u_{l,m}^{*} + \alpha_y \delta_y^2 u_{l,m}^{n+1}\right]. \tag{14.83}$$

Equation (14.82) can be viewed as a half-step ($\Delta t/2$) to produce an intermediate approximation $u_{l,m}^{*}$ by considering the differentiation in x implicitly and that in y explicitly. In the second half-step, Eq. (14.83), the situation is reversed; the differentiation in y is implicit while that in y is evaluated explicitly. The scheme (14.82)-(14.83) is called the *Alternating Direction Implicit* method or ADI.

Note that each half-step, gives us a tridiagonal linear system of equations, as in the one dimensional case, which can be solved efficiently with the tridiagonal solver. However, we need a boundary condition for $u^*_{l,m}$. It is easy to show that

$$u^*_{l,m} = u(t_n + \Delta t/2, x_l, y_m) + O(\Delta t) + O(\Delta x)^2 + O(\Delta y)^2. \tag{14.84}$$

Thus, we could take as boundary condition for u^* the boundary value of $u(t_n + \Delta t/2, x_l, y_m)$. It is remarkable that the second half-step corrects the $O(\Delta t)$ discretization error of the first half step to produce an $O(\Delta t)^2$ method that is also *unconditionally stable*!

14.6 Wave Propagation and Upwinding

We now look at a simple model for wave propagation. As we will see, numerical methods for these type of equations must obey a condition to ensure the numerical approximation evolves with the correct speed of propagation and the direction of the flow (upwinding) needs to be taken into account.

The model is the IVP for the one-way wave equation or transport equation:

$$u_t + a u_x = 0, \tag{14.85}$$
$$u(0,x) = f(x), \tag{14.86}$$

where a is constant and we are considering the problem, for the moment, in the entire real line. This linear, first order PDE can be solved easily by using the *method of characteristics*, which consists in employing curves (called characteristics) along which the PDE reduces to a simple ODE that can be readily integrated. For (14.85), the characteristics $X(t)$ are the curves that satisfy

$$\frac{dX(t)}{dt} = a \tag{14.87}$$
$$X(0) = x_0, \tag{14.88}$$

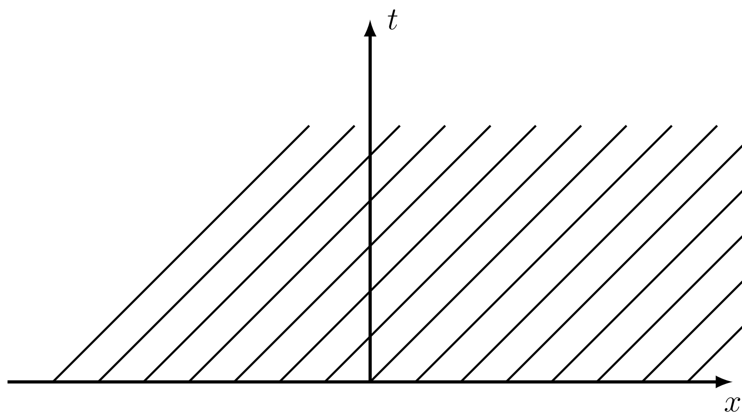

Figure 14.7: Characteristic curves $X(t) = x_0 + at$, for $u_t + u_x = 0$ with $a > 0$. Note that the slope of the characteristic lines is $1/a$.

where $x_0 \in \mathbb{R}$ is a starting point (we get one curve for each value of x_0) . Thus, the characteristics for (14.85)-(14.86) are the lines

$$X(t) = x_0 + at, \quad t \geq 0. \tag{14.89}$$

Figure 14.7 displays a few characteristics in the xt plane.

Let us look at u along the characteristics. We have

$$\frac{du(t, X(t))}{dt} = u_t + u_x \frac{dX}{dt} = u_t + a u_x = 0. \tag{14.90}$$

Thus, u is constant along the characteristic lines $X(t) = x_0 + at$ and consequently

$$u(t, X(t)) = u(0, X(0)) = f(x_0) = f\left(X(t) - at\right). \tag{14.91}$$

The solution to the pure IVP (14.85)-(14.86) is therefore

$$u(t, x) = f\left(x - at\right), \tag{14.92}$$

which corresponds to a traveling wave moving with speed a; the solution is just a translation of the initial condition f. If $a > 0$ the wave moves to the left and if $a < 0$ it moves to the right.

Suppose $a > 0$ and consider the finite difference scheme

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + a \frac{u_{j+1}^n - u_j^n}{\Delta x} = 0. \tag{14.93}$$

The local truncation error of this scheme is

$$\begin{aligned}\tau_j^{n+1}(\Delta t, \Delta x) &= \frac{u(t_{n+1}, x_j) - u(t_n, x_j)}{\Delta t} + a\frac{u(t_n, x_{j+1}) - u(t_n, x_j)}{\Delta x} \\ &= O(\Delta t) + O(\Delta x),\end{aligned} \tag{14.94}$$

assuming the exact solution is sufficiently smooth. Thus, the method (14.93) is consistent with $u_t + au_x = 0$. Let us do von Neumann analysis to look at the stability of this scheme. As in the example of the heat equation we take an individual Fourier mode $u_j^n = \xi^n e^{ikj\Delta x}$ and see how this evolves under the finite difference scheme. Substituting $u_j^n = \xi^n e^{ikj\Delta x}$ into (14.93) we get

$$\xi^n e^{ikj\Delta x}\left(\frac{\xi - 1}{\Delta t} + a\frac{e^{ik\Delta x} - 1}{\Delta x}\right) = 0 \tag{14.95}$$

and cancelling the common term and setting $\lambda = a\Delta t/\Delta x$ we obtain that the amplification factor satisfies

$$\xi = 1 + \lambda - \lambda e^{ik\Delta x}. \tag{14.96}$$

Since ξ is complex let us compute the square of its modulus

$$|\xi|^2 = (1 + \lambda - \lambda\cos\theta)^2 + (\lambda\sin\theta)^2, \tag{14.97}$$

where we have set $\theta = k\Delta x$. Developing the square and using $\sin^2\theta + \cos^2\theta = 1$ we have

$$|\xi|^2 = 1 + 2(1 + \lambda)\lambda - 2(1 + \lambda)\lambda\cos\theta. \tag{14.98}$$

Now, $\lambda > 0$ because $a > 0$. Thus, except for $\theta = \pi/2$

$$|\xi|^2 > 1 + 2(1 + \lambda)\lambda - 2(1 + \lambda)\lambda = 1. \tag{14.99}$$

Consequently, the scheme is *unstable* regardless of the value of Δt. On the other hand if $a < 0$ then $\lambda < 0$. It follows that

$$|\xi|^2 \leq 1, \text{ if and only if } -1 \leq \lambda < 0. \tag{14.100}$$

In other words, the scheme (14.93) is stable for $a < 0$ if and only if

$$|a|\frac{\Delta t}{\Delta x} \leq 1. \tag{14.101}$$

This stability constraint is know as the CFL condition (after Courant, Friedrichs, and Lewy). An interpretation of this condition is that the "numerical speed" $\Delta x/\Delta t$ must be greater or equal than the actual speed of propagation $|a|$. Similarly, for $a > 0$ the scheme

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + a\frac{u_j^n - u_{j-1}^n}{\Delta x} = 0 \tag{14.102}$$

is stable if and only if the CFL condition $a\Delta t/\Delta x \leq 1$ is satisfied. The approximation of au_x by a backward or forward finite difference depending on whether a is positive or negative, respectively is called *upwinding*, because we are using the direction of the flow (propagation) for our discretization:

$$au_x \approx \begin{cases} a\dfrac{u_j^n - u_{j-1}^n}{\Delta t} & \text{if } a > 0, \\[2ex] a\dfrac{u_{j+1}^n - u_j^n}{\Delta t} & \text{if } a < 0. \end{cases} \tag{14.103}$$

Let us look at another finite difference scheme for $u_t + au_x = 0$, this one with a centered difference to approximate u_x:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + a\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} = 0 \tag{14.104}$$

It is easy to show that

$$\tau_j^{n+1}(\Delta t, \Delta x) = O(\Delta t) + O(\Delta x)^2 \tag{14.105}$$

if the exact solution of $u_t + au_x = 0$ is smooth enough. Thus, the scheme is consistent. Let us do von Neumann analysis. Set $u_j^n = \xi^n e^{ikj\Delta x}$, we get

$$\xi^{n+1}e^{ikj\Delta x} = \xi^n e^{ikj\Delta x} - \frac{\lambda}{2}\xi^n\left[e^{ik(j+1)\Delta x} - e^{ik(j-1)\Delta x}\right], \tag{14.106}$$

cancelling $\xi^n e^{ikj\Delta x}$, setting $\theta = k\Delta x$ and using

$$\sin\theta = \frac{e^{i\theta} - e^{-i\theta}}{2i} \tag{14.107}$$

we find that the amplification factor satisfies

$$\xi = 1 - i\lambda\sin\theta. \tag{14.108}$$

Consequently

$$|\xi|^2 = 1 + \lambda^2 \sin^2\theta > 1 \tag{14.109}$$

except for $\theta = 0, \pi$. Therefore, scheme (14.104) is unconditionally unstable!

The previous three examples illustrate the importance of understanding the underlying PDE problem to be solved and the pitfalls that a blind application of the method of lines could cause.

The following finite different scheme is an example of a method that is not constructed by a direct discretization of the PDE and provides a stable modification of the second order in space scheme (14.104). First, we note that since $u_t + au_x = 0$ then

$$u_t = -au_x, \tag{14.110}$$
$$u_{tt} = -au_{xt} = -a(u_t)_x = -a(-au_x)_x = a^2 u_{xx}, \tag{14.111}$$

where we assumed the exact solution has continuous second derivatives. Moreover,

$$u(t+\Delta t) = u(t,x) + u_t(t,x)\,\Delta t + \frac{1}{2}u_{tt}(t,x)(\Delta t)^2 + O(\Delta t)^3 \tag{14.112}$$
$$= u(t,x) - au_x(t,x)\,\Delta t + \frac{1}{2}a^2 u_{xx}(t,x)(\Delta t)^2 + O(\Delta t)^3, \tag{14.113}$$

where we have used (14.110) and (14.111). Employing a centered, second order discretization for u_x and u_{xx} we obtain the following finite difference scheme:

$$u_j^{n+1} = u_j^n - \frac{\lambda}{2}(u_{j+1}^n - u_{j-1}^n) + \frac{\lambda^2}{2}(u_{j+1}^n - 2u_j^n + u_{j-1}^n), \tag{14.114}$$

with $\lambda = a\Delta t/\Delta x$ as before and is considered to be fixed. This numerical scheme is called *Lax-Wendroff*. By construction,

$$\tau_j^{n+1}(\Delta t, \Delta x) = O(\Delta t)^2 + O(\Delta x)^2 \tag{14.115}$$

so this is a consistent, second order method in space and time. It supports a Fourier mode $u_j^n = \xi^n e^{ikj\Delta x}$ provided

$$\xi = 1 - i\lambda\sin\theta - \lambda^2(1-\cos\theta) \tag{14.116}$$

with $\theta = k\Delta x$. Therefore,

$$|\xi|^2 = 1 - 4\lambda^2 \sin^2 \frac{1}{2}\theta + 4\lambda^4 \sin^4 \frac{1}{2}\theta + \lambda^2 \sin^2 \theta, \tag{14.117}$$

where we have used $1 - \cos\theta = 2\sin\frac{1}{2}\theta$. The right hand side of (14.117), let us call it $g(\theta)$, is an analytic and periodic function of θ. Thus, it achieves its extreme values at the critical points $g'(\theta) = 0$:

$$g'(\theta) = -4\lambda^2 \sin\frac{1}{2}\theta \cos\frac{1}{2}\theta + 8\lambda^2 \sin^3 \frac{1}{2}\theta \cos\frac{1}{2}\theta + 2\lambda^2 \sin\theta\cos\theta. \tag{14.118}$$

Therefore, $g'(\theta) = 0$ only for $\theta = 0, \pm\pi$. Moreover, $g(0) = 1$ so we only need to consider $\theta = \pm\pi$. Now,

$$|\xi|^2 \leq g(\pm\pi) = 1 - 4\lambda^2 + 4\lambda^4 \tag{14.119}$$

and $1 - 4\lambda^2 + 4\lambda^4 \leq 1$ implies $-4\lambda^2(1 - \lambda^2) \leq 0$ from which it follows that

$$|\lambda| \leq 1. \tag{14.120}$$

The Lax-Wendroff scheme is stable provided the CFL condition (14.120) is satisfied.

Our last example is the two-step method

$$\frac{u_j^{n+1} - u_j^{n-1}}{2\Delta t} + a\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} = 0. \tag{14.121}$$

This multistep finite difference scheme is know as the *leap frog* method. Like, the Lax-Wendroff, the leap frog is consistent and second order in space and time. As a multistep method, it requires another method to initialize it, i.e. to compute u_j^1. The Lax-Wendroff method could be used to that effect. Again, to do von Neumann analysis we substitute $u_j^n = \xi^n e^{ikj\Delta x}$ into the scheme. We obtain that the amplification factor in this case satisfies a quadratic equation (this is a two-step method):

$$\xi^2 + 2i\lambda\sin\theta\xi - 1 = 0, \tag{14.122}$$

with $\theta = k\Delta x$ and $\lambda = a\Delta t/\Delta x$ as before. The solutions of this quadratic equation are

$$\xi_{\pm} = -i\lambda\sin\theta \pm \sqrt{1 - \lambda^2 \sin^2\theta}. \tag{14.123}$$

If the roots are distinct then both Fourier modes $\xi_+^n e^{ikj\Delta x}$ and $\xi_-^n e^{ikj\Delta x}$ are solutions of the scheme and if $\xi_+ = \xi_-$ then $\xi_+^n e^{ikj\Delta x}$ and $n\xi_+^n e^{ikj\Delta x}$ are.

If $|\lambda| > 1$, for $\theta = \pi/2$ we have $|\xi_-| = |\lambda| + \sqrt{\lambda^2 - 1} > 1$. Therefore, the leap frog scheme is unstable for $|\lambda| > 1$. Now, for $|\lambda| \leq 1$

$$|\xi_+|^2 = |\xi_-|^2 = 1 - \lambda^2 \sin^2\theta + \lambda^2 \sin^2\theta = 1. \tag{14.124}$$

In this case $\xi_+ = \xi_-$ only when $|\lambda| = 1$ (and $\theta = \pi/2$) and because $n\xi_+^n e^{ikj\Delta x}$ is a solution, the leap frog scheme is stable if and only if $|\lambda| < 1$.

14.7 Advection-Diffusion

We consider now a PDE that models the combined effects of transport (also called advection) and diffusion. The equation is

$$u_t + a u_x = D u_{xx}, \tag{14.125}$$

where $D > 0$, is supplemented with initial and boundary conditions. Let us consider the following explicit finite difference scheme

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + a\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} = D\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2}. \tag{14.126}$$

This is a first order in time and second order in space method. With $\alpha = D\Delta t/(\Delta x)^2$ fixed, the advection term contributes $O(\Delta t)$ to the amplification factor in the von Neumann analysis. Thus, (14.42) applies and the stability is dictated by the discretization of the (higher order) diffusion term. That is, (14.126) is stable if and only if $\alpha \leq 1/2$.

Using the definitions of λ and α, (14.126) can be written as

$$u_j^{n+1} = (1 - 2\alpha)\, u_j^n + \left(\alpha - \frac{\lambda}{2}\right) u_{j+1}^n + \left(\alpha + \frac{\lambda}{2}\right) u_{j-1}^n. \tag{14.127}$$

Recall that for $D = 0$, (14.126) is unstable so it is important to examine the behavior of the numerical scheme when diffusion is much smaller than advection. To quantify this, we introduce a *numerical Péclet number*

$$\mu = \frac{1}{2}\left(\frac{\lambda}{\alpha}\right). \tag{14.128}$$

Then, we can write (14.127) as

$$u_j^{n+1} = (1-2\alpha)\,u_j^n + \alpha\,(1-\mu)\,u_{j+1}^n + \alpha\,(1+\mu)\,u_{j-1}^n. \tag{14.129}$$

If $|\mu| \leq 1$ and $\alpha \leq 1/2$, we have

$$|u_j^{n+1}| \leq (1-2\alpha)\,|u_j^n| + \alpha\,(1-\mu)\,|u_{j+1}^n| + \alpha\,(1+\mu)\,|u_{j-1}^n| \tag{14.130}$$

and taking the maximum over j

$$\|u^{n+1}\|_\infty \leq [1-2\alpha+\alpha\,(1-\mu)+\alpha\,(1+\mu)]\,\|u^n\|_\infty = \|u^n\|_\infty. \tag{14.131}$$

Therefore

$$\|u^{n+1}\|_\infty \leq \|u^n\|_\infty \quad \text{for all } n \tag{14.132}$$

and the scheme is *monotone*. Thus, if $\alpha \leq 1/2$ and $|\mu| \leq 1$ the finite difference method (14.126) is both stable and monotone. If on the other hand $|\mu| > 1$, there is no monotonicity and the numerical solution could be oscillatory. However, the oscillations would remain bounded as the scheme is stable for $\alpha \leq 1/2$.

The condition for monotonicity $|\mu| \leq 1$ means that

$$\Delta x \leq 2\frac{D}{|a|}. \tag{14.133}$$

This is a condition on the Δx needed to resolve the length scale associated with the diffusion process. It is not a stability constraint!

One way to avoid the oscillations when $|\mu| > 1$ is to use upwinding to approximate au_x. For example, for $a > 0$

$$\frac{u_j^{n+1}-u_j^n}{\Delta t} + a\frac{u_j^n - u_{j-1}^n}{\Delta x} = D\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2}, \tag{14.134}$$

which we can rewrite as

$$u_j^{n+1} = [1-2\alpha(1+\mu)]\,u_j^n + \alpha u_{j+1}^n + \alpha\,(1+2\mu)\,u_{j-1}^n. \tag{14.135}$$

Thus, we get monotonicity when $1-2\alpha(1+\mu) \geq 0$ i.e. when

$$2\alpha(1+\mu) \leq 1 \tag{14.136}$$

or equivalently

$$2D\frac{\Delta t}{(\Delta x)^2}\left(1+\frac{\Delta x}{2}\frac{a}{D}\right) \leq 1. \tag{14.137}$$

Thus, for a/D large (advection dominating diffusion) we get a much milder condition, close to the CFL.

14.8 The Wave Equation

Consider a stretched string of length L pinned at its end points. Assuming small deformations from its stretched horizontal state, the string vertical displacement at the time t and at the point x, $u(t,x)$, satisfies the wave equation

$$u_{tt}(t,x) = a^2 u_{xx}(t,x) \quad 0 < x < L, \quad t > 0, \tag{14.138}$$

with initial conditions $u(0,x) = f(x)$, $u_t(0,x) = g(x)$, and boundary conditions $u(t,0) = u(t,L) = 0$. Here, $a > 0$ is the speed of propagation.

It is instructive to consider the pure initial value problem (the so-called Cauchy problem) for the wave equation:

$$u_{tt}(t,x) = a^2 u_{xx}(t,x) \quad -\infty < x < \infty, \quad t > 0, \tag{14.139}$$

$$u(0,x) = f(x), \tag{14.140}$$

$$u_t(0,x) = g(x). \tag{14.141}$$

Using the characteristic coordinates

$$\mu = x + at, \tag{14.142}$$

$$\eta = x - at \tag{14.143}$$

and defining

$$U(\mu,\eta) = u(t(\mu,\eta), x(\mu,\eta)) \tag{14.144}$$

we have

$$U_\mu = u_t \frac{1}{2a} + u_x \frac{1}{2}, \tag{14.145}$$

$$U_{\mu\eta} = -u_{tt}\frac{1}{4a^2} + u_{tx}\frac{1}{4a} - u_{xt}\frac{1}{4a} + \frac{1}{4}u_{xx} \tag{14.146}$$

and assuming u has continuous second derivatives we get

$$U_{\mu\eta} = -\frac{1}{4a^2}\left[u_{tt} - a^2 u_{xx}\right] = 0. \tag{14.147}$$

that it U has the form

$$U(\mu,\eta) = F(\mu) + G(\eta) \tag{14.148}$$

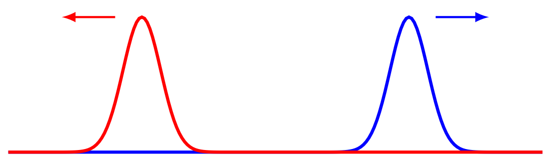

Figure 14.8: Solution of the pure initial value problem for the wave equation consists of a wave traveling to the left, $F(x + at)$, plus one traveling to the right, $G(x - at)$. Here $a > 0$.

for some functions F and G, to be determined by the initial conditions. Note that, going back to the original variables,

$$u(t, x) = F(x + at) + G(x - at). \tag{14.149}$$

So the solutions consists of the sum of a wave traveling to the left and one traveling to the right as Fig. 14.8 illustrates.

At $t = 0$

$$F(x) + G(x) = f(x), \tag{14.150}$$

$$aF'(x) - aG'(x) = g(x). \tag{14.151}$$

Integrating (14.151) we get

$$F(x) - G(x) = \frac{1}{a}\int_0^x g(s)ds + C, \tag{14.152}$$

where C is a constant. Combining (14.150) and (14.152) we find

$$F(x) = \frac{1}{2}f(x) + \frac{1}{2a}\int_0^x g(s)ds + \frac{1}{2}C, \tag{14.153}$$

$$G(x) = \frac{1}{2}f(x) - \frac{1}{2a}\int_0^x g(s)ds - \frac{1}{2}C, \tag{14.154}$$

and therefore the solution to the pure initial value problem of the the wave equations $u_{tt} - a^2 u_{xx} = 0$ is given by

$$u(t,x) = \frac{1}{2}\left[f(x+at) + f(x-at)\right] + \frac{1}{2a}\int_{x-at}^{x+at} g(s)ds, \tag{14.155}$$

an expression which is known as *D'Alembert's formula.*

Let us go back to the original initial boundary value problem for the deformations of a string. Consider the following finite difference scheme

$$\frac{u_j^{n+1} - 2u_j^n + u_j^{n-1}}{(\Delta t)^2} = a^2 \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2}, \tag{14.156}$$

where $\Delta x = L/M$ and the interval $[0, L]$ has been discretized with the equispaced points $x_j = j\Delta x$, for $j = 0, 1, \ldots, M$. This scheme is clearly a second order, both in space and time, and hence consistent with the wave equation. It is also a two-step method. To initialize this multistep scheme we use $u_j^0 = f(x_j)$ for $j = 1, 2, \ldots, M-1$, from the first initial condition, $u(0,x) = f(x)$, and to obtain u_j^1 we can employ the second initial condition, $u_t(0,x) = g(x)$, as follows

$$g(x_j) = u_t(0, x_j) \approx \frac{u_j^1 - u_j^0}{\Delta t} \tag{14.157}$$

that is

$$u_j^1 = u_j^0 + \Delta t\; g(x_j), \quad \text{for } j = 1, 2, \ldots, M-1. \tag{14.158}$$

Let us do von Neumann to look at the stability of (14.156). Substituting $u_j^n = \xi^n e^{ikj\Delta x}$ into (14.156) and cancelling the common term we find that

$$\xi - 2 + \frac{1}{\xi} = -4\lambda^2 \sin^2 \frac{1}{2}\theta, \tag{14.159}$$

where, as before, $\lambda = a\Delta t/\Delta x$ and $\theta = k\Delta x$. We can write (14.159) as

$$\left(\sqrt{\xi} - \frac{1}{\sqrt{\xi}}\right)^2 = \left(\pm 2i\lambda \sin\frac{1}{2}\theta\right)^2 \tag{14.160}$$

and thus

$$\sqrt{\xi} - \frac{1}{\sqrt{\xi}} = \pm 2i\lambda \sin\frac{1}{2}\theta. \tag{14.161}$$

Multiplying (14.161) by $\sqrt{\xi}$ we get

$$\xi \pm 2i\sqrt{\xi}\lambda \sin\frac{1}{2}\theta - 1 = 0. \tag{14.162}$$

This is a quadratic equation for $\sqrt{\xi}$ and its roots are

$$\xi_{\pm}^{1/2} = \pm i\lambda \sin\frac{1}{2}\theta \pm \sqrt{1 - \lambda^2 \sin^2\frac{1}{2}\theta} \tag{14.163}$$

and consequently

$$\xi_{\pm} = \left(\sqrt{1 - \lambda^2 \sin^2\frac{1}{2}\theta} \pm i\lambda \sin\frac{1}{2}\theta\right)^2. \tag{14.164}$$

Thus, $|\xi_{\pm}| \leq 1$ if and only if $|\lambda| \leq 1$. Also $\xi_+ = \xi_-$ for $\theta = 0$ or if $|\lambda| = 1$ and $\theta = \pi$. Recall that with equal roots, $n\xi_+^{n-1}e^{ikj\Delta x}$ is also a solution of the numerical scheme. However, since the wave equation is a second order PDE in time, it allows linearly growing solutions like Ct so the mode $n\xi_+^{n-1}e^{ikj\Delta x}$ with $|\xi_+| = 1$ is permissible here. We conclude that the scheme (14.156) is stable if and only if it satisfies the CFL condition

$$|\lambda| \leq 1. \tag{14.165}$$

14.9 Bibliographic Notes

Our main reference for the theory of finite difference method is the classical book by Richtmyer and Morton [RM67]. Other more modern, specialized texts covering finite differences, and which we have also used in this chapter, are the books by Strikwerda [Str04], Leveque [LeV07], Iserles [Ise08], and

Thomas [Tho98].

Section 14.1 . This section follows, with some variations, the masterful introduction in [RM67]. It is an attempt to present a simplified introduction to the main concepts and theory of finite difference without the use of more advanced mathematics. The monograph by Richtmyer and Morton [RM67] offers an unsurpassed treatment of the theory of finite differences and an elegant proof of the Lax-Richtmyer equivalence theorem [LR56]. The Fourier analysis for stability, known as von Neumann analysis, was first used by J. von Neumann in the 1940's but the first published version of it appeared in a paper by Crank and Nicolson in 1947 [CN47], as Gustafsson recounts [Gus18]. This widely used stability technique was only published until 1950 [CFvN50], with von Neumann as coauthor, in the more general form described in Subsection 14.1.1.

Section 14.2 . The method of lines applied to linear problems can be linked to A-stability of the ODE method employed [LeV07], with one important caveat. One is interested not on an ODE system of a finite size M but one for which $M \to \infty$ as $\Delta x \to 0$.

Section 14.3 . The Crank-Nicolson method for the heat equation was proposed by J. Crank and P. Nicolson in the aforementioned 1947 paper [CN47], where the (von Neumann) Fourier analysis to examine stability of finite differences is described. But the method was already mentioned in the pioneering 1911 paper on finite differences for PDE's by L F. Richardson, as a way to initialize a two step (the Leap Frog) method [Ric11][§2.2]. As described in this remarkable paper, the computations were done manually by people Richardson hired and called computers.

Section 14.4 . Here, we only presented one possibility for implementing a Neumann boundary condition. Other choices, using sided differences, are described in [Str04].

Section 14.5 . The ADI method was developed by D. W. Peaceman and H. H. Rachford [PR55] and by J. Douglas [Dou55]. The version of the ADI method presented here is the Peaceman-Rachford method. The natural extension of this method to 3D loses unconditional stability and the accuracy drops to first order in Δt. However, the more general ADI procedure proposed by Douglas and Gunn [DG64] written in terms of Crank-Nicolson intermediate steps produces an unconditionally stable and second order method in 3D and higher dimensions [RM67][8.8].

Section 14.6 . The method of characteristics for first order PDEs and the transport (one-way wave) equation are described in most PDE texts for example in [McO03, Eva10]. The CFL condition was introduced by R. Courant, K. O. Friedrichs, and H. Lewy in a remarkable paper [CFL28] (the English version is [CFL67]) that set the basis for understanding stability and convergence of finite difference methods, well before the advent of electronic computers. The Lax-Wendroff scheme was proposed by P. Lax and B. Wendroff in 1960 [LW60] and, as mentioned above, the leap frog FD was suggested for the heat equation in the landmark paper by Richardson [Ric11].

Section 14.7 . This section follows 6.4 in Strikwerda's text [Str04].

Section 14.7 . The representation formula for the solution of the pure initial value problem for the wave equation (d'Alembert's formula) was derived by J. R. d'Alembert in 1747 [d'A47]. The stability analysis of the centered scheme for the wave equation follows that in [Str04][8.2].

Bibliography

[Ale04] V. B. Alekseev. *Abel's Theorem in Problems and Solutions: Based on the lectures of Professor V.I. Arnold*. Kluwer Academic Publishers, 2004.

[BA83] F. Bashforth and J. C. Adams. *An Attempt to Test the Theories of Capillary Action by Comparing the Theoretical and Measured Forms of Drops of Fluid, with an Explanation of Integration Employed in Construction of Integrating the Tables Which Give the Theoretical Forms of Such Drops.* Cambridge University Press, 1883.

[BBC+94] R. Barrett, M.W. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, 1994.

[Bel97] R. Bellman. *Introduction to Matrix Analysis: Second Edition*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1997.

[Ben24] E. Benoît. Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés à un système d'équations linéaires en nombre inférieur à celui des inconnues, (procédé du commandant Cholesky). *Bulletin géodésique*, 2:67–77, 1924.

[Bre10] C. Brezinski. Some pioneers of extrapolation methods. In A. Bultheel and R. Cools, editors, *The Birth of Numerical Analysis*, pages 1–22. World Scientific, Singapore, 2010.

[BT04] J.-P. Berrut and L. N. Trefethen. Barycentric Lagrange interpolation. *SIAM Review*, 46(3):501–517, 2004.

[BT14] C. Brezinski and D. Tournès. *André-Louis Cholesky. Mathematician, Topographer and Army Officer*. Birkhäuser Cham, 2014.

[But08] J.C. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2008.

[Cau40] A. Cauchy. Sur les fonctions interpolaires. *Comptes rendus de l'Académie des Sciences*, 11:775–789, November 1840.

[CC60] C. W. Clenshaw and A. R. Curtis. A method for numerical integration on an automatic computer. *Numerische Mathematik*, 2:197–205, 1960.

[CdB72] S. D. Conte and C. W. de Boor. *Elementary Numerical Analysis: An Algorithmic Approach*. McGraw-Hill Book Company, 2nd edition, 1972.

[CFL28] R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100:32–74, 1928.

[CFL67] R. Courant, K. Friedrichs, and H. Lewy. On the Partial Difference Equations of Mathematical Physics. *IBM Journal of Research and Development*, 11:215–234, March 1967.

[CFvN50] J. G. Charney, R. Fjörtoft, and J. von Neumann. Numerical integration of the barotropic vorticity equation. *Tellus*, 2(4):237–254, 1950.

[Che82] E. W. Cheney. *Introduction to Approximation Theory*. International series in pure and applied mathematics. McGraw-Hill Book Company, 2nd edition, 1982.

[Cho05] A.-L. Cholesky. Sur la résolution numérique des systèmes d'équations linéaires. *Bulletin de la Sabix*, 39:81–95, 2005.

[Cia89] P.G. Ciarlet. *Introduction to Numerical Linear Algebra and Optimisation*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 1989.

[CL84] A. Coddington and N. Levinson. *Theory of Ordinary Differential Equations*. International series in pure and applied mathematics. R.E. Krieger, 1984.

[CN47] J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Advances in Computational Mathematics*, 6:207–226, 1947.

[CT65] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19:297–301, 1965.

[d'A47] J. R. d'Alembert. Recherches sur la courbe que forme une corde tenduë mise en vibration. *Histoire de l'académie royale des sciences et belles lettres de Berlin*, 3:214–219, 1747.

[Dah56] G. Dahlquist. Convergence and stability in the numerical integration of ordinary differential equations. *MATHEMATICA SCANDINAVICA*, 4:33–53, Dec. 1956.

[Dav75] P. J. Davis. *Interpolation and Approximation*. Dover Publications, 1975.

[dB78] C. de Boor. *A Practical Guide to Splines*, volume 27 of *Applied Mathematical Sciences*. Springer, New York, 1978.

[Dem97] J.W. Demmel. *Applied Numerical Linear Algebra*. EngineeringPro collection. Society for Industrial and Applied Mathematics, 1997.

[DG64] J. Douglas and J. E. Gunn. A general formulation of alternating direction methods. *Numerische Mathematik*, 6(1):428–453, 1964.

[dLVP19] C. J. de La Vallée Poussin. *Leçons sur l'approximation des fonctions d'une variable réelle*. Collection de monographies sur la théorie des fonctions. Gauthier-Villars, 1919.

[Dou55] J. Douglas. On the numerical integration $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial u}{\partial t}$ by implicit methods. *Journal of the Society for Industrial and Applied Mathematics*, 3(1):42–65, 1955.

[DR84] P. J. Davis and P. Rabinowitz. *Methods of Numerical Integration*. Academic Press, INC, London, second edition, 1984.

[Erd64] P. Erdős. Problems and results on the theory of interpolation. II. *Acta Mathematica Academiae Scientiarum Hungarica*, 12(1):235–244, 1964.

[Eul68] L. Euler. *Institutionum calculi integralis*. Number V. 1 in Institutionum calculi integralis. imp. Acad. imp. Saènt., 1768.

[Eva10] L.C. Evans. *Partial Differential Equations*. Graduate studies in mathematics. American Mathematical Society, 2010.

[Far02] G. Farin. Chapter 1 - A history of curves and surfaces in CAGD. In G. Farin, J. Hoschek, and M.-S. Kim, editors, *Handbook of Computer Aided Geometric Design*, pages 1–21. North-Holland, Amsterdam, 2002.

[For96] B. Fornberg. *A practical guide to pseudospectral methods*. Cambridge University Press, 1996.

[Fra61] J. G. F. Francis. The QR Transformation A Unitary Analogue to the LR Transformation—Part 1. *The Computer Journal*, 4(3):265–271, 01 1961.

[Fra62] J. G. F. Francis. The QR Transformation—Part 2. *The Computer Journal*, 4(4):332–345, 01 1962.

[Fre71] G. Freud. *Orthogonal Polynomials*. Pergamon Press, 1971.

[Gau16] C.F. Gauss. *Methodus nova integralium valores per approximationem inveniendi*. Dietrich, 1816.

[Gau81] W. Gautschi. A survey of Gauss-Christoffel quadrature formulae. In P. L. Butzer and F. Fehér, editors, *E. B. Christoffel: The Influence of His Work on Mathematics and the Physical Sciences*. Birkhäuser, Basel, 1981.

[Gau04] W. Gautschi. *Orthogonal Polynomials: Computation and Approximation*. Numerical Mathematics and Science. Oxford University Press, 2004.

[Gau11] W. Gautschi. *Numerical Analysis*. Birkhäuser, Boston, 2011.

[GC12] A. Greenbaum and T. P. Chartier. *Numerical Methods: Design, Analysis, and Computer Implementation of Algorithms*. Princeton University Press, 2012.

[Gen72] W. M. Gentleman. Implementing Clenshaw-Curtis quadrature, I. Methodology and experience. *Communications of the ACM*, 15(5):337–342, 1972.

[GLR07] A. Glaser, X. Liu, and V. Rokhlin. A fast algorithm for the calculation of the roots of special functions. *SIAM Journal on Scientific Computing*, 29(4):1420–1438, 2007.

[Gol77] H. H. Goldstine. *A history of numerical analysis. From the 16th century through the 19th century*. Studies in the History of Mathematics and Physical Sciences 2. Springer-Verlag, 1977.

[GR71] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Linear Algebra*, pages 134–151, 1971.

[Grc11] J. Grcar. Mathematicians of Gaussian elimination. *Notices of the American Mathematical Society*, 58(06):782–792, 2011.

[Gre97] Anne Greenbaum. *Iterative Methods for Solving Linear Systems*. Society for Industrial and Applied Mathematics, 1997.

[Gus18] B. Gustafsson. *Scientific Computing: A Historical Perspective*. Texts in Computational Science and Engineering. Springer International Publishing, 2018.

[GVL13] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013.

[GW69] G. H. Golub and J. H. Welsch. Calculation of Gauss quadrature rules. *Mathematics of Computation*, 23(106):221–230, May 1969.

[Hac94] W. Hackbusch. *Iterative Solution of Large Sparse Systems of Equations*. Springer-Verlag, Applied mathematical sciences, 1994.

[Hen61] P. Henrici. Two remarks on the Kantorovich inequality. *The American Mathematical Monthly*, 68(9):904–906, 1961.

[Hen62] P. Henrici. *Discrete Variable Methods in Ordinary Differential Equations*. Wiley, 1962.

[Hen64] P. Henrici. *Elements of Numerical Analysis*. Wiley, 1964.

[Her77] C. Hermite. Sur la formule d'interpolation de Lagrange. *Journal für die reine und angewandte Mathematik*, 84:70–79, 1877.

[Heu00] K. Heun. Neue methode zur approximativen integration der differentialgleichungen einer unabhängigen variablen. *Z. Math. Phys.*, 45:23–38, 1900.

[Hil13] F. B. Hildebrand. *Introduction to Numerical Analysis: Second Edition*. Dover Books on Mathematics. Dover Publications, 2013.

[HJ94] R.A. Horn and C.R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1994.

[HJ13] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 2013.

[HJB85] M. T. Heideman, D. H. Johnson, and C. S. Burrus. Gauss and the history of the fast Fourier transform. *Archive for History of Exact Sciences*, 34(3):265–277, 1985.

[HNW93] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I Nonstiff problems*. Springer, Berlin, second edition, 1993.

[HNW96] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer, second edition, 1996.

[Hou58] A. S. Householder. Unitary triangularization of a nonsymmetric matrix. *J. ACM*, 5(4):339–342, oct 1958.

[HS52] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J Res NIST*, 49(6):409–436, 1952.

[HSD04] M.W. Hirsch, S. Smale, and R.L. Devaney. *Differential Equations, Dynamical Systems, and an Introduction to Chaos*. Pure and Applied Mathematics - Academic Press. Elsevier Science, 2004.

[HT13] N. Hale and A. Townsend. Fast and accurate computation of Gauss–Legendre and Gauss–Jacobi quadrature nodes and weights. *SIAM Journal on Scientific Computing*, 35(2):A652–A674, 2013.

[HTF09] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, 2009.

[Huy09] C. Huygens. *De Circuli Magnitudine Inventa (1654)*. Kessinger Publishing, 2009.

[IK94] E. Isaacson and H. B. Keller. *Analysis of Numerical Methods*. Dover Books on Mathematics. Dover Publications, 1994.

[Ise08] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge University Press, USA, 2nd edition, 2008.

[KC02] D.R. Kincaid and E.W. Cheney. *Numerical Analysis: Mathematics of Scientific Computing*. (The Sally Series; Pure and Applied Undergraduate Texts, Vol. 2). Brooks/Cole, 2002.

[Kry12] V. I. Krylov. *Approximate Calculation of Integrals*. Dover books on mathematics. Dover Publications, 2012.

[Kub62] V.N. Kublanovskaya. On some algorithms for the solution of the complete eigenvalue problem. *USSR Computational Mathematics and Mathematical Physics*, 1(3):637–657, 1962.

[Kut01] W. Kutta. Beitrag zur näherungsweisen integration totaler differentialgleichungen. *Z. Math. Phys.*, 46:23–38, 1901.

[Lam91] J.D. Lambert. *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. John Wiley and Sons, 1991.

[LeV07] R. J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. Society for Industrial and Applied Mathematics, 2007.

[LR56] P. D. Lax and R. D. Richtmyer. Survey of the stability of linear finite difference equations. *Communications on Pure and Applied Mathematics*, 9(2):267–293, 1956.

[LS66] J. D. Lambert and B. Shaw. A generalisation of multistep methods for ordinary differential equations. *Numerische Mathematik*, 8(3):250–263, 1966.

[LW60] P. Lax and B. Wendroff. Systems of conservation laws. *Communications on Pure and Applied Mathematics*, 13(2):217–237, 1960.

[LY08] D.G. Luenberger and Y. Ye. *Linear and Nonlinear Programming*. International Series in Operations Research & Management Science. Springer US, 2008.

[McO03] R.C. McOwen. *Partial Differential Equations: Methods and Applications*. Prentice Hall, 2003.

[Mul56] D. E. Muller. A method for solving algebraic equations using an automatic computer. *Mathematics of Computation*, 10:208–215, 1956.

[NCW99] I. Newton, I.B. Cohen, and A. Whitman. *The Principia: Mathematical Principles of Natural Philosophy*. The Principia: Mathematical Principles of Natural Philosophy. University of California Press, 1999.

[NW06] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.

[OR70] J.M. Ortega and W.C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1970.

[Ove01] M. L. Overton. *Numerical Computing with IEEE Floating Point Arithmetic: Including One Theorem, One Rule of Thumb, and One Hundred and One Exercises*. Other titles in applied mathematics. Society for Industrial and Applied Mathematics, 2001.

[PR55] D. W. Peaceman and H. H. Rachford. The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for Industrial and Applied Mathematics*, 3(1):28–41, 1955.

[Rap90] J. Raphson. *Analysis aequationum universalis seu ad aequationes algebraicas resolvendas methodus generalis, & expedita, ex nova infinitarum serierum methodo, deducta ac demonstrata*. 1690.

[Ric11] L. F. Richardson. IX. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 210(459-470):307–357, 1911.

[Riv81] T. J. Rivlin. *An Introduction to the Approximation of Functions*. Dover Publications, 1981.

[Riv20] T. J. Rivlin. *Chebyshev Polynomials*. Dover Publications, 2020.

[RM67] R.D. Richtmyer and K.W. Morton. *Difference Methods for Initial Value Problems*. Wiley, 1967.

[Rom55] W. Romberg. Vereinfachte numerische integration. *Det Kongelige Norske Videnskabers Selskab Forhandlinger*, 28(7):30–36, 1955.

[RR01] A. Ralston and P. Rabinowitz. *A First Course in Numerical Analysis*. Dover books on mathematics. Dover Publications, 2001.

[Run95] C. Runge. Ueber die numerische auflösung von differentialgleichungen. *Mathematische Annalen*, 46:167–178, 1895.

[Run01] C. Runge. Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten. *Zeit. fur Math. Physik*, 246:224–243, 1901.

[Saa03] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, second edition, 2003.

[Sal72] H. E. Salzer. Lagrangian Interpolation at the Chebyshev Points $x_{n,\nu} \equiv \cos(\nu\pi/n), \nu = O(1)n$; some Unnoted Advantages. *The Computer Journal*, 15(2):156–159, 1972.

[Sau12] T. Sauer. *Numerical Analysis.* Pearson Addison Wesley, second edition, 2012.

[SB02] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* Texts in Applied Mathematics. Springer, New York, 2002.

[Sch89] H. R. Schwarz. *Numerical Analysis: A Comprehensive Introduction.* Wiley, 1989. With a contribution by J. Waldvogel.

[Sid13] T.C. Sideris. *Ordinary Differential Equations and Dynamical Systems.* Atlantis Studies in Differential Equations. Atlantis Press, 2013.

[Sim43] T. Simpson. *Mathematical Dissertations on a Variety of Physical and Analytical Subjects.* T. Woodward, London, 1743.

[Str04] J. Strikwerda. *Finite Difference Schemes and Partial Differential Equations.* Other titles in applied mathematics. Society for Industrial and Applied Mathematics, 2004.

[Sze39] G. Szegő. *Orthogonal Polynomials.* American Mathematical Society, 1939.

[TB97] L.N. Trefethen and D. Bau. *Numerical Linear Algebra.* Society for Industrial and Applied Mathematics, 1997.

[Tho98] J.W. Thomas. *Numerical Partial Differential Equations: Finite Difference Methods.* Texts in Applied Mathematics. Springer New York, 1998.

[Tim94] A. F. Timan. *Theory of Approximation of Functions of a Real Variable.* Dover books on advanced mathematics. Dover Publications, 1994.

[Tre92] L. N. Trefethen. The definition of numerical analysis. *SIAM News*, 25, November 1992.

[Tre00] L. N. Trefethen. *Spectral Methods in MATLAB.* Society for Industrial and Applied Mathematics, 2000.

[Tre08] L. N. Trefethen. Is Gauss quadrature better than Clenshaw–Curtis? *SIAM Review*, 50(1):67–87, 2008.

[Wil65] J.H. Wilkinson. *The Algebraic Eigenvalue Problem*. Monographs on numerical analysis. Clarendon Press, 1965.

[Wil94] J. H. Wilkinson. *Rounding Errors in Algebraic Processes*. Dover books on advanced mathematics. Dover, 1994.