Introduction to Numerical Analysis

Hector D. Ceniceros

Contents

List of Figures

List of Tables

Preface

These notes were prepared by the author for use in the upper division undergraduate course of Numerical Analysis (Math 104 ABC) at the University of California at Santa Barbara. They were written with the intent to emphasize the foundations of Numerical Analysis rather than to present a long list of numerical methods for different mathematical problems.

We begin with an introduction to Approximation Theory and then use the different ideas of function approximation in the derivation and analysis of many numerical methods.

These notes are intended for undergraduate students with a strong mathematics background. The prerequisites are Advanced Calculus, Linear Algebra, and introductory courses in Analysis, Differential Equations, and Complex Variables. The ability to write computer code to implement the numerical methods is also a necessary and essential part of learning Numerical Analysis.

These notes are not in finalized form and may contain errors, misprints, and other inaccuracies. They cannot be used or distributed without written consent from the author.

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 basically anything where computation is involved.

1.2 An Illustrative Example: Approximating a Definite Integral

The main principles and objectives of Numerical Analysis are better illustrated with concrete examples and this is the purpose of this chapter.

Consider the problem of calculating 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 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 linear polynomial $p_1(x)$, that passes through $(a, f(a))$ and $(b, f(b))$. That is

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

and

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

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 Trapezoidal Rule Quadrature*. A quadrature 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, i.e. it would be exact. The underlying question is: how well does a linear polynomial p_1, satisfying

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

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

approximate f on the interval $[a, b]$? We can almost guess the answer. The approximation is exact at $x = a$ and $x = b$ because of (1.5)-(1.6) and it is

exact for all polynomials of degree ≤ 1. This suggests that $f(x) - p_1(x) = Cf''(\xi)(x-a)(x-b)$, where C is a constant. But where is f'' evaluated at? it cannot be at x for if it did f would be the solution of a second order ODE and f is an arbitrary (but sufficiently smooth, $C^2[a,b]$) function so it has to be at some undetermined point $\xi(x)$ in (a,b). Now, if we take the particular case $f(x) = x^2$ on $[0,1]$ then $p_1(x) = x$, $f(x) - p_1(x) = x(x-1)$, and $f''(x) = 2$, which implies that C would have to be $1/2$. So our conjecture is

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

There is a beautiful 19th Century proof of this result by A. Cauchy. It goes as follows. If $x = a$ or $x = b$ (1.7) holds trivially. So let us take x in (a,b) and define the following function of a new variable t as

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

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 get 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 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 if we shift 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

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

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, apply the Trapezoidal Rule on each of them, and sum up the result.

Let us divide $[a,b]$ in N 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 Quadrature* (CTR):

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

The error term 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 ($f \in C^2[a,b]$) then by the Intermediate Value Theorem, there is a point $\xi \in (a,b)$ such that $f''(\xi)$ is equal to the quantity in the brackets so we obtain that

$$E_h[f] = -\frac{1}{12}(b-a)h^2 f''(\xi), \tag{1.20}$$

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

1.2.3 Convergence and Rate of Convergence

We do not not know what the point ξ is in (1.20). If we knew, the error could be evaluated and we would know the integral exactly, at least in principle, because

$$I[f] = T_h[f] + E_h[f]. \tag{1.21}$$

But (1.20) gives us two important properties of the approximation method in question. First, (1.20) 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 $h \to 0$ [1]. Recall $h = (b-a)/N$, so as we increase N our approximation to the integral gets better and better. Second, (1.20) 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) then 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 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 $e-1$. Observe how the error $|I[e^x] - T_{1/N}[e^x]|$ decreases by a factor of (approximately) 1/4 as N is doubled, in accordance to (1.20).*

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

N	$T_{1/N}[e^x]$	$\lvert I[e^x] - T_{1/N}[e^x]\rvert$	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

1.2.4 Error Correction

We can get an upper bound for the error using (1.20) and 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

$$|E_h[f]| \leq \frac{1}{12}(b-a)h^2 M_2. \tag{1.22}$$

[1] Neglecting round-off errors introduced by finite precision number representation and computer arithmetic.

However, this bound does not in general provide an accurate estimate of the error. It could grossly overestimate it. This can be seen from (1.19). As $N \to \infty$ the term in brackets converges to a mean value of f'', i.e.

$$\frac{1}{N}\sum_{j=0}^{N-1} f''(\xi_j) \longrightarrow \frac{1}{b-a}\int_a^b f''(x)dx = \frac{1}{b-a}[f'(b) - f'(a)], \tag{1.23}$$

as $N \to \infty$, which could be significantly smaller than the maximum of $|f''|$. Take for example $f(x) = e^{100x}$ on $[0, 1]$. Then $\max |f''| = 10000e^{100}$, whereas the mean value (1.23) is equal to $100(e^{100} - 1)$ so the error bound (1.22) overestimates the actual error by two orders of magnitude. Thus, (1.22) is of little practical use.

Equation (1.19) and (1.23) suggest that asymptotically, that is for sufficiently small h,

$$E_h[f] = C_2h^2 + R(h), \tag{1.24}$$

where

$$C_2 = -\frac{1}{12}[f'(b) - f'(a)] \tag{1.25}$$

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

$$\lim_{h\to 0} \frac{R(h)}{h^2} = 0. \tag{1.26}$$

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

$$I[f] = T_h[f] + C_2h^2 + R(h). \tag{1.27}$$

and, for *sufficiently small* h, C_2h^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_2h^2 and use this leading order approximation of the error to obtain the improved approximation

$$\widetilde{T}_h[f] = T_h[f] - \frac{1}{12}[f'(b) - f'(a)]h^2. \tag{1.28}$$

This is called the (composite) *Modified Trapezoidal Rule*. It then follows from (1.27) that error of this "corrected approximation" is $R(h)$, which goes to zero faster than h^2. In fact, we will prove later 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 (1.27). If we halve h we get

$$I[f] = T_{h/2}[f] + \frac{1}{4}C_2h^2 + R(h/2). \tag{1.29}$$

Subtracting (1.29) from (1.27) we get

$$C_2h^2 = \frac{4}{3}\left(T_{h/2}[f] - T_h[f]\right) + \frac{4}{3}\left(R(h/2) - R(h)\right). \tag{1.30}$$

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

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

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

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

The key here is that h has to be sufficiently small to make the asymptotic approximation (1.31) valid. We can check this by working backwards. If h is sufficiently small, then evaluating (1.31) at $h/2$ we get

$$C_2\left(\frac{h}{2}\right)^2 \approx \frac{4}{3}\left(T_{h/4}[f] - T_{h/2}[f]\right) \tag{1.33}$$

and consequently the ratio

$$q(h) = \frac{T_{h/2}[f] - T_h[f]}{T_{h/4}[f] - T_{h/2}[f]} \tag{1.34}$$

should be approximately 4. Thus, $q(h)$ offers a reliable, computable indicator of whether or not h is sufficiently small for (1.32) to be an accurate estimate of the error.

We can now use (1.31) and the idea of error correction to improve the accuracy of $T_h[f]$ with the following approximation [2]

$$S_h[f] := T_h[f] + \frac{4}{3}\left(T_{h/2}[f] - T_h[f]\right). \tag{1.35}$$

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 (1.29) by 4 and substracting (1.27) to the result we get

$$I[f] = \frac{4T_{h/2}[f] - T_h[f]}{3} + \frac{4R(h/2) - R(h)}{3} \tag{1.36}$$

Note that $S_h[f]$ is exactly the first term in the right hand side of (1.36) 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*

$$q\left(\frac{1}{16}\right) = \frac{T_{1/32}[e^x] - T_{1/16}[e^x]}{T_{1/64}[e^x] - T_{1/32}[e^x]} \approx 3.9998 \tag{1.37}$$

and the improved approximation is

$$S_{1/16}[e^x] = T_{1/16}[e^x] + \frac{4}{3}\left(T_{1/32}[e^x] - T_{1/16}[e^x]\right) = 1.718281837561771 \tag{1.38}$$

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

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

It appears that $S_h[f]$ gives us superior accuracy to that of $T_h[f]$ but at roughly twice the computational cost. If we group together the common terms 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

$$S_h[f] = \frac{h}{6}\left[f(a) + 2\sum_{k=1}^{N-1} f(a+kh) + 4\sum_{k=1}^{N-1} f(a+kh/2) + f(b)\right]. \quad (1.39)$$

The resulting quadrature formula $S_h[f]$ is known as the *Composite Simpson's Rule* and, as we will see in Chapter 8, can be derived by approximating the integrand by quadratic polynomials. 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.

Example 1.3. *Consider the integral*

$$I[1/(2+\sin x)] = \int_0^{2\pi} \frac{dx}{2+\sin x}. \quad (1.40)$$

Using Complex Variables techniques (Residues) the exact integral can be computed and $I[1/(2+\sin x)] = 2\pi/\sqrt{3}$. *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 show in Table 1.2.*

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

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

N	$T_{2\pi/N}[1/(2+\sin x)]$	$\lvert I[1/(2+\sin x)] - T_{2\pi/N}[1/(2+\sin x)]\rvert$
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 CTR 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 $f(x_0 = 0) = f(x_N = 2\pi)$ and the CTR becomes

$$T_h[f] = h\left[\frac{f(x_0)}{2} + f(x_1) + \ldots + f(x_{N-1}) + \frac{f(x_N)}{2}\right] = h\sum_{j=0}^{N-1} f(x_j). \quad (1.41)$$

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

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos kx + b_k \sin kx) \quad (1.42)$$

where

$$a_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos kx\, dx, \quad k = 0, 1, \ldots \quad (1.43)$$

$$b_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin kx\, dx, \quad k = 1, 2, \ldots \quad (1.44)$$

Using the Euler formula[3].

$$e^{ix} = \cos x + i \sin x \tag{1.45}$$

we can write

$$\cos x = \frac{e^{ix} + e^{-ix}}{2}, \tag{1.46}$$

$$\sin x = \frac{e^{ix} - e^{-ix}}{2i} \tag{1.47}$$

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 (1.42) becomes

$$f(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx}, \tag{1.48}$$

where

$$c_k = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-ikx} dx. \tag{1.49}$$

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 (1.48) in (1.41) we get

$$T_h[f] = h \sum_{j=0}^{N-1} \left(\sum_{k=-\infty}^{\infty} c_k e^{ikx_j} \right). \tag{1.50}$$

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

$$T_h[f] = \frac{2\pi}{N} \sum_{k=-\infty}^{\infty} c_k \sum_{j=0}^{N-1} e^{ik\frac{2\pi}{N}j}. \tag{1.51}$$

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

But

$$\sum_{j=0}^{N-1} e^{ik\frac{2\pi}{N}j} = \sum_{j=0}^{N-1} \left(e^{ik\frac{2\pi}{N}}\right)^j. \tag{1.52}$$

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

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

Otherwise, if $k \neq lN$, then

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

Using (1.53) and (1.54) we thus get that

$$T_h[f] = 2\pi \sum_{l=-\infty}^{\infty} c_{lN}. \tag{1.55}$$

On the other hand

$$c_0 = \frac{1}{2\pi}\int_0^{2\pi} f(x)dx = \frac{1}{2\pi} I[f]. \tag{1.56}$$

Therefore

$$T_h[f] = I[f] + 2\pi\left[c_N + c_{-N} + c_{2N} + c_{-2N} + \ldots\right], \tag{1.57}$$

that is

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

So now, the relevant question is how fast the Fourier coefficients c_{lN} of f decay with N. The answer is tied to the smoothness of f. Doing integration by parts in the formula (4.11) for the Fourier coefficients of f we have

$$c_k = \frac{1}{2\pi}\frac{1}{ik}\left[\int_0^{2\pi} f'(x)e^{-ikx}dx - f(x)e^{-ikx}\Big|_0^{2\pi}\right] \qquad k \neq 0 \tag{1.59}$$

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

$$c_k = \frac{1}{2\pi}\frac{1}{ik}\int_0^{2\pi} f'(x)e^{-ikx}dx \qquad k \neq 0. \tag{1.60}$$

Integrating by parts m times we obtain

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

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

$$|c_k| \leq \frac{A_m}{|k|^m}, \tag{1.62}$$

where A_m is a constant (depending only on m). Using this in (1.58) we get

$$\begin{aligned} |T_h[f] - I[f]| &\leq 2\pi A_m \left[\frac{2}{N^m} + \frac{2}{(2N)^m} + \frac{2}{(3N)^m} + \dots\right] \\ &= \frac{4\pi A_m}{N^m}\left[1 + \frac{1}{2^m} + \frac{1}{3^m} + \dots\right], \end{aligned} \tag{1.63}$$

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

$$|T_h[f] - I[f]| \leq \frac{C_m}{N^m}. \tag{1.64}$$

Thus, in this particular case, the rate of convergence of the CTR 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 CTR converges to the exact integral at a rate faster than any power of $1/N$ (or h)! This is called *super-algebraic convergence*.

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 an interval $[a, b]$, we would like to find a good approximation to it by simpler functions, such as polynomials, trigonometric polynomials, wavelets, rational functions, 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 simpler functions. These are the main topics of this introductory chapter to Approximation Theory.

2.1 Norms

A norm on a vector space V over a field $K = \mathbb{R}$ (or $\mathbb{C}$) 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) $\|\lambda x\| = |\lambda| \, \|x\| \ \forall x \in V, \ \lambda \in K$.

If we relax (i) to just $\|x\| \geq 0$, we obtain 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 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 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 $x \in V$ and for any two norms $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$ defined 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 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, which states that any continuous function can be approximated uniformly, i.e. using the norm $\|\cdot\|_\infty$, with arbitrary accuracy by a polynomial. This is the celebrated Weierstrass Approximation Theorem. We are going to present a constructive proof due to Sergei 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

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

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 $b_{k,n}(x)$ concentrate more and more around the points k/n close to x as Fig. 2.1 indicates for $b_{k,n}(0.25)$ and $b_{k,n}(0.75)$.

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 $\mathbf{P_0} = (x_0, y_0)$ and $\mathbf{P_1} = (x_1, y_1)$, the segment of the straight line connecting them can be written in parametric form as

$$B_1(t) = (1-t)\mathbf{P_0} + t\,\mathbf{P_1}, \qquad t \in [0,1]. \tag{2.20}$$

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

$$B_2(t) = (1-t)^2\mathbf{P_0} + 2t(1-t)\mathbf{P_1} + t^2\,\mathbf{P_2}, \qquad t \in [0,1]. \tag{2.21}$$

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 $B_2'(0) = 2(\mathbf{P_1} - \mathbf{P_0})$ and $B_2'(1) = 2(\mathbf{P_2} - \mathbf{P_1})$, 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. To allow the representation of complex shapes, quadratic or cubic Bézier curves

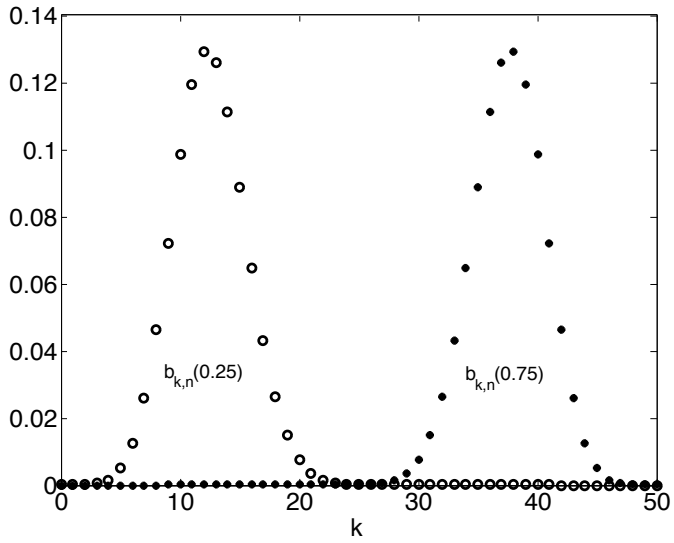

Figure 2.1: The Bernstein weights $b_{k,n}(x)$ for $x = 0.25$ ($\circ$)and $x = 0.75$ ($\bullet$), $n = 50$ and $k = 1 \ldots n$.

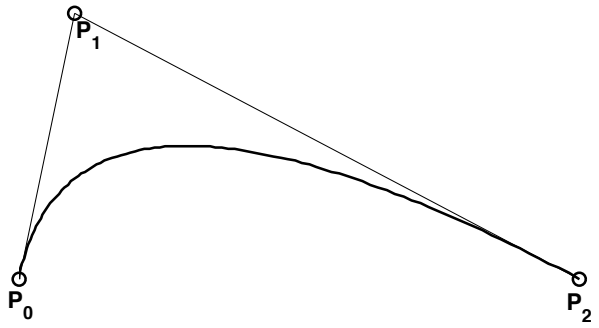

Figure 2.2: Quadratic Bézier curve.

are pieced together to form *composite Bézier curves*. 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. For example, the TrueType font used in 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 are stitched together.

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

$$\begin{aligned}\sum_{k=0}^{n} \frac{k}{n}\binom{n}{k} x^k (1-x)^{n-k} &= \sum_{k=1}^{n} \frac{kn!}{n(n-k)!k!} x^k (1-x)^{n-k} \\ &= x \sum_{k=1}^{n} \binom{n-1}{k-1} x^{k-1} (1-x)^{n-k} \\ &= x \sum_{k=0}^{n-1} \binom{n-1}{k} x^k (1-x)^{n-1-k} \\ &= x\left[x + (1-x)\right]^{n-1} = x.\end{aligned} \tag{2.22}$$

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

$$\sum_{k=0}^{n} \left(\frac{k}{n}\right)^2 \binom{n}{k} x^k (1-x)^{n-k} = \sum_{k=1}^{n} \frac{k}{n}\binom{n-1}{k-1} x^k (1-x)^{n-k} \tag{2.23}$$

and writing

$$\frac{k}{n} = \frac{k-1}{n} + \frac{1}{n} = \frac{n-1}{n}\frac{k-1}{n-1} + \frac{1}{n}, \tag{2.24}$$

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,

$$\sum_{k=0}^{n}\left(\frac{k}{n}\right)^2\binom{n}{k}x^k(1-x)^{n-k} = \frac{n-1}{n}x^2+\frac{x}{n}. \tag{2.25}$$

Now, expanding $\left(\frac{k}{n}-x\right)^2$ and using (2.19), (2.22), and (2.25) it follows that

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

2.2.2 Weierstrass Approximation Theorem

Theorem 2.1. *(Weierstrass Approximation Theorem) Let f be a continuous function in $[a,b]$. 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 simple 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

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

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

$$|f(x) - f(k/n)| < \frac{\epsilon}{2} \quad \text{if } |x - k/n| < \delta. \tag{2.28}$$

Moreover,

$$|f(x) - f(k/n)| \leq 2\|f\|_\infty \quad \text{for all } x \in [0,1],\ k = 0, 1, \ldots, n. \tag{2.29}$$

We now split the sum in (2.27) in two sums, one over the points such that $|k/n - x| < \delta$ and the other over the points such that $|k/n - x| \geq \delta$:

$$\begin{aligned} f(x) - B_n f(x) = &\sum_{|k/n-x|<\delta} \left[f(x) - f\left(\frac{k}{n}\right)\right] \binom{n}{k} x^k (1-x)^{n-k} \\ &+ \sum_{|k/n-x|\geq\delta} \left[f(x) - f\left(\frac{k}{n}\right)\right] \binom{n}{k} x^k (1-x)^{n-k}. \end{aligned} \tag{2.30}$$

Using (2.28) and (2.19) it follows immediately that the first sum is bounded by $\epsilon/2$. For the second sum we have

$$\begin{aligned} &\sum_{|k/n-x|\geq\delta} \left|f(x) - f\left(\frac{k}{n}\right)\right| \binom{n}{k} x^k (1-x)^{n-k} \\ &\qquad \leq 2\|f\|_\infty \sum_{|k/n-x|\geq\delta} \binom{n}{k} x^k (1-x)^{n-k} \\ &\qquad \leq \frac{2\|f\|_\infty}{\delta^2} \sum_{|k/n-x|\geq\delta} \left(\frac{k}{n} - x\right)^2 \binom{n}{k} x^k (1-x)^{n-k} \\ &\qquad \leq \frac{2\|f\|_\infty}{\delta^2} \sum_{k=0}^{n} \left(\frac{k}{n} - x\right)^2 \binom{n}{k} x^k (1-x)^{n-k} \\ &\qquad = \frac{2\|f\|_\infty}{n\delta^2} x(1-x) \leq \frac{\|f\|_\infty}{2n\delta^2}. \end{aligned} \tag{2.31}$$

Therefore, there is N such that for all $n \geq N$ the second sum in (2.30) is bounded by $\epsilon/2$ and this completes the proof.

□

2.3 Best Approximation

We just saw that any continuous function f on a closed 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 linear space V and a subspace W of V, $p^* \in W$ is called the best approximation of $f \in V$ by elements in W if*

$$\|f - p^*\| \leq \|f - p\|, \quad \text{for all } p \in W. \tag{2.32}$$

For example, the normed linear 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 linear 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

$$\|f - p\| \leq \|f - 0\| = \|f\|. \tag{2.33}$$

Therefore we can restrict our search to the set

$$F = \{p \in W : \|f - p\| \leq \|f\|\}. \tag{2.34}$$

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. □

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 of subspace of V). Take $f(x) = 1/(1-x)$. Then, given $\epsilon > 0$ there is N such that*

$$\max_{x\in[0,1/2]} \left| \frac{1}{1-x} - (1 + x + x^2 + \ldots + x^N) \right| < \epsilon. \tag{2.35}$$

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

$$p^*(x) = \frac{1}{1-x}, \tag{2.36}$$

which is impossible.

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.

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

$$\|\theta(f - p^*) + (1-\theta)(f - q^*)\| < M, \quad \text{for all } 0 < \theta < 1. \tag{2.37}$$

Taking $\theta = 1/2$ we get

$$\|f - \frac{1}{2}(p^* + q^*)\| < M, \tag{2.38}$$

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 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 linear space $\mathbb{P}_n$, which is that the only element of $\mathbb{P}_n$ that has more than n roots is the zero element, it is possible to prove that the best approximation is unique.

The crux of the matter is that error function

$$e_n(x) = f(x) - p_n^*(x), \quad x \in [a, b], \tag{2.39}$$

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

$$\begin{aligned} e_n(x_1) &= \pm\|e_n\|_\infty \\ e_n(x_2) &= -e_n(x_1), \\ e_n(x_3) &= -e_n(x_2), \\ &\vdots \\ e_n(x_k) &= -e_n(x_{k-1}), \end{aligned} \tag{2.40}$$

for if not, it would be possible to find a polynomial of degree at most n, with the same sign at the extrema 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 extrema 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 take c as one half the gap between the minimum m of e_n and $-\|e_n\|_\infty$,

$$c = \frac{1}{2}\left(m + \|e_n\|_\infty\right) > 0, \tag{2.41}$$

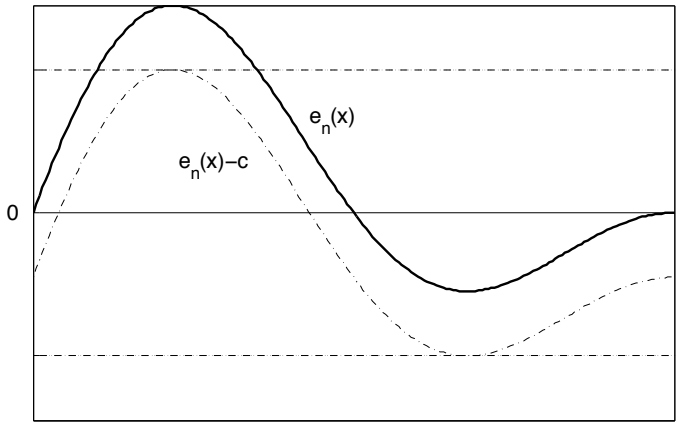

Figure 2.3: If the error function e_n does not equioscillate at least twice we could lower $\|e_n\|_\infty$ by an amount $c > 0$.

and subtract it to e_n, as shown in Fig. 2.3, we have

$$-\|e_n\|_\infty + c \leq e_n(x) - c \leq \|e_n\|_\infty - c. \tag{2.42}$$

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 used when $e_n(x_1) = -\|e_n\|_\infty$. □

Before proceeding to the general case, let us look at the $n = 1$ situation. Suppose there are only two alternating extrema x_1 and x_2 for e_1 as described in (2.40). 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

$$e_1(x) > \frac{\|e_1\|_\infty}{2} \quad \text{for all } x \in I_1, \tag{2.43}$$

$$e_1(x) < -\frac{\|e_1\|_\infty}{2} \quad \text{for all } x \in I_2. \tag{2.44}$$

Clearly I_1 and I_2 are disjoint sets so we can choose a point x_0 between the two intervals. Then, it is possible to find a linear polynomial q that passes through x_0 and that is positive in I_1 and negative in I_2. We are now going

to find 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

$$0 < \alpha < \frac{1}{2\|q\|_\infty} \left(\|e_1\|_\infty - d\right). \tag{2.45}$$

On I_1, we have

$$0 < \alpha q(x) < \frac{1}{2\|q\|_\infty} \left(\|e_1\|_\infty - d\right) q(x) \leq \frac{1}{2} \left(\|e_1\|_\infty - d\right) < e_1(x). \tag{2.46}$$

Therefore

$$|e_1(x) - \alpha q(x)| = e_1(x) - \alpha q(x) < \|e_1\|_\infty, \quad \text{for all } x \in I_1. \tag{2.47}$$

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

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

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 (2.40).*

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

$$\|f - q_n\|_\infty < \|f - p_n^*\|_\infty. \tag{2.49}$$

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

$$|f(x_j) - q_n(x_j)| < |f(x_j) - p_n^*(x_j)|, \quad j = 1, \ldots, k \tag{2.50}$$

and since

$$f(x_j) - p_n^*(x_j) = -[f(x_{j+1}) - p_n^*(x_{j+1})], \quad j = 1, \ldots, k-1 \tag{2.51}$$

we have that

$$q_n(x_j) - p_n^*(x_j) = f(x_j) - p_n^*(x_j) - [f(x_j) - q_n(x_j)] \tag{2.52}$$

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 (2.49), 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

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

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

$$q(x) = (t_1 - x)(t_2 - x) \cdots (t_m - x) \tag{2.53}$$

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^*. □

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

$$\begin{aligned} \|f - r\|_\infty &= \|\frac{1}{2}(f - p_n^*) + \frac{1}{2}(f - q_n^*)\|_\infty \\ &\leq \frac{1}{2}\|f - p_n^*\|_\infty + \frac{1}{2}\|f - q_n^*\|_\infty = \|e_n\|_\infty. \end{aligned} \tag{2.54}$$

Let $x_1, \ldots, x_{n+2}$ be extremal points of $f - r$ with the alternating property (2.40), 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

$$\frac{f(x_j) - p_n^*(x_j)}{2} + \frac{f(x_j) - q_n^*(x_j)}{2} = (-1)^{m+j} \|e_n\|_\infty, \quad j = 1, \ldots, n+2. \tag{2.55}$$

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,

$$f(x_j) - p_n^*(x_j) = f(x_j) - q_n^*(x_j) = (-1)^{m+j} \|e_n\|_\infty, \quad j = 1, \ldots, n+2, \tag{2.56}$$

and it follows that

$$p_n^*(x_j) = q_n^*(x_j), \quad j = 1, \ldots, n+2 \tag{2.57}$$

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.

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

$$-1 \leq x_1 < x_2 < \cdots < x_{n+2} \leq 1, \tag{2.58}$$

such that

$$e_n^2(x_j) = \|e_n\|_\infty^2, \quad \text{for } j = 1, \ldots, n+2. \tag{2.59}$$

Now consider the polynomial

$$q(x) = \|e_n\|_\infty^2 - e_n^2(x). \tag{2.60}$$

Then, $q(x_j) = 0$ for $j = 1, \ldots, n+2$. Each of 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

$$(1-x^2)[e_n'(x)]^2 = (n+1)^2 \left[\|e_n\|_\infty^2 - e_n^2(x)\right]. \tag{2.61}$$

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 (2.61) we get

$$\frac{e_n'(x)}{\sqrt{\|e_n\|_\infty^2 - e_n^2(x)}} = \frac{n+1}{\sqrt{1-x^2}}, \quad \text{for } x \in [-1, x_2]. \tag{2.62}$$

Using the trigonometric substitution $x = \cos\theta$, we can integrate to obtain

$$e_n(x) = \|e_n\|_\infty \cos[(n+1)\theta], \tag{2.63}$$

for $x = \cos\theta \in [-1, x_2]$ with $0 < \theta \leq \pi$, where we have chosen the constant of integration to be zero so that $e_n(1) = \|e_n\|_\infty$. 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 $[-1, x_2]$, (2.63) 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*

$$T_n(x) = \cos n\theta, \quad x = \cos\theta, \quad 0 \le \theta \le \pi. \tag{2.64}$$

Note that (2.64) 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

$$\cos[(n+1)\theta] + \cos[(n-1)\theta] = 2\cos n\theta \cos\theta, \tag{2.65}$$

we immediately get

$$T_{n+1}(\cos\theta) + T_{n-1}(\cos\theta) = 2T_n(\cos\theta) \cdot \cos\theta \tag{2.66}$$

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

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x), \quad n \ge 1, \end{aligned} \tag{2.67}$$

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.

$$\begin{aligned} &T_0(x) = 1, \\ &T_1(x) = x, \\ &T_2(x) = 2x \cdot x - 1 = 2x^2 - 1, \\ &T_3(x) = 2x \cdot (2x^2 - 1) - x = 4x^3 - 3x, \\ &T_4(x) = 2x(4x^3 - 3x) - (2x^2 - 1) = 8x^4 - 8x^2 + 1 \\ &T_5(x) = 2x(8x^4 - 8x^2 + 1) - (4x^3 - 3x) = 16x^5 - 20x^3 + 5x. \end{aligned} \tag{2.68}$$

From these few Chebyshev polynomials, and from (2.67), we see that

$$T_n(x) = 2^{n-1}x^n + \text{lower order terms} \tag{2.69}$$

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

$$T_n(-x) = (-1)^n T_n(x). \tag{2.70}$$

Going back to (2.63), 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

$$p_n^*(x) = x^{n+1} - \frac{1}{2^n} T_{n+1}(x) \tag{2.71}$$

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 infinity norm in $[-1, 1]$ is

$$\tilde{T}_n(x) = \frac{1}{2^{n-1}} T_n(x). \tag{2.72}$$

Hence, for any other monic polynomial p of degree n

$$\max_{x\in[-1,1]} |p(x)| > \frac{1}{2^{n-1}}. \tag{2.73}$$

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

$$\bar{x}_j = \cos\left(\frac{(2j+1)}{n}\frac{\pi}{2}\right) \quad j = 0, \ldots, n-1. \tag{2.74}$$

The extrema 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

$$x_j = \cos\left(\frac{j\pi}{n}\right), \quad j = 0, 1, \ldots, n. \tag{2.75}$$

These points are called Chebyshev or Gauss-Lobatto points and are extremely useful in applications. Note that x_j for $j = 1, \ldots, n-1$ are local extrema. Therefore

$$T_n'(x_j) = 0, \quad \text{for } j = 1, \ldots, n-1. \tag{2.76}$$

In other words, the Chebyshev points (2.75) 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 we get

$$T_n'(x) = -n \sin n\theta \frac{d\theta}{dx} = n\frac{\sin n\theta}{\sin\theta}, \quad (x = \cos\theta). \tag{2.77}$$

Therefore

$$\frac{T'_{n+1}(x)}{n+1} - \frac{T'_{n-1}(x)}{n-1} = \frac{1}{\sin\theta}\left[\sin(n+1)\theta - \sin(n-1)\theta\right] \tag{2.78}$$

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

$$\frac{T'_{n+1}(x)}{n+1} - \frac{T'_{n-1}(x)}{n-1} = 2T_n(x). \tag{2.79}$$

The polynomial

$$U_n(x) = \frac{T'_{n+1}(x)}{n+1} = \frac{\sin(n+1)\theta}{\sin\theta}, \quad (x = \cos\theta) \tag{2.80}$$

of degree n is called the Chebyshev polynomial of second kind. Thus, the Chebyshev nodes (2.75) are the zeros of the polynomial

$$q_{n+1}(x) = (1-x^2)U_{n-1}(x). \tag{2.81}$$

Chapter 3

Interpolation

3.1 Polynomial Interpolation

One of the basic tools for approximating a function or a given data set is interpolation. In this chapter we focus on polynomial and piece-wise 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 property:

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

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 can come from a function f we are trying to approximate, in which case $f_j = f(x_j)$ for $j = 0, 1, \ldots, n$.

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

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

$$a_0 + a_1 x_1 + \cdots + a_n x_1^n = f_1,$$

$$\vdots$$

$$a_0 + a_1 x_n + \cdots + a_n x_n^n = f_n.$$

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 linear polynomial, p_1. Suppose we are given (x_0, f_0) and (x_1, f_1). We have written p_1 explicitly in the Introduction, we write it now in a different form:*

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

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

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

Example 3.2. *Given (x_0, f_0), (x_1, f_1), and (x_2, f_2) let us construct $p_2 \in \mathbb{P}_2$ that interpolates these points. The way we have written p_1 in (3.2) is suggestive of how to explicitly 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 interpolation property*

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

We can now write down the polynomial 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 are called the *elementary Lagrange polynomials* of degree n. For simplicity, we are omitting in the notation their dependence on the $n+1$ nodes. Since $l_j(x_k) = \delta_{jk}$, we have that

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

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

interpolates the given data, i.e., it satisfies the interpolation property $p_n(x_j) = f_j$ 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. For convenience we are going to take the interval $[-1, 1]$. For a general interval $[a, b]$, we can do the simple change of variables

$$x = \frac{1}{2}(a+b) + \frac{1}{2}(b-a)t, \quad t \in [-1, 1]. \tag{3.12}$$

The uniform or equispaced nodes are given by

$$x_j = -1 + jh, \quad j = 0, 1, \ldots, n \text{ and } h = 2/n. \tag{3.13}$$

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* or *Gauss-Lobatto* set

$$x_j = \cos\left(\frac{j\pi}{n}\right), \qquad j = 0, \ldots, n, \tag{3.14}$$

which, as discussed in Section 2.4, are the extrema of the Chebyshev polynomial (2.64) 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 (3.14), the nodes go from 1 to -1 so often 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.

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 (2.40), 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. Of course, we do not construct p_n^* by finding these particular $n+1$ interpolation nodes. A more practical question is: given $(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))$, where $x_0, \ldots, x_n$ are distinct interpolation nodes in $[a, b]$, how close is p_n, the interpolating polynomial of degree at most n of f at the given nodes, to the best uniform approximation p_n^* of f in $\mathbb{P}_n$?

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

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

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

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

Using (3.17) we obtain

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

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:

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

where $a \le x_0^{(n)} < x_1^{(n)} < \cdots < x_n^{(n)} \le b$ 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 $\|f - p_n\|_\infty$ as $n \to \infty$, i.e. that the interpolating polynomial, as we add more and more nodes, converges uniformly to f. That depends on the regularity of f and on the distribution of the nodes. In fact, if we are given the triangular array of interpolation nodes (3.21) in advance, it is possible to construct a continuous function f such that p_n will not converge uniformly to f as $n \to \infty$.

3.3 Barycentric Formula

The Lagrange form of the interpolating polynomial is not convenient for computations. 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 very 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 elementary Lagrange polynomials may be written as

$$l_j(x) = \frac{\dfrac{\omega(x)}{x - x_j}}{\omega'(x_j)} = \frac{\omega(x)}{(x - x_j)\omega'(x_j)}, \quad \text{for } j = 0, 1, \ldots, n, \tag{3.26}$$

for $x \neq x_j$ 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 write Lagrange formula for the interpolating polynomial of f at $x_0, x_1, \ldots, x_n$ as

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

Now, note 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:

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

For $x = x_j$, $j = 0, 1, \ldots, n$, 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 (3.14) and for the equally spaced nodes (3.13) 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 (3.12) 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

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

3.3.3 Barycentric Weights for General Sets of Nodes

For general arrays of nodes we can precompute the barycentric weights efficiently as follows.

$\lambda_0^{(0)} = 1;$
for $m = 1 : n$
  for $j = 0 : m-1$
    $\lambda_j^{(m)} = \frac{\lambda_j^{(m-1)}}{x_j - x_m};$
  end
  $\lambda_m^{(m)} = \frac{1}{\prod_{k=0}^{m-1}(x_m - x_k)};$
end

If we want to add one more point (x_{n+1}, f_{n+1}) we just extend the m-loop to $n+1$ to generate $\lambda_0^{(n+1)}, \lambda_1^{(n+1)}, \cdots, \lambda_{n+1}^{(n+1)}$.

3.4 Newton's Form and Divided Differences

There is another representation of the interpolating polynomial which is both very efficient computationally and very convenient in the derivation of numerical methods based on interpolation. 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 $j = 0, \ldots, n-1$

$$r(x_j) = p_n(x_j) - p_{n-1}(x_j) = f_j - f_j = 0. \tag{3.36}$$

Therefore, r can be factored as

$$r(x) = c_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}). \tag{3.37}$$

The constant c_n is called the *n-th divided difference* of $f = [f_0, f_1, \ldots, f_n]$ with respect to $x_0, x_1, ..., x_n$, and is usually denoted as $f[x_0, \ldots, x_n]$. Thus, we have

$$p_n(x) = p_{n-1}(x) + f[x_0, \ldots, x_n](x - x_0)(x - x_1)\cdots(x - x_{n-1}). \tag{3.38}$$

By the same argument, we have

$$p_{n-1}(x) = p_{n-2}(x) + f[x_0, \ldots, x_{n-1}](x - x_0)(x - x_1)\cdots(x - x_{n-2}), \quad (3.39)$$

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

$$p_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + \ldots + f[x_0, \ldots, x_n](x - x_0)\cdots(x - x_{n-1}). \quad (3.40)$$

Note that for $n = 1$

$$\begin{aligned} p_1(x) &= f[x_0] + f[x_0, x_1](x - x_0), \\ p_1(x_0) &= f[x_0] = f_0, \\ p_1(x_1) &= f[x_0] + f[x_0, x_1](x_1 - x_0) = f_1. \end{aligned}$$

Therefore

$$f[x_0] = f_0, \quad (3.41)$$

$$f[x_0, x_1] = \frac{f_1 - f_0}{x_1 - x_0}, \quad (3.42)$$

and

$$p_1(x) = f_0 + \frac{f_1 - f_0}{x_1 - x_0}(x - x_0). \quad (3.43)$$

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

Theorem 3.2.

$$f[x_0, x_1, ..., x_k] = \frac{f[x_1, x_2, ..., x_k] - f[x_0, x_1, ..., x_{k-1}]}{x_k - x_0}. \quad (3.44)$$

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

$$p(x) = p_{k-1}(x) + \frac{x - x_k}{x_k - x_0}[p_{k-1}(x) - q_{k-1}(x)]. \quad (3.45)$$

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

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

Moreover, $p(x_0) = q_{k-1}(x_0) = f_0$ and $p(x_k) = p_{k-1}(x_k) = f_k$. Therefore, p is the interpolation polynomial of degree at most k of the points $(x_0, f_0), (x_1, f_1), \ldots, (x_k, f_k)$. The leading order coefficient of p_k is $f[x_0, ..., x_k]$ and 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 (3.44). □

To obtain the divided difference we construct a table using (3.44), column by column as illustrated below for $n = 3$.

x_j	0th order	1th order	2th order	3th order
x_0	f_0			
		$f[x_0, x_1]$		
x_1	f_1		$f[x_0, x_1, x_2]$	
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$
x_2	f_2		$f[x_1, x_2, x_3]$	
		$f[x_2, x_3]$		
x_3	f_3			

Example 3.3. *Let $f(x) = 1 + x^2$, $x_j = j$, and $f_j = f(x_j)$ for $j = 0, \ldots, 3$. Then*

x_j	*0th order*	*1th order*	*2th order*	*3th order*
0	(*1*)			
		$\frac{2-1}{1-0} =$ (*1*)		
1	2		$\frac{3-1}{2-0} =$ (*1*)	
		$\frac{5-2}{2-1} = 3$		(*0*)
2	5		$\frac{5-3}{3-1} = 1$	
		$\frac{10-5}{3-2} = 5$		
3	10			

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 following *Horner*-like scheme to get $p = p_n(x)$:
$p = c_n$;
for $k = n - 1 : 0$
  $p = c_k + (x - x_k) * p$;
end

3.5 Cauchy Remainder

We now assume that 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 the Introduction 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 $f \in C^{n+1}[a, b]$, $x_0, x_1, ..., x_n, x$ be contained in $[a, b]$, and p_n be the interpolation polynomial of degree at most n of f at $x_0, ..., x_n$ then*

$$f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi(x))(x - x_0)(x - x_1) \cdots (x - x_n), \quad (3.46)$$

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

Proof. The right hand side of (3.46) is known as the Cauchy Remainder and the following proof is due to Cauchy.

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

$$\phi(t) = f(t) - p_n(t) - [f(x) - p_n(x)]\frac{(t-x_0)(t-x_1)\cdots(t-x_n)}{(x-x_0)(x-x_1)\cdots(x-x_n)}. \quad (3.47)$$

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 $n+2$ zeros. Applying Rolle's Theorem $n+1$ times we conclude that there exists a point $\xi(x) \in (a,b)$ such that $\phi^{(n+1)}(\xi(x)) = 0$. 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 (3.46) 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 $\max\{x_0, x_1, ..., x_n, x\}$. □

We are now going to find a beautiful connection between Chebyshev polynomials and the interpolation error as given by the Cauchy remainder (3.46). Let us consider the interval $[-1,1]$. We have no control on the term $f^{(n+1)}(\xi(x))$. However, we can choose the interpolation nodes $x_0, \ldots, x_n$ so that the factor

$$w(x) = (x-x_0)(x-x_1)\cdots(x-x_n) \quad (3.48)$$

is smallest as possible in the infinity norm. The function w is a monic polynomial of degree $n+1$ and we have proved in Section 2.4 that the Chebyshev polynomial $\widetilde{T}_{n+1}$, defined in (2.72), is the monic polynomial of degree $n+1$ with smallest infinity norm. Hence, if the interpolation nodes are taken to be the zeros of $\widetilde{T}_{n+1}$, namely

$$x_j = \cos\left(\frac{(2j+1)}{n+1}\frac{\pi}{2}\right), \quad j = 0, 1, \ldots n. \quad (3.49)$$

$\|w\|_\infty$ is minimized and $\|w\|_\infty = 2^{-n}$. The following theorem summarizes this observation.

Theorem 3.4. *Let p_n^T be the interpolating polynomial of degree at most n of $f \in C^{n+1}[-1,1]$ with respect to the nodes (3.49) then*

$$\|f - p_n^T\|_\infty \leq \frac{1}{2^n(n+1)!}\|f^{n+1}\|_\infty. \quad (3.50)$$

The Gauss-Lobatto Chebyshev points,

$$x_j = \cos\left(\frac{j\pi}{n}\right), \quad j = 0, 1, \ldots, n, \tag{3.51}$$

which are the extrema 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 Gauss-Lobatto nodes are the zeros of the (monic) polynomial [see (2.81) and (3.31)]

$$\frac{1}{2^{n-1}}(1-x^2)U_{n-1}(x) = \frac{1}{2^{n-1}}\sin\theta\sin n\theta, \quad x = \cos\theta. \tag{3.52}$$

We have that

$$\|w\|_\infty = \max_{x\in[-1,1]} \left|\frac{1}{2^{n-1}}(1-x^2)U_{n-1}(x)\right| \leq \frac{1}{2^{n-1}}. \tag{3.53}$$

So, the Gauss-Lobatto nodes yield a $\|w\|_\infty$ of no more than a factor of two from the optimal value.

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 (3.38) implies

$$p_{n+1}(x) = p_n(x) + f[x_0, \ldots, x_n, t](x-x_0)(x-x_1)\cdots(x-x_n). \tag{3.54}$$

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

$$f(t) = p_n(t) + f[x_0, \ldots, x_n, t](t-x_0)(t-x_1)\cdots(t-x_n). \tag{3.55}$$

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

$$f(x) = p_n(x) + f[x_0, \ldots, x_n, x](x-x_0)(x-x_1)\cdots(x-x_n), \tag{3.56}$$

and upon comparing with the Cauchy remainder we get

$$f[x_0, ..., x_n, x] = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}. \tag{3.57}$$

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

$$f[x_0, ..., x_k] = \frac{1}{k!}f^{(k)}(\xi), \tag{3.58}$$

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

$$\lim_{x_1,\ldots,x_k \to x_0} f[x_0, ..., x_k] = \frac{1}{k!} f^{(k)}(x_0). \tag{3.59}$$

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 the following interpolation problem.

3.6 Hermite Interpolation

The Hermite interpolation problem is: 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 the Newton's form representation.

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 (3.59), is

$$\begin{array}{c|cccc} x_0 & f(x_0) & & & \\ x_0 & f(x_0) & f'(x_0) & & \\ x_1 & f(x_1) & f[x_0, x_1] & f[x_0, x_0, x_1] & \\ x_1 & f(x_1) & f'(x_1) & f[x_0, x_1, x_1] & f[x_0, x_0, x_1, x_1] \end{array} \tag{3.60}$$

and

$$\begin{aligned} p(x) = f(x_0) + f'(x_0)(x - x_0) + f[x_0, x_0, x_1](x - x_0)^2 \\ + f[x_0, x_0, x_1, x_1](x - x_0)^2(x - x_1). \end{aligned} \tag{3.61}$$

Example 3.4. *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:

$$\begin{array}{c|ccc} 0 & \textit{1} & & \\ 0 & 1 & \textit{0} & \\ & & & \\ 1 & \sqrt{2} & \sqrt{2}-1 & \sqrt{2}-1 \end{array} \tag{3.62}$$

and therefore

$$p(x) = 1 + 0(x-0) + (\sqrt{2}-1)(x-0)^2 = 1 + (\sqrt{2}-1)x^2. \tag{3.63}$$

3.7 Convergence of Polynomial Interpolation

From the Cauchy Remainder formula

$$f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi(x))(x-x_0)(x-x_1)\cdots(x-x_n) \tag{3.64}$$

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

In the Runge example

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

is very smooth. 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 (3.13) 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. On the other hand, there is fast and uniform convergence of p_n to f when the Chebyshev nodes (3.14) are employed.

It is then natural to ask: Given any $f \in C[a,b]$, can we guarantee that if we choose the Chebyshev nodes $\|f - p_n\|_\infty \to 0$? The answer is no. Bernstein and Faber proved in 1914 that 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 $n \to \infty$. However, if f is slightly smoother, for example $f \in C^1[a,b]$, then for the Chebyshev array of nodes $\|f - p_n\|_\infty \to 0$.

If $f(x) = e^{-x^2}$ and there is convergence of p_n even with the equidistributed nodes. What is so special about this function? The function

$$f(z) = e^{-z^2}, \tag{3.65}$$

$z = x + iy$ is analytic in the entire complex plane. Using complex variables analysis it can be shown that if f is analytic in a sufficiently large region in the complex plane containing $[a,b]$ 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 $n \to \infty$.

In the limit as $n \to \infty$, we can think of the nodes as a continuum with a density ρ so that for sufficiently large n,

$$(n+1)\int_a^x \rho(t)dt \tag{3.66}$$

is the total number of nodes in $[a,x]$. Take for example, $[-1,1]$. For equispaced nodes $\rho(x) = 1/2$ and for the Chebyshev nodes $\rho(x) = 1/(\pi\sqrt{1-x^2})$.

It turns out that the relevant domain of analyticity is given in terms of the function

$$\phi(z) = -\int_a^b \rho(t)\ln|z-t|dt. \tag{3.67}$$

Let Γ_c be the level curve consisting of all the points $z \in \mathbb{C}$ such that $\phi(z) = c$ for c constant. For very large and negative c, Γ_c approximates a large circle. As c is increased, Γ_c shrinks. We take the "smallest" level curve, Γ_{c_0}, which contains $[a,b]$. The relevant domain of analyticity is

$$R_{c_0} = \{z \in \mathbb{C} : \phi(z) \geq c_0\}. \tag{3.68}$$

Then, if f is analytic in R_{c_0}, $\|f - p_n\|_\infty \to 0$, not only in $[a,b]$ but for every point in R_{c_0}. Moreover,

$$|f(x) - p_n(x)| \leq Ce^{-N(\phi(x)-c_0)}, \tag{3.69}$$

for some constant C. That is p_n **converges exponentially fast to** f. For the Chebyshev nodes R_{c_0} approximates $[a, b]$, so if f is analytic in any region containing $[a, b]$, however thin this region might be, p_n will converge uniformly to f. For equidistributed nodes, R_{c_0} looks like a football, with $[a, b]$ as its longest axis. In the Runge example, the function is singular at $z = \pm i/5$ which happens to be inside this football-like domain and this explain the observed lack of convergence for this particular function.

The moral of the story is that polynomial interpolation using *Chebyshev* nodes converges very rapidly for smooth functions and thus yields very accurate approximations.

3.8 Piece-wise Linear Interpolation

One way to reduce the error in linear interpolation is to divide $[a, b]$ into small subintervals $[x_0, x_1], ..., [x_{n-1}, x_n]$. In each of the subintervals $[x_j, x_{j+1}]$ we approximate f by

$$p(x) = f(x_j) + \frac{f(x_{j+1}) - f(x_j)}{x_{j+1} - x_j}(x - x_j), \qquad x \in [x_j, x_{j+1}]. \tag{3.70}$$

We know that

$$f(x) - p(x) = \frac{1}{2}f''(\xi)(x - x_j)(x - x_{j+1}), \qquad x \in [x_j, x_{j+1}] \tag{3.71}$$

where ξ is some point between x_j and x_{j+1}.

Suppose that $|f''(x)| \leq M_2, \forall x \in [a, b]$ then

$$|f(x) - p(x)| \leq \frac{1}{2}M_2 \max_{x_j \leq x \leq x_{j+1}} |(x - x_j)(x - x_{j+1})|. \tag{3.72}$$

Now the max at the right hand side is attained at the midpoint $(x_j + x_{j+1})/2$ and

$$\max_{x_j \leq x \leq x_{j+1}} |(x - x_j)(x - x_{j+1})| = \left(\frac{x_{j+1} - x_j}{2}\right)^2 = \frac{1}{4}h_j^2, \tag{3.73}$$

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

$$\max_{x_j \leq x \leq x_{j+1}} |f(x) - p(x)| \leq \frac{1}{8}M_2 h_j^2. \tag{3.74}$$

If we want this error to be smaller than a prescribed tolerance δ we can take sufficiently small subintervals. Namely, we can pick h_j such that $\frac{1}{8}M_2h_j^2 \leq \delta$ which implies that

$$h_j \leq \sqrt{\frac{8\delta}{M_2}}. \tag{3.75}$$

3.9 Cubic Splines

Several applications require a smoother curve than that provided by a piecewise linear approximation. Continuity of the first and second derivatives provide that required smoothness.

One of the most frequently used such approximations is a cubic spline, which is is a piecewise cubic function, s, which interpolates a set points $(x_0, f_0), (x_1, f_1), \ldots (x_n, f_n)$, and has two continuous derivatives. In each subinterval $[x_j, x_{j+1}]$, s is a cubic polynomial, which we may represent as

$$s_j(x) = A_j(x - x_j)^3 + B_j(x - x_j)^2 + C_j(x - x_j) + D_j. \tag{3.76}$$

Let

$$h_j = x_{j+1} - x_j. \tag{3.77}$$

The spline $s(x)$ interpolates the given data:

$$s_j(x_j) = f_j = D_j, \tag{3.78}$$
$$s_j(x_{j+1}) = A_jh_j^3 + B_jh_j^2 + C_jh_j + D_j = f_{j+1}. \tag{3.79}$$

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

$$s_j'(x_j) = C_j, \tag{3.80}$$
$$s_j'(x_{j+1}) = 3A_jh_j^2 + 2B_jh_j + C_j, \tag{3.81}$$
$$s_j''(x_j) = 2B_j, \tag{3.82}$$
$$s_j''(x_{j+1}) = 6A_jh_j + 2B_j. \tag{3.83}$$

We are going to write the spline coefficients A_j, B_j, C_j, and D_j in terms of f_j and f_{j+1} and the unknown values $z_j = s_j''(x_j)$ and $z_{j+1} = s_j''(x_{j+1})$. We

have

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

and substituting these values in (3.79) we get

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

Let us collect all our formulas for the spline coefficients:

$$A_j = \frac{1}{6h_j}(z_{j+1} - z_j), \tag{3.84}$$

$$B_j = \frac{1}{2} z_j, \tag{3.85}$$

$$C_j = \frac{1}{h_j}(f_{j+1} - f_j) - \frac{1}{6} h_j (z_{j+1} + 2z_j), \tag{3.86}$$

$$D_j = f_j. \tag{3.87}$$

Note that the second derivative of s is continuous, $s_j''(x_{j+1}) = z_{j+1} = s_{j+1}''(x_{j+1})$, and by construction s interpolates the given data. We are now going to use the condition of continuity of the first derivative of s to determine equations for the unknown values z_j, $j = 1, 2, \ldots, n-1$:

$$\begin{aligned} s_j'(x_{j+1}) &= 3A_j h_j^2 + 2B_j h_j + C_j \\ &= 3\frac{1}{6h_j}(z_{j+1} - z_j)h_j^2 + 2\frac{1}{2} z_j h_j + \frac{1}{h_j}(f_{j+1} - f_j) - \frac{1}{6} h_j (z_{j+1} + 2z_j) \\ &= \frac{1}{h_j}(f_{j+1} - f_j) + \frac{1}{6} h_j (2z_{j+1} + z_j). \end{aligned}$$

Decreasing the index by 1 we get

$$s_{j-1}'(x_j) = \frac{1}{h_{j-1}}(f_j - f_{j-1}) + \frac{1}{6} h_{j-1}(2z_j + z_{j-1}). \tag{3.88}$$

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

$$\frac{1}{h_{j-1}}(f_j - f_{j-1}) + \frac{1}{6}h_{j-1}(2z_j + z_{j-1}) = C_j = \frac{1}{h_j}(f_{j+1} - f_j) - \frac{1}{6}h_j(z_{j+1} + 2z_j)$$

which can be written as

$$\begin{aligned} h_{j-1}z_{j-1} + 2(h_{j-1} + h_j)z_j + h_j z_{j+1} = \\ -\frac{6}{h_{j-1}}(f_j - f_{j-1}) + \frac{6}{h_j}(f_{j+1} - f_j), \quad j = 1, \ldots, n-1. \end{aligned} \tag{3.89}$$

This is a linear system of $n-1$ equations for the $n-1$ unknowns $z_1, z_2, \ldots, z_{n-1}$. In matrix form

$$\begin{bmatrix} 2(h_0+h_1) & h_1 & \cdots & 0 \\ h_1 & 2(h_1+h_2) & h_2 \ddots & 0 \\ \vdots & \ddots & \ddots & h_{n-2} \\ 0 & \ddots & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-1} \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \end{bmatrix}, \tag{3.90}$$

where

$$\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-2} \\ d_{n-1} \end{bmatrix} = \begin{bmatrix} -\frac{6}{h_0}(f_1 - f_0) + \frac{6}{h_1}(f_2 - f_1) - h_0 z_0 \\ -\frac{6}{h_1}(f_2 - f_1) + \frac{6}{h_2}(f_3 - f_2) \\ \vdots \\ -\frac{6}{h_{n-3}}(f_{n-2} - f_{n-3}) + \frac{6}{h_{n-2}}(f_{n-1} - f_{n-2}) \\ -\frac{6}{h_{n-2}}(f_{n-1} - f_{n-2}) + \frac{6}{h_{n-1}}(f_n - f_{n-1}) - h_{n-1}z_n \end{bmatrix}. \tag{3.91}$$

Note that $z_0 = f''_0$ and $z_n = f''_n$ are unspecified. The values $z_0 = z_n = 0$ defined what is called a *Natural Spline*. The matrix of the linear system (3.90) is strictly diagonally dominant, a concept we make precise in the definition below. A consequence of this property, as we will see shortly, is that the matrix is nonsingular and therefore there is a unique solution for $z_1, z_2, \ldots, z_{n-1}$ corresponding the second derivative values of the spline at the interior nodes.

Definition 3.1. *An $n \times n$ matrix A with entries a_{ij}, $i, j = 1, \ldots, n$ is strictly diagonally dominant if*

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \quad \textit{for } i = 1, \ldots, n. \tag{3.92}$$

Theorem 3.5. *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

$$a_{kk}x_k + \sum_{\substack{j=1 \\ j\neq k}}^{n} a_{kj}x_j = 0 \tag{3.93}$$

and consequently

$$|a_{kk}||x_k| \leq \sum_{\substack{j=1 \\ j\neq k}}^{n} |a_{kj}||x_j|. \tag{3.94}$$

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

$$|a_{kk}| \leq \sum_{\substack{j=1 \\ j\neq k}}^{n} |a_{kj}|, \tag{3.95}$$

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

If the nodes are equidistributed, $x_j = x_0 + jh$ for $j = 0, 1, \ldots, n$ then the linear system (3.90), after dividing by h simplifies to

$$\begin{bmatrix} 4 & 1 & \cdots & 0 \\ 1 & 4 & 1 \ddots & 0 \\ \vdots & 1 & \ddots & 1 \\ 0 & \ddots & 1 & 4 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-2} \\ z_{n-1} \end{bmatrix} = \begin{bmatrix} \frac{6}{h^2}(f_0 - 2f_1 + f_2) - z_0 \\ \frac{6}{h^2}(f_1 - 2f_2 + f_3) \\ \vdots \\ \frac{6}{h^2}(f_{n-3} - 2f_{n-2} + f_{n-1}) \\ \frac{6}{h^2}(f_{n-2} - 2f_{n-1} + f_n) - z_n \end{bmatrix}. \tag{3.96}$$

Once the $z_1, z_2, \ldots, z_{n-1}$ are found the spline coefficients can be computed from (3.84)-(3.87).

Example 3.5. *Find the natural cubic spline that interpolates* $(0, 0), (1, 1), (2, 16)$. *We know* $z_0 = 0$ *and* $z_2 = 0$. *We only need to find* z_1 *(only 1 interior node). The system (3.89) degenerates to just one equation and* $h = 1$, *thus*

$$z_0 + 4z_1 + z_2 = 6[f_0 - 2f_1 + f_2] \Rightarrow z_1 = 21$$

In $[0,1]$ *we have*

$$\begin{aligned}
A_0 &= \frac{1}{6h}(z_1 - z_0) = \frac{1}{6} \times 21 = \frac{7}{2}, \\
B_0 &= \frac{1}{2}z_0 = 0 \\
C_0 &= \frac{1}{h}(f_1 - f_0) - \frac{1}{6}h(z_1 + 2z_0) = 1 - \frac{1}{6}21 = -\frac{5}{2}, \\
D_0 &= f_0 = 0.
\end{aligned}$$

Thus, $s_0(x) = A_0(x-0)^3 + B_0(x-0)^3 + C_0(x-0) + D_0 = \frac{7}{2}x^3 - \frac{5}{2}x$. *Now in* $[1,2]$

$$\begin{aligned}
A_1 &= \frac{1}{6h}(z_2 - z_1) = \frac{1}{6}(-21) = -\frac{7}{2}, \\
B_1 &= \frac{1}{2}z_1 = \frac{21}{2}, \\
C_1 &= \frac{1}{h}(f_2 - f_1) - \frac{1}{6}h(z_2 + 2z_1) = 16 - 1 - \frac{1}{6}(2 \cdot 21) = 8, \\
D_1 &= f_1 = 1.
\end{aligned}$$

and $s_1(x) = -\frac{7}{2}(x-1)^3 + \frac{21}{2}(x-1)^2 + 8(x-1) + 1$. *Therefore the spline is given by*

$$s(x) = \begin{cases} \frac{7}{2}x^3 - \frac{5}{2}x & x \in [0,1], \\ -\frac{7}{2}(x-1)^3 + \frac{21}{2}(x-1)^2 + 8(x-1) + 1 & x \in [1,2]. \end{cases}$$

3.9.1 Solving the Tridiagonal System

The matrix of coefficients of the linear system (3.90) has the tridiagonal form

$$A = \begin{bmatrix} a_1 & b_1 & & & & & \\ c_1 & a_2 & b_2 & & & & \\ & c_2 & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & \ddots & \ddots & b_{N-1} \\ & & & & & c_{N-1} & a_N \end{bmatrix} \tag{3.97}$$

where for the natural splines $(n-1)\times(n-1)$ system (3.90), the non-zero tridiagonal entries are

$$a_j = 2(h_{j-1}+h_j), \qquad j = 1,2,\ldots,n-1 \tag{3.98}$$

$$b_j = h_j, \qquad j = 1,2,\ldots,n-2 \tag{3.99}$$

$$c_j = h_j \qquad j = 1,2,\ldots,n-2. \tag{3.100}$$

We can solve the corresponding linear system of equations $Ax = d$ by factoring this matrix A into the product of a lower triangular matrix L and an upper triangular matrix U. To illustrate the idea let us take $N = 5$. A 5×5 tridiagonal linear system has the form

$$\begin{bmatrix} a_1 & b_1 & 0 & 0 & 0 \\ c_1 & a_2 & b_2 & 0 & 0 \\ 0 & c_2 & a_3 & b_3 & 0 \\ 0 & 0 & c_3 & a_4 & b_4 \\ 0 & 0 & 0 & c_4 & a_5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{bmatrix} \tag{3.101}$$

and we seek a factorization of the form

$$\begin{bmatrix} a_1 & b_1 & 0 & 0 & 0 \\ c_1 & a_2 & b_2 & 0 & 0 \\ 0 & c_2 & a_3 & b_3 & 0 \\ 0 & 0 & c_3 & a_4 & b_4 \\ 0 & 0 & 0 & c_4 & a_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ l_1 & 1 & 0 & 0 & 0 \\ 0 & l_2 & 1 & 0 & 0 \\ 0 & 0 & l_3 & 1 & 0 \\ 0 & 0 & 0 & l_4 & 1 \end{bmatrix} \begin{bmatrix} m_1 & u_1 & 0 & 0 & 0 \\ 0 & m_2 & u_2 & 0 & 0 \\ 0 & 0 & m_3 & u_3 & 0 \\ 0 & 0 & 0 & m_4 & u_4 \\ 0 & 0 & 0 & 0 & m_5 \end{bmatrix}$$

Note the the first matrix on the right hand side is lower triangular and the second one is upper triangular. Performing the product of the matrices and comparing with the corresponding entries of the left hand side matrix we have

$$\begin{aligned} \text{1st row:} \quad & a_1 = m_1, b_1 = u_1, \\ \text{2nd row:} \quad & c_1 = m_1 l_1, a_2 = l_1 u_1 + m_2, b_2 = u_2, \\ \text{3rd row:} \quad & c_2 = m_2 l_2, a_3 = l_2 u_2 + m_3, b_3 = u_3, \\ \text{4th row:} \quad & c_3 = m_3 l_3, a_4 = l_3 u_3 + m_4, b_4 = u_4, \\ \text{5th row:} \quad & c_4 = m_4 l_4, a_5 = l_4 u_4 + m_5, b_5 = u_5. \end{aligned}$$

So we can determine the unknowns in the following order

$$m_1; u_1, l_1, m_2; u_2, l_2, m_3; \ldots, u_4, l_4, m_5.$$

Since $u_j = b_j$ for all j we can write down the algorithm for general N as

% Determine the factorization coefficients
$m_1 = a_1$
for $j = 1 : N - 1$
    $l_j = c_j / m_j$
    $m_{j+1} = a_{j+1} - l_j * b_j$
end

% Forward substitution on $Ly = d$
$y_1 = d_1$
for $j = 2 : N$
    $y_j = d_j - l_{j-1} * y_{j-1}$
end

% Backward substitution to solve $Ux = y$
$x_N = y_N / m_N$
for $j = N - 1 : 1$
    $x_j = (y_j - b_j * x_{j+1}) / m_j$
end

3.9.2 Complete Splines

Sometimes it is more appropriate to specify the first derivative at the end points instead of the second derivative. This is called a complete or clamped spline. In this case $z_0 = f_0''$ and $z_n = f_n''$ become unknowns together with $z_1, z_2, \ldots, z_{n-1}$.

We need to add two more equations to have a complete system for all the $n + 1$ unknown values $z_0, z_1, \ldots, z_n$ in a complete spline. 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

$$s_0'(x_0) = C_0 = f_0' \tag{3.102}$$
$$s_{n-1}'(x_n) = 3A_{n-1}h_{n-1}^2 + 2B_{n-1}h_{n-1} + C_{n-1} = f_n'. \tag{3.103}$$

Substituting C_0, A_{n-1}, B_{n-1}, and C_{n-1} from (3.84)-(3.86) we get

$$2h_0z_0 + h_0z_1 = \frac{6}{h_0}(f_1 - f_0) - 6f_0', \tag{3.104}$$
$$h_{n-1}z_{n-1} + 2h_{n-1}z_n = -\frac{6}{h_{n-1}}(f_n - f_{n-1}) + 6f_n'. \tag{3.105}$$

These two equations together with (3.89) uniquely determine the second derivative values at all the nodes. The resulting $(n+1) \times (n+1)$ is also tridiagonal and diagonally dominant (hence nonsingular). Once the values $z_0, z_1, \ldots, z_n$ are found the splines coefficients are obtained from (3.84)-(3.87).

3.9.3 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 one needs to determine two splines interpolating (t_j, x_j) and (t_j, y_j) $(j = 0, 1, \ldots n)$, respectively.

The arc length of the curve is a natural choice for the parameter t. However, this is not known *a priori* and instead the nodes t_j's are usually chosen as the distances of consecutive, judiciously chosen points:

$$t_0 = 0, \quad t_j = t_{j-1} + \sqrt{(x_j - x_{j-1})^2 + (y_j - y_{j-1})^2}, \quad j = 1, 2, \ldots n. \tag{3.106}$$

Chapter 4

Trigonometric Approximation

We will study approximations employing truncated Fourier series. This type of approximation finds multiple applications in digital signal and image processing, and in the construction of highly accurate approximations to the solution of some partial differential equations. We will look at the problem of best approximation in the convenient L^2 norm and then consider the more practical approximation using interpolation. The latter will put us in the discrete framework to introduce the leading star of this chapter, the Discrete Fourier Transform, and one of the top ten algorithms of all time, the Fast Fourier Transform, to compute it.

4.1 Approximating a Periodic Function

We begin with the problem of approximating a periodic function f at the continuum level. Without loss of generality, we can assume that f is of period 2π (if it is of period p then the function $F(y) = f(\frac{p}{2\pi}y)$ has period 2π).

If f is a smooth periodic function we can approximate it with the first n few terms of its Fourier series:

$$f(x) \approx \frac{1}{2}a_0 + \sum_{k=1}^{n}(a_k \cos kx + b_k \sin kx), \tag{4.1}$$

where

$$a_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos kx\,dx, \quad k = 0, 1, \ldots, n \tag{4.2}$$

$$b_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin kx\,dx, \quad k = 1, 2, \ldots, n. \tag{4.3}$$

We will show that this is the best approximation to f, in the L^2 norm, by a trigonometric polynomial of degree n (the right hand side of (4.1)). For convenience, we write a trigonometric polynomial S_n (of degree n) in complex form (see (1.45)-(1.48)) as

$$S_n(x) = \sum_{k=-n}^{n} c_k e^{ikx}. \tag{4.4}$$

Consider the square of the error

$$J_n = \|f - S_n\|_2^2 = \int_0^{2\pi} [f(x) - S_n(x)]^2\,dx. \tag{4.5}$$

Let us try to find the coefficients c_k ($k = 0, \pm 1 \ldots, \pm n$) in (4.4) that minimize J_n. We have

$$\begin{aligned} J_n &= \int_0^{2\pi} \left[f(x) - \sum_{k=-n}^{n} c_k e^{ikx} \right]^2 dx \\ &= \int_0^{2\pi} [f(x)]^2 dx - 2\sum_{k=-n}^{n} c_k \int_0^{2\pi} f(x) e^{ikx} dx \\ &+ \sum_{k=-n}^{n}\sum_{l=-n}^{n} c_k c_l \int_0^{2\pi} e^{ikx} e^{ilx} dx. \end{aligned} \tag{4.6}$$

This problem simplifies if we use the orthogonality of the set $\{1, e^{ix}, e^{-ix}, \ldots, e^{inx}, e^{-inx}\}$, as for $k \neq -l$

$$\int_0^{2\pi} e^{ikx} e^{ilx} dx = \int_0^{2\pi} e^{i(k+l)x} dx = \left. \frac{1}{i(k+l)} e^{i(k+l)x} \right|_0^{2\pi} = 0 \tag{4.7}$$

and for $k = -l$

$$\int_0^{2\pi} e^{ikx} e^{ilx} dx = \int_0^{2\pi} dx = 2\pi. \tag{4.8}$$

Thus, we get

$$J_n = \int_0^{2\pi} [f(x)]^2 dx - 2\sum_{k=-n}^{n} c_k \int_0^{2\pi} f(x)e^{ikx} dx + 2\pi \sum_{k=-n}^{n} c_k c_{-k}. \tag{4.9}$$

J_n is a quadratic function of the coefficients c_k and so to find the its minimum, we determine the critical point of J_n as a function of the c_k's

$$\frac{\partial J_n}{\partial c_m} = -2\int_0^{2\pi} f(x)e^{imx} dx + 2(2\pi)c_{-m} = 0, \quad m = 0, \pm 1, \ldots, \pm n. \tag{4.10}$$

Therefore, relabeling the coefficients with k again, we get

$$c_k = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-ikx} dx, \quad k = 0, \pm 1, \ldots, \pm n, \tag{4.11}$$

which are the complex Fourier coefficients of f. The real Fourier coefficients (4.2)-(4.3) follow from Euler's formula $e^{ikx} = \cos kx + i \sin kx$, which produces the relation

$$c_0 = \frac{1}{2}a_0, \ c_k = \frac{1}{2}(a_k - ib_k), \ c_{-k} = \frac{1}{2}(a_k + ib_k), \ k = 1, \ldots, n. \tag{4.12}$$

This concludes the proof to the claim that the best L^2 approximation to f by trigonometric polynomials of degree n is furnished by the Fourier series of f truncated up to wave number $k = n$.

Now, if we substitute the Fourier coefficients (4.11) in (11.3) we get

$$0 \le J_n = \int_0^{2\pi} [f(x)]^2 dx - 2\pi \sum_{k=-n}^{n} |c_k|^2$$

that is

$$\sum_{k=-n}^{n} |c_k|^2 \le \frac{1}{2\pi}\int_0^{2\pi} [f(x)]^2 \, dx. \tag{4.13}$$

This is known as Bessel's inequality. In terms of the real Fourier coefficients, Bessel's inequality becomes

$$\frac{1}{2}a_0^2 + \sum_{k=1}^{n}(a_k^2 + b_k^2) \le \frac{1}{\pi}\int_0^{2\pi} [f(x)]^2 \, dx. \tag{4.14}$$

If $\int_0^{2\pi} [f(x)]^2 dx$ is finite, then then series

$$\frac{1}{2}a_0^2 + \sum_{k=1}^{\infty}(a_k^2 + b_k^2)$$

converges and consequently $\lim_{k\to\infty} a_k = \lim_{k\to\infty} b_k = 0$.

The convergence of a Fourier series is a delicate question. For a continuous function f, it does not follow that its Fourier series converges point-wise to it, only that it does so in the mean

$$\lim_{n\to\infty} \int_0^{2\pi} \left[f(x) - S_n(x)\right]^2 dx = 0. \tag{4.15}$$

This convergence in the mean for a continuous periodic function implies that Bessel's inequality becomes the equality

$$\frac{1}{2}a_0^2 + \sum_{k=1}^{\infty}(a_k^2 + b_k^2) = \frac{1}{\pi}\int_0^{2\pi} [f(x)]^2 \, dx, \tag{4.16}$$

which is known as Parseval's identity. We state now a convergence result without proof.

Theorem 4.1. *Suppose that f is piecewise continuous and periodic in $[0, 2\pi]$ and with a piecewise continuous first derivative. Then*

$$\frac{1}{2}a_0 + \sum_{k=1}^{\infty}(a_k \cos kx + b_k \sin kx) = \frac{1}{2}\left[f^+(x) + f^-(x)\right]$$

for each $x \in [0, 2\pi]$, where a_k and b_k are the Fourier coefficients of f. Here

$$\lim_{h\to 0^+} f(x+h) = f^+(x), \tag{4.17}$$

$$\lim_{h\to 0^+} f(x-h) = f^-(x). \tag{4.18}$$

In particular if x is a point of continuity of f

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

We have been working on the interval $[0, 2\pi]$ but we can choose any other interval of length 2π. This is so because if g is 2π periodic then we have the following result

Lemma 2.

$$\int_0^{2\pi} g(x)dx = \int_t^{t+2\pi} g(x)dx \tag{4.19}$$

for any real t.

Proof. Define

$$G(t) = \int_t^{t+2\pi} g(x)dx. \tag{4.20}$$

Then

$$G(t) = \int_t^0 g(x)dx + \int_0^{t+2\pi} g(x)dx = \int_0^{t+2\pi} g(x)dx - \int_0^t g(x)dx. \tag{4.21}$$

By the Fundamental theorem of calculus

$$G'(t) = g(t + 2\pi) - g(t) = 0$$

since g is 2π periodic. Thus, G is independent of t. □

Example 4.1. *$f(x) = |x|$ on $[-\pi, \pi]$.*

$$\begin{aligned}
a_k &= \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos kx dx = \frac{1}{\pi}\int_{-\pi}^{\pi} |x|\cos kx dx \\
&= \frac{2}{\pi}\int_0^{\pi} x\cos kx dx \\
&\quad u = x,\ dv = \cos kx dx \\
&\quad du = dx,\ v = \frac{1}{k}\sin kx \\
&= \frac{2}{\pi}\left[\frac{x}{k}\sin kx\Big|_0^{\pi} - \frac{1}{k}\int_0^{\pi}\sin kx dx\right] = \frac{2}{\pi k^2}\cos kx\Big|_0^{\pi} \\
&= \frac{2}{\pi k^2}[(-1)^k - 1], \quad k = 1, ...
\end{aligned}$$

and $a_0 = \pi$, $b_k = 0$ *for all* k *as the function is even. Therefore*

$$\begin{aligned} S(x) &= \lim_{n\to\infty} S_n(x) \\ &= \frac{1}{2}\pi + \sum_{k=1}^{\infty} \frac{2}{\pi k^2}[(-1)^k - 1]\cos kx \\ &= \frac{1}{2}\pi - \frac{4}{\pi}\left[\frac{\cos x}{1^2} + \frac{\cos 3x}{3^2} + \frac{\cos 5x}{5^2} + \cdots\right]. \end{aligned}$$

How do we find accurate numerical approximations to the Fourier coefficients? We know that for periodic smooth integrands, integrated over one (or multiple) period(s), the Composite Trapezoidal Rule using equidistributed nodes gives spectral accuracy. Let $x_j = jh$, $j = 0, 1, \ldots N$, $h = 2\pi/N$, then we can approximate

$$c_k \approx h\frac{1}{2\pi}\left[\frac{1}{2}f(x_0)e^{-ikx_0} + \sum_{j=1}^{N-1} f(x_j)e^{-ikx_j} + \frac{1}{2}f(x_N)e^{-ikx_N}\right].$$

But due to periodicity $f(x_0)e^{-ikx_0} = f(x_N)e^{-ikx_N}$ so we have

$$c_k \approx \frac{1}{N}\sum_{j=1}^{N} f(x_j)e^{-ikx_j} = \frac{1}{N}\sum_{j=0}^{N-1} f(x_j)e^{-ikx_j}. \tag{4.22}$$

and for the real Fourier coefficients we have

$$a_k \approx \frac{2}{N}\sum_{j=1}^{N} f(x_j)\cos kx_j = \frac{2}{N}\sum_{j=0}^{N-1} f(x_j)\cos kx_j, \tag{4.23}$$

$$b_k \approx \frac{2}{N}\sum_{j=1}^{N} f(x_j)\sin kx_j = \frac{2}{N}\sum_{j=0}^{N-1} f(x_j)\sin kx_j. \tag{4.24}$$

4.2 Interpolating Fourier Polynomial

Let f be a 2π-periodic function and $x_j = j\frac{2\pi}{N}$, $j = 0, 1, \ldots N$, equidistributed nodes in $[0, 2\pi]$. The interpolation problem is to find a trigonometric polynomial of lowest order S_n such that $S_n(x_j) = f(x_j)$, for $j = 0, 1, \ldots, N$. Because of periodicity $f(x_0) = f(x_N)$ so we only have N independent values.

If we take $N = 2n$ then S_n has $2n + 1 = N + 1$ coefficients. So we need one more condition. At the equidistributed nodes, the sine term of highest frequency vanishes as $\sin(\frac{N}{2}x_j) = \sin(j\pi) = 0$ so the coefficient $b_{N/2}$ is irrelevant for interpolation. We thus look for an interpolating trigonometric polynomial of the form

$$P_N(x) = \frac{1}{2}a_0 + \sum_{k=1}^{N/2-1} (a_k \cos kx + b_k \sin kx) + \frac{1}{2}a_{N/2} \cos\left(\frac{N}{2}x\right). \tag{4.25}$$

The convenience of the 1/2 factor in the last term will be seen in the formulas for the coefficients below. It is conceptually and computationally simpler to work with the corresponding polynomial in complex form

$$P_N(x) = \sum_{k=-N/2}^{N/2}{}'' \, c_k e^{ikx}, \tag{4.26}$$

where the double prime in the sum means that the first and last term (for $k = -N/2$ 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 (4.25).

Theorem 4.2.

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

with

$$c_k = \frac{1}{N} \sum_{j=0}^{N-1} f(x_j) e^{-ikx_j}, \quad k = -\frac{N}{2}, \ldots, \frac{N}{2}$$

interpolates f at the equidistributed points $x_j = j(2\pi/N)$, $j = 0, 1, \ldots, N$.

Proof. We have

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

Defining

$$l_j(x) = \frac{1}{N} \sum_{k=-N/2}^{N/2}{}'' \, e^{ik(x-x_j)} \tag{4.27}$$

we have

$$P_N(x) = \sum_{j=0}^{N-1} f(x_j) l_j(x). \tag{4.28}$$

Thus, we only need to prove that for j and m in the range $0, \ldots, N-1$

$$l_j(x_m) = \begin{cases} 1 & \text{for } m = j, \\ 0 & \text{for } m \neq j. \end{cases} \tag{4.29}$$

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

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

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

and, as we proved in the introduction

$$\frac{1}{N} \sum_{k=0}^{N-1} e^{-ik(j-m)2\pi/N} = \begin{cases} 0 & \text{if } j-m \text{ is not divisible by } N \\ 1 & \text{otherwise.} \end{cases} \tag{4.30}$$

Then (4.29) follows and

$$P_N(x_m) = f(x_m), \quad m = 0, 1, \ldots N-1.$$

□

Using the relations (4.12) between the c_k and the a_k and b_k coefficients we find that

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

interpolates f at the equidistributed nodes $x_j = j(2\pi/N)$, $j = 0, 1, \ldots, N$ if and only if

$$a_k = \frac{2}{N} \sum_{j=1}^{N} f(x_j) \cos kx_j, \tag{4.31}$$

$$b_k = \frac{2}{N} \sum_{j=1}^{N} f(x_j) \sin kx_j. \tag{4.32}$$

A smooth periodic function f can be approximated very accurately by its Fourier interpolant P_N. Note that derivatives of P_N can be easily computed

$$P_N^{(p)}(x) = \sum_{k=-N/2}^{N/2}{}'' (ik)^p c_k e^{ikx} \tag{4.33}$$

The discrete Fourier coefficients of the p-th derivative of P_N are $(ik)^p c_k$. Thus, once these Fourier coefficients have been computed a very accurate approximation of the derivatives of f is obtained, $f^{(p)}(x) \approx P_N^{(p)}(x)$. Figure 5.1 shows the approximation of $f(x) = \sin x e^{\cos x}$ on $[0, 2\pi]$ by P_8. The graph of f and of the Fourier interpolant are almost indistinguishable.

Let us go back to the complex Fourier interpolant (4.26). Its coefficients c_k are periodic of period N,

$$c_{k+N} = \frac{1}{N} \sum_{j=0}^{N-1} f_j e^{-i(k+N)x_j} = \frac{1}{N} \sum_{j=0}^{N-1} f_j e^{-ikx_j} e^{-ij2\pi} = c_k \tag{4.34}$$

and in particular $c_{-N/2} = c_{N/2}$. Using the interpolation property and setting $f_j = f(x_j)$, we have

$$f_j = \sum_{k=-N/2}^{N/2}{}'' c_k e^{ikx_j} = \sum_{k=-N/2}^{N/2-1} c_k e^{ikx_j} \tag{4.35}$$

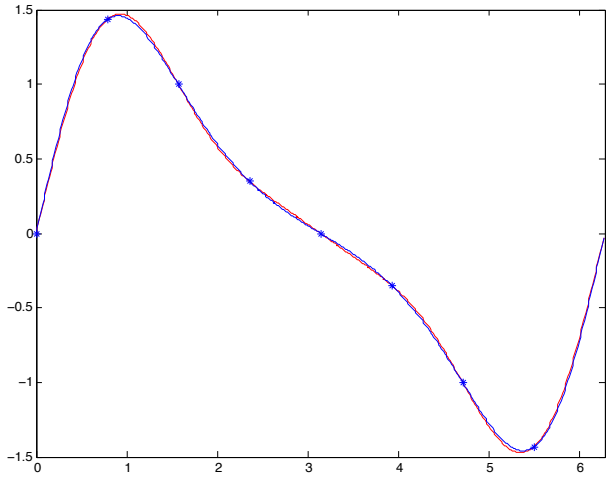

Figure 4.1: $S_8(x)$ for $f(x) = \sin x e^{\cos x}$ on $[0, 2\pi]$.

and

$$\begin{aligned}\sum_{k=-N/2}^{N/2-1} c_k e^{ikx_j} &= \sum_{k=-N/2}^{-1} c_k e^{ikx_j} + \sum_{k=0}^{N/2-1} c_k e^{ikx_j} \\ &= \sum_{k=N/2}^{N-1} c_k e^{ikx_j} + \sum_{k=0}^{N/2-1} c_k e^{ikx_j} = \sum_{k=0}^{N-1} c_k e^{ikx_j},\end{aligned} \tag{4.36}$$

where we have used that $c_{k+N} = c_k$. Combining this with the formula for the c_k's we get Discrete Fourier Transform (DFT) pair

$$c_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j e^{-ikx_j}, \quad k = 0, \ldots, N-1, \tag{4.37}$$

$$f_j = \sum_{k=0}^{N-1} c_k e^{ikx_j}, \qquad j = 0, \ldots, N-1. \tag{4.38}$$

The set of discrete coefficients (4.39) is known as DFT of the periodic array $f_0, f_1, \ldots, f_{N-1}$ and (4.40) is referred to as the Inverse DFT.

The direct evaluation of the DFT is computationally expensive, it requires order N^2 operations. However, there is a remarkable algorithm which

achieves this in merely order $N \log N$ operations. This is known as the Fast Fourier Transform.

4.3 The Fast Fourier Transform

The DFT was defined as

$$c_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j e^{-ikx_j}, \quad k = 0, \ldots, N-1, \tag{4.39}$$

$$f_j = \sum_{k=0}^{N-1} c_k e^{ikx_j}, \qquad j = 0, \ldots, N-1. \tag{4.40}$$

The direct computation of either (4.39) or (4.40) is requires order N^2 operations. As N increasing the cost quickly becomes prohibitive. In many applications N could easily be on the order of thousands, millions, etc.

One of the top algorithms of all times is the Fast Fourier Transform (FFT). It is usually attributed to Cooley and Tukey (1965) but its origin can be tracked back to C. F. Gauss (1777-1855). We now look at the main ideas of this famous and widely used algorithm.

Let us define $d_k = Nc_k$ for $k = 0, 1, \ldots, N-1$. Then we can rewrite (4.39) as

$$d_k = \sum_{j=0}^{N-1} f_j \omega_N^{kj}, \tag{4.41}$$

where $\omega_N = e^{-i2\pi/N}$. In matrix form

$$\begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{N-1} \end{bmatrix} = \begin{bmatrix} \omega_N^0 & \omega_N^0 & \cdots & \omega_N^0 \\ \omega_N^0 & \omega_N^1 & \cdots & \omega_N^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_N^0 & \omega_N^{N-1} & \cdots & \omega_N^{(N-1)^2} \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \end{bmatrix} \tag{4.42}$$

Let us call the matrix on the right F_N. Then F_N is N times the matrix of the DFT and the matrix of the Inverse DFT is simply the complex conjugate

of F_N. This follows from the identities:

$$\begin{aligned} 1+\omega_N+\omega_N^2+\cdots+\omega_N^{N-1} &= 0,\\ 1+\omega_N^2+\omega_N^4+\cdots+\omega_N^{2(N-1)} &= 0,\\ &\vdots\\ 1+\omega_N^{N-1}+\omega_N^{2(N-1)}+\cdots+\omega_N^{(N-1)^2} &= 0,\\ 1+\omega_N^{N}+\omega_N^{2N}+\cdots+\omega_N^{N(N-1)} &= N. \end{aligned}$$

We already proved the first of these identities. This geometric sum is equal to $(1-\omega_N^N)/(1-\omega_N)$, which in turn is equal to zero because $\omega_N^N = 1$. The other are proved similarly. We can summarize these identities as

$$\sum_{k=0}^{N-1} \omega_N^{jk} = \begin{cases} 0 & j \neq 0 \pmod N \\ N & j = 0 \pmod N, \end{cases} \tag{4.43}$$

where $j = k \pmod N$ means $j-k$ is an integer multiple of N. Then

$$\frac{1}{N}(F_N \bar{F}_N)_{jl} = \frac{1}{N}\sum_{k=0}^{N-1} \omega_N^{jk}\omega_N^{-lk} = \frac{1}{N}\sum_{k=0}^{N-1} \omega_N^{(j-l)k} = \begin{cases} 0 & j \neq l \pmod N \\ 1 & j = l \pmod N. \end{cases} \tag{4.44}$$

which shows that $\bar{F}_N$ is the inverse of $\frac{1}{N}F_N$.

Let $N = 2n$. Going back to (4.41), if we split the even-numbered and the odd-numbered points we have

$$d_k = \sum_{j=0}^{n-1} f_{2j}\omega_N^{2jk} + \sum_{j=0}^{n-1} f_{2j+1}\omega_N^{(2j+1)k} \tag{4.45}$$

But

$$\omega_N^{2jk} = e^{-i2jk\frac{2\pi}{N}} = e^{-ijk\frac{2\pi}{\frac{N}{2}}} = e^{-ijk\frac{2\pi}{n}} = \omega_n^{kj}, \tag{4.46}$$

$$\omega_N^{(2j+1)k} = e^{-i(2j+1)k\frac{2\pi}{N}} = e^{-ik\frac{2\pi}{N}}e^{-i2jk\frac{2\pi}{N}} = \omega_N^k\omega_n^{kj}. \tag{4.47}$$

So denoting $f_j^e = f_{2j}$ and $f_j^0 = f_{2j+1}$, we get

$$d_k = \sum_{j=0}^{n-1} f_j^e\omega_n^{jk} + \omega_N^k \sum_{j=0}^{n-1} f_j^0\omega_n^{jk} \tag{4.48}$$

We have reduced the problem to two DFT of size $n = \frac{N}{2}$ plus N multiplications (and N sums). The numbers ω_N^k, $k = 0, 1, \ldots, N-1$ only depend on N so they can be precomputed once and stored for other DFT 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. A very efficient implementation of the FFT is the FFTW ("the Fastest Fourier Transform in the West"), which employs a variety of code generation and runtime optimization techniques and is a free software.

Chapter 5

Least Squares Approximation

5.1 Continuous Least Squares Approximation

Let f be a continuous function on $[a, b]$. We would like to find the best approximation to f by a polynomial of degree at most n in the L^2 norm. We have already studied this problem for the approximation of periodic functions by trigonometric (complex exponential) polynomials. The problem is to find a polynomial p_n of degree $\leq n$ such that

$$\int_a^b [f(x) - p_n(x)]^2 dx = \min \tag{5.1}$$

Such polynomial, is also called the Least Squares approximation to f. As an illustration let us consider $n = 1$. We look for $p_1(x) = a_0 + a_1 x$, for $x \in [a, b]$, which minimizes

$$\begin{aligned} J(a_0, a_1) &= \int_a^b [f(x) - p_1(x)]^2 dx \\ &= \int_a^b f^2(x) dx - 2\int_a^b f(x)(a_0 + a_1 x) dx + \int_a^b (a_0 + a_1 x)^2 dx. \end{aligned} \tag{5.2}$$

$J(a_0, a_1)$ is a quadratic function of a_0 and a_1 and thus a necessary condition for the minimum is that it is a critical point:

$$\begin{aligned} \frac{\partial J(a_0, a_1)}{\partial a_0} &= -2\int_a^b f(x) dx + 2\int_a^b (a_0 + a_1 x) dx = 0, \\ \frac{\partial J(a_0, a_1)}{\partial a_1} &= -2\int_a^b x f(x) dx + 2\int_a^b (a_0 + a_1 x) x\, dx = 0, \end{aligned}$$

which yields the following linear 2×2 system for a_0 and a_1:

$$\left(\int_a^b 1dx\right) a_0 + \left(\int_a^b xdx\right) a_1 = \int_a^b f(x)dx, \tag{5.3}$$

$$\left(\int_a^b xdx\right) a_0 + \left(\int_a^b x^2dx\right) a_1 = \int_a^b xf(x)dx. \tag{5.4}$$

These two equations are known as the Normal Equations for $n = 1$.

Example 5.1. *Let $f(x) = e^x$ for $x \in [0, 1]$. Then*

$$\int_0^1 e^x dx = e - 1, \tag{5.5}$$

$$\int_0^1 xe^x dx = 1, \tag{5.6}$$

and the normal equations are

$$a_0 + \frac{1}{2}a_1 = e - 1, \tag{5.7}$$

$$\frac{1}{2}a_0 + \frac{1}{3}a_1 = 1, \tag{5.8}$$

whose solution is $a_0 = 4e - 10$, $a_1 = -6e + 18$. Therefore the least squares approximation to $f(x) = e^x$ by a linear polynomial is

$$p_1(x) = 4e - 10 + (18 - 6e)x.$$

p_1 and f are plotted in Fig. 5.1

The Least Squares Approximation to a function f on an interval $[a, b]$ by a polynomial of degree at most n is the best approximation of f in the L^2 norm by that class of polynomials. It is the polynomial

$$p_n(x) = a_0 + a_1x + \cdots + a_nx^n$$

such that

$$\int_a^b [f(x) - p_n(x)]^2 dx = \min. \tag{5.9}$$

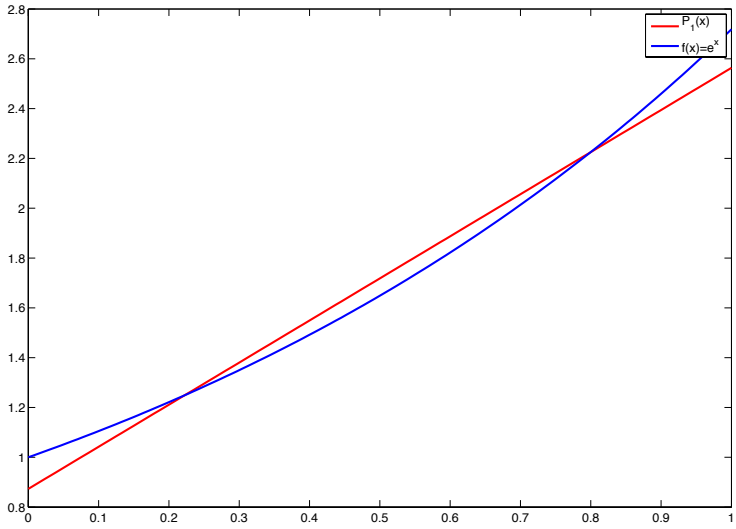

Figure 5.1: The function $f(x) = e^x$ on $[0, 1]$ and its Least Squares Approximation $p_1(x) = 4e - 10 + (18 - 6e)x$.

Defining this squared L^2 error as $J(a_0, a_1, ..., a_n)$ we have

$$\begin{aligned} J(a_0, a_1, ..., a_n) &= \int_a^b [f(x) - (a_0 + a_1 x + \cdots + a_n x^n)]^2 dx \\ &= \int_a^b f^2(x)dx - 2\sum_{k=0}^{n} a_k \int_a^b x^k f(x)dx + \sum_{k=0}^{n}\sum_{l=0}^{n} a_k a_l \int_a^b x^{k+l} dx. \end{aligned}$$

$J(a_0, a_1, ..., a_n)$ is a quadratic function of the parameters $a_0, ..., a_n$. A necessary condition is that the set of $a_0, ..., a_n$ which minimizes J is the critical point. That is

$$\begin{aligned} 0 &= \frac{\partial J(a_0, a_1, \ldots, a_n)}{\partial a_m} \\ &= -2\int_a^b x^m f(x)dx + \sum_{k=0}^{n} a_k \int_a^b x^{k+m} dx + \sum_{l=0}^{n} a_l \int_a^b x^{l+m} dx \\ &= -2\int_a^b x^m f(x)dx + 2\sum_{k=0}^{n} a_k \int_a^b x^{k+m} dx, \quad m = 0, 1, \ldots, n \end{aligned}$$

and we get the *Normal equations*

$$\sum_{k=0}^{n} a_k \int_a^b x^{k+m} dx = \int_a^b x^m f(x) dx, \quad m = 0, 1, ..., n. \tag{5.10}$$

We can write the Normal Equations in matrix as

$$\begin{bmatrix} \int_a^b 1dx & \int_a^b xdx & \cdots \int_a^b x^n dx \\ \int_a^b xdx & \int_a^b x^2 dx & \cdots \int_a^b x^{n+1} dx \\ \vdots & \vdots & \vdots \\ \int_a^b x^n dx & \int_a^b x^{n+1} dx & \cdots \int_a^b x^{2n} dx \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \int_a^b f(x) dx \\ \int_a^b x f(x) dx \\ \vdots \\ \int_a^b x^n f(x) dx \end{bmatrix}. \tag{5.11}$$

The matrix in this system is clearly symmetric. Denoting this matrix by H and $\mathbf{a} = [a_0\, a_1 \cdots a_n]^T$ any $n+1$ row vector, we have

$$\begin{aligned} \mathbf{a}^T H \mathbf{a} &= \sum_{i=0}^{n} \sum_{j=0}^{n} a_i a_j H_{ij} \\ &= \sum_{i=0}^{n} \sum_{j=0}^{n} a_i a_j \int_a^b x^{i+j} dx \\ &= \sum_{i=0}^{n} \sum_{j=0}^{n} \int_a^b a_i x^i a_j x^j dx \\ &= \int_a^b \sum_{i=0}^{n} \sum_{j=0}^{n} a_i x^i a_j x^j dx \\ &= \int_a^b (\sum_{j=0}^{n} a_j x^j)^2 dx \geq 0. \end{aligned}$$

Moreover, $\mathbf{a}^T H \mathbf{a} = 0$ if and only if $\mathbf{a} = \mathbf{0}$, i.e. H is *positive definite*. This implies that H is nonsigular and hence there is a unique solution to (5.11). For if there is $\mathbf{a} \neq \mathbf{0}$ such that $H\mathbf{a} = 0$ then $\mathbf{a}^T H \mathbf{a} = 0$ contradicting the fact that H is positive definite. Furthermore, the Hessian, $\frac{\partial^2 J}{\partial a_i \partial a_j}$ is equal to $2H$ so it is also positive definite and therefore the critical point is indeed a minimum.

In the interval $[0, 1]$,

$$H = \begin{bmatrix} \frac{1}{1} & \frac{1}{2} & \cdots & \frac{1}{n+1} \\ \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n+2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n+1} \end{bmatrix}, \tag{5.12}$$

which is known as the $[(n+1) \times (n+1)]$ Hilbert matrix.

In principle, the direct process to obtain the Least Squares Approximation p_n to f is to solve the normal equations (5.10) for the coefficients $a_0, a_1, \ldots, a_n$ and set $p_n(x) = a_0 + a_1 x + \ldots a_n x^n$. 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 H is very sensitive to small perturbations and this sensitivity increases rapidly with n. For example, numerical solutions in double precision (about 16 digits of accuracy) of a linear system with the Hilbert matrix (5.12) will lose all accuracy for $n \geq 11$.

2. If we want to increase the degree of the approximating polynomial we need to start over again and solve a larger set of normal equations. That is, we cannot use the $a_0, a_1, \ldots, a_n$ we already found.

It is more efficient and easier to solve the Least Squares Approximation problem using orthogonality, as we did with approximation by trigonometric polynomials. Suppose that we have a set of polynomials defined on an interval $[a, b]$,

$$\{\phi_0, \phi_1, ..., \phi_n\},$$

such that ϕ_k is a polynomial of degree k. Then, we can write any polynomial of degree at most n as a linear combination of these polynomials. In particular, the Least Square Approximating polynomial p_n can be written as

$$p_n(x) = a_0\phi_0(x) + a_1\phi_1(x) + ... + a_n\phi_n(x) = \sum_{k=0}^{n} a_k\phi_k(x),$$

for some coefficients $a_0, \ldots, a_n$ to be determined. Then

$$\begin{aligned} J(a_0, ..., a_n) &= \int_a^b [f(x) - \sum_{k=0}^{n} a_k \phi_j(x)]^2 dx \\ &= \int_a^b f^2(x)dx - 2\sum_{k=0}^{n} a_j \int_a^b \phi_k(x) f(x)dx \\ &+ \sum_{k=0}^{n}\sum_{l=0}^{n} a_k a_l \int_a^b \phi_k(x)\phi_l(x)dx \end{aligned} \tag{5.13}$$

and

$$0 = \frac{\partial J}{\partial a_m} = -2\int_a^b \phi_m(x) f(x)dx + 2\sum_{k=0}^{n} a_k \int_a^b \phi_k(x)\phi_m(x)dx.$$

for $m = 0, 1, \ldots, n$, which gives the normal equations

$$\sum_{k=0}^{n} a_k \int_a^b \phi_k(x)\phi_m(x)dx = \int_a^b \phi_m(x) f(x)dx, \quad m = 0, 1, ..., n. \tag{5.14}$$

Now, if the set of approximating functions $\{\phi_0, ...\phi_n\}$ is *orthogonal*, i.e.

$$\int_a^b \phi_k(x)\phi_m(x)dx = 0 \quad \text{if } k \neq m \tag{5.15}$$

then the coefficients of the least squares approximation are explicitly given by

$$a_m = \frac{1}{\alpha_m}\int_a^b \phi_m(x) f(x)dx, \quad \alpha_m = \int_a^b \phi_m^2(x)dx, \quad m = 0, 1, ..., n. \tag{5.16}$$

and

$$p_n(x) = a_0\phi_0(x) + a_1\phi_1(x) + ... + a_n\phi_n(x).$$

Note that if the set $\{\phi_0, ...\phi_n\}$ is orthogonal, (5.16) and (5.13) imply the *Bessel inequality*

$$\sum_{k=0}^{n} \alpha_k a_k^2 \leq \int_a^b f^2(x)dx. \tag{5.17}$$

This inequality shows that if f is square integrable, i.e. if

$$\int_a^b f^2(x)dx < \infty,$$

then the series $\sum_{k=0}^{\infty} \alpha_k a_k^2$ converges.

We can consider the Least Squares approximation for a class of linear combinations of orthogonal functions $\{\phi_0, ..., \phi_n\}$ not necessarily polynomials. We saw an example of this with Fourier approximations [1]. It is convenient to define a weighted L^2 norm associated with the Least Squares problem

$$\|f\|_{w,2} = \left(\int_a^b f^2(x)w(x)dx \right)^{\frac{1}{2}}, \tag{5.18}$$

where $w(x) \geq 0$ for all $x \in (a, b)$ [2]. A corresponding inner product is defined by

$$\langle f, g \rangle = \int_a^b f(x)g(x)w(x)dx. \tag{5.19}$$

Definition 5.1. *A set of functions $\{\phi_0, ..., \phi_n\}$ is orthogonal, with respect to the weighted inner product (5.19), if $\langle \phi_k, \phi_l \rangle = 0$ for $k \neq l$.*

5.2 Linear Independence and Gram-Schmidt Orthogonalization

Definition 5.2. *A set of functions $\{\phi_0(x), ..., \phi_n(x)\}$ defined on an interval $[a, b]$ is said to be linearly independent if*

$$a_0\phi_0(x) + a_1\phi_1(x) + \ldots a_n\phi_n(x) = 0, \quad \text{for all } x \in [a, b] \tag{5.20}$$

then $a_0 = a_1 = \ldots = a_n = 0$. Otherwise, it is said to be linearly dependent.

[1] For complex-valued functions orthogonality means $\int_a^b \phi_k(x)\bar{\phi}_l(x)dx = 0$ if $k \neq l$, where the bar denotes the complex conjugate

[2] More precisely, we will assume $w \geq 0$, $\int_a^b w(x)dx > 0$, and $\int_a^b x^k w(x)dx < +\infty$ for $k = 0, 1, \ldots$. We call such a w an admissible weight function.

Example 5.2. *The set of functions* $\{\phi_0(x), ..., \phi_n(x)\}$*, where* $\phi_k(x)$ *is a polynomial of degree* k *for* $k = 0, 1, \ldots, n$ *is linearly independent on any interval* $[a, b]$*. For* $a_0\phi_0(x) + a_1\phi_1(x) + \ldots a_n\phi_n(x)$ *is a polynomial of degree at most* n *and hence* $a_0\phi_0(x) + a_1\phi_1(x) + \ldots a_n\phi_n(x) = 0$ *for all* x *in a given interval* $[a, b]$ *implies* $a_0 = a_1 = \ldots = a_n = 0$*.*

Given a set of linearly independent functions $\{\phi_0(x), ..., \phi_n(x)\}$ we can produce an orthogonal set $\{\psi_0(x), ..., \psi_n(x)\}$ by doing the Gram-Schmidt procedure:

$$\begin{aligned}
\psi_0(x) &= \phi_0(x) \\
\psi_1(x) &= \phi_1(x) - c_0\psi_0(x), \\
&\quad \langle\psi_1, \psi_0\rangle = 0 \Rightarrow c_0 = \frac{\langle\phi_1, \psi_0\rangle}{\langle\psi_0, \psi_0\rangle} \\
\psi_2(x) &= \phi_2(x) - c_0\psi_0(x) - c_1\psi_1(x), \\
&\quad \langle\psi_2, \psi_0\rangle = 0 \Rightarrow c_0 = \frac{\langle\phi_2, \psi_0\rangle}{\langle\psi_0, \psi_0\rangle} \\
&\quad \langle\psi_2, \psi_1\rangle = 0 \Rightarrow c_1 = \frac{\langle\phi_2, \psi_1\rangle}{\langle\psi_1, \psi_1\rangle} \\
&\vdots
\end{aligned}$$

We can write this procedure recursively as

$$\begin{aligned}
&\psi_0(x) = \phi_0(x), \\
&\psi_k(x) = \phi_k(x) - \sum_{j=0}^{k-1} c_j\psi_j(x), \quad c_j = \frac{\langle\phi_k, \psi_j\rangle}{\langle\psi_j, \psi_j\rangle}.
\end{aligned} \tag{5.21}$$

5.3 Orthogonal Polynomials

Let us take the set $\{1, x, \ldots, x^n\}$ on a interval $[a, b]$. We can use the Gram-Schmidt process to obtain an orthogonal set $\{\psi_0(x), ..., \psi_n(x)\}$ of polynomials with respect to the inner product (5.19). Each of the ψ_k is a polynomial of degree k, determined up to a multiplicative constant (orthogonality is not changed). Suppose we select the $\psi_k(x)$, $k = 0, 1, \ldots, n$ to be monic, i.e. the coefficient of x^k is 1. Then $\psi_{k+1}(x) - x\psi_k(x) = r_k(x)$, where $r_k(x)$ is a

polynomial of degree at most k. So we can write

$$\psi_{k+1}(x) - x\psi_k(x) = -\alpha_k\psi_k(x) - \beta_k\psi_{k-1}(x) + \sum_{j=0}^{k-2} c_j\psi_j(x). \tag{5.22}$$

Then 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

$$\alpha_k = \frac{\langle x\psi_k, \psi_k\rangle}{\langle\psi_k, \psi_k\rangle}.$$

Similarly, taking the inner product 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 $x\psi_{k-1}(x) = \psi_k(x)+q_{k-1}(x)$, where $q_{k-1}(x)$ is a polynomial of degree at most $k-1$ then

$$\langle\psi_k, x\psi_{k-1}\rangle = \langle\psi_k, \psi_k\rangle + \langle\psi_k, q_{k-1}\rangle = \langle\psi_k, \psi_k\rangle,$$

where we have used orthogonality in the last equation. Therefore

$$\beta_k = \frac{\langle\psi_k, \psi_k\rangle}{\langle\psi_{k-1}, \psi_{k-1}\rangle}.$$

Finally, taking the inner product of (5.22) 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 $x\psi_m(x)$ is a polynomial of degree at most $k-1$ and hence it is orthogonal to $\psi_k(x)$. Collecting the results we obtain a three-term recursion formula

$$\psi_0(x) = 1, \tag{5.23}$$

$$\psi_1(x) = x - \alpha_0, \quad \alpha_0 = \frac{\langle x\psi_0, \psi_0\rangle}{\langle\psi_0, \psi_0\rangle} \tag{5.24}$$

for $k = 1, \ldots n$

$$\psi_{k+1}(x) = (x - \alpha_k)\psi_k(x) - \beta_k\psi_{k-1}(x), \tag{5.25}$$

$$\alpha_k = \frac{\langle x\psi_k, \psi_k\rangle}{\langle\psi_k, \psi_k\rangle}, \tag{5.26}$$

$$\beta_k = \frac{\langle\psi_k, \psi_k\rangle}{\langle\psi_{k-1}, \psi_{k-1}\rangle}. \tag{5.27}$$

Example 5.3. *Let* $[a, b] = [-1, 1]$ *and* $w(x) \equiv 1$. *The corresponding orthogonal polynomials are known as the* **Legendre Polynomials** *and are widely used in a variety of numerical methods. Because* $x\psi_k^2(x)w(x)$ *is an odd function it follows that* $\alpha_k = 0$ *for all* k. *We have* $\psi_0(x) = 1$ *and* $\psi_1(x) = x$. *We can now use the three-term recursion (5.25) 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 found:*

$$\begin{aligned} \psi_0(x) &= 1, \\ \psi_1(x) &= x, \\ \psi_2(x) &= x^2 - \frac{1}{3}, \\ \psi_3(x) &= x^3 - \frac{3}{5}x, \\ &\vdots \end{aligned}$$

Theorem 5.1. *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

$$\int_a^b \psi_k(x)w(x)dx = 0 \tag{5.28}$$

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 root, 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 $l = k$. □

5.3.1 Chebyshev Polynomials

We introduced in Section 2.4 the Chebyshev polynomials, which as we have seen possess remarkable properties. We now add one more important property of this outstanding class of polynomials, namely orthogonality.

The Chebyshev polynomials are orthogonal with respect to the weight function

$$w(x) = \frac{1}{\sqrt{1-x^2}}. \tag{5.29}$$

Indeed. Recall that $T_n(x) = \cos n\theta$, $(x = \cos\theta)$. Then,

$$\int_{-1}^{1} \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}}dx = \quad = \quad \int_0^{\pi} \cos n\theta \cos m\theta d\theta \tag{5.30}$$

and since $2\cos n\theta \cos m\theta = \cos(m+n)\theta + \cos(m-n)\theta$, we get for $m \neq n$

$$\int_{-1}^{1} \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}}dx = \frac{1}{2}\left[\frac{1}{n+m}\sin(n+m)\theta + \frac{1}{n-m}\sin(n-m)\theta\right]\Bigg|_0^{\pi} = 0.$$

Consequently,

$$\int_{-1}^{1} \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}}dx = \begin{cases} 0 & m \neq n, \\ \frac{\pi}{2} & m = n > 0, \\ \pi & m = n = 0. \end{cases} \tag{5.31}$$

5.4 Discrete Least Squares Approximation

Suppose that we are given the data $(x_1, f_1), (x_2, f_2), \cdots, (x_N, f_N)$ obtained from an experiment. Can we find a simple function that *appropriately fits* these data? Suppose that empirically we determine that there is an approximate linear behavior between f_j and x_j, $j = 1, \ldots, N$. What is the straight line $y = a_0 + a_1 x$ that best fits these data? The answer depends on how we measure the error, the deviations $f_j - (a_0 + a_1 x_j)$, i.e. which norm we use for the error. The most convenient measure is the squared of the 2-norm (Euclidean norm) because we will end up with a linear system of equations to find a_0 and a_1. Other norms will yield a nonlinear system for the unknown parameters. So the problem is: Find a_0, a_1 which minimize

$$J(a_0, a_1) = \sum_{j=1}^{N} [f_j - (a_0 + a_1 x_j)]^2. \tag{5.32}$$

We can repeat all the Least Squares Approximation theory that we have seen at the continuum level except that *integrals are replaced by sums*. The conditions for the minimum

$$\frac{\partial J(a_0, a_1)}{\partial a_0} = 2 \sum_{j=1}^{N} [f_j - (a_0 + a_1 x_j)](-1) = 0, \tag{5.33}$$

$$\frac{\partial J(a_0, a_1)}{\partial a_1} = 2 \sum_{j=1}^{N} [f_j - (a_0 + a_1 x_j)](-x_j) = 0, \tag{5.34}$$

produce the Normal Equations:

$$a_0 \sum_{j=1}^{N} 1 + a_1 \sum_{j=1}^{N} x_j = \sum_{j=1}^{N} f_j, \tag{5.35}$$

$$a_0 \sum_{j=1}^{N} x_j + a_1 \sum_{j=1}^{N} x_j^2 = \sum_{j=1}^{N} x_j f_j. \tag{5.36}$$

Approximation by a higher order polynomial

$$p_n(x) = a_0 + a_1 x + \cdots + a_n x^n, \quad n < N - 1$$

is similarly done. In practice we would like $n << N$ to avoid *overfitting*. We find the $a_0, a_1, ..., a_n$ which minimize

$$J(a_0, a_1, ..., a_n) = \sum_{j=1}^{N} [f_j - (a_0 + a_1 x_j + \cdots + a_n x_j^n)]^2. \tag{5.37}$$

Proceeding as in the continuum case, we get the *Normal Equations*

$$\sum_{k=0}^{n} a_k \sum_{j=1}^{N} x_j^{k+m} = \sum_{j=1}^{N} x_j^m f_j, \quad m = 0, 1, ..., n. \tag{5.38}$$

That is

$$a_0 \sum_{j=1}^{N} x_j^0 + a_1 \sum_{j=1}^{N} x_j^1 + \cdots + a_n \sum_{j=1}^{N} x_j^n = \sum_{j=1}^{N} x_j^0 f_j$$

$$a_0 \sum_{j=1}^{N} x_j^1 + a_1 \sum_{j=1}^{N} x_j^2 + \cdots + a_n \sum_{j=1}^{N} x_j^{n+1} = \sum_{j=1}^{N} x_j^1 f_j$$

$$\vdots$$

$$a_0 \sum_{j=1}^{N} x_j^n + a_1 \sum_{j=1}^{N} x_j^{n+1} + \cdots + a_n \sum_{j=1}^{N} x_j^{2n} = \sum_{j=1}^{N} x_j^n f_j.$$

The matrix of coefficients of this linear system is, as in the continuum case, symmetric, positive definite, and highly sensitive to small perturbations in the data. And again, if we want to increase the degree of the approximating polynomial we cannot reuse the coefficients we already computed for the lower order polynomial. But now we know how to go around these two problems: we use orthogonality.

Let us define the (weighted) discrete inner product as

$$\langle f, g \rangle_N = \sum_{j=1}^{N} f_j g_j \omega_j, \tag{5.39}$$

where $\omega_j > 0$, for $j = 1, \ldots, N$ are given weights. Let $\{\phi_0, ..., \phi_n\}$ be a set of polynomials such that ϕ_k is of degree exactly k. Then the solution to the

discrete Least Squares Approximation problem by a polynomial of degree $\leq n$ can be written as

$$p_n(x) = a_0\phi_0(x) + a_1\phi_1(x) + \cdots + a_n\phi_n(x) \tag{5.40}$$

and the square of the error is given by

$$J(a_0, \cdots, a_n) = \sum_{j=1}^{N} [f_j - (a_0\phi_0(x_j) + \cdots + a_n\phi_n(x_j))]^2\omega_j. \tag{5.41}$$

Consequently, the normal equations are

$$\sum_{k=0}^{n} a_k\langle\phi_k, \phi_l\rangle_N = \langle\phi_l, f\rangle_N, \quad l = 0, 1, ..., n. \tag{5.42}$$

If $\{\phi_0, \cdots, \phi_n\}$ are orthogonal with respect to the inner product $\langle\cdot,\cdot\rangle_N$, i.e. if $\langle\phi_k, \phi_l\rangle_N = 0$ for $k \neq l$, then the coefficients of the Least Squares Approximation are given by

$$a_k = \frac{\langle\phi_k, f\rangle_N}{\langle\phi_k, \phi_k\rangle_N}, \quad k = 0, 1, ..., n \tag{5.43}$$

and $p_n(x) = a_0\phi_0(x) + a_1\phi_1(x) + \cdots + a_n\phi_n(x)$.

If the $\{\phi_0, \cdots, \phi_n\}$ are not orthogonal we can produce an orthogonal set $\{\psi_0, \cdots, \psi_n\}$ using the 3-term recursion formula adapted to the discrete inner product. We have

$$\psi_0(x) \equiv 1, \tag{5.44}$$

$$\psi_1(x) = x - \alpha_0, \quad \alpha_0 = \frac{\langle x\psi_0, \psi_0\rangle_N}{\langle\psi_0, \psi_0\rangle_N}, \tag{5.45}$$

for $k = 1, ..., n$

$$\psi_{k+1}(x) = (x - \alpha_k)\psi_k(x) - \beta_k\psi_{k-1}(x), \tag{5.46}$$

$$\alpha_k = \frac{\langle x\psi_k, \psi_k\rangle_N}{\langle\psi_k, \psi_k\rangle_N}, \quad \beta_k = \frac{\langle\psi_k, \psi_k\rangle_N}{\langle\psi_{k-1}, \psi_{k-1}\rangle_N}. \tag{5.47}$$

Then,

$$a_j = \frac{\langle\psi_j, f\rangle_N}{\langle\psi_j, \psi_j\rangle_N}, \quad j = 0, ..., n. \tag{5.48}$$

and the Least Squares Approximation is $p_n(x) = a_0\psi_0(x) + a_1\psi_1(x) + \cdots + a_n\psi_n(x)$.

Example 5.4. *Suppose we are given the data:* $x_j : 0, 1, 2, 3$, $f_j = 1.1, 3.2, 5.1, 6.9$ *and we would like to fit to a line. The normal equations are*

$$a_0 \sum_{j=1}^{4} 1 + a_1 \sum_{j=1}^{4} x_j = \sum_{j=1}^{4} f_j \tag{5.49}$$

$$a_0 \sum_{j=1}^{4} x_j + a_1 \sum_{j=1}^{4} x_j^2 = \sum_{j=1}^{4} x_j f_j \tag{5.50}$$

and performing the sums we have

$$4a_0 + 6a_1 = 16.3, \tag{5.51}$$

$$6a_0 + 14a_1 = 34.1. \tag{5.52}$$

Solving this 2×2 *linear system we get* $a_0 = 1.18$ *and* $a_1 = 1.93$. *Thus, the Least Squares Approximation is*

$$p_1(x) = 1.18 + 1.93x$$

and the square of the error is

$$J(1.18, 193) = \sum_{j=1}^{4} [f_j - (1.18 + 1.93x_j)]^2 = 0.023.$$

Example 5.5. *Fitting to an exponential* $y = be^{ax_k}$. *Defining*

$$J(a, b) = \sum_{j=1}^{N} [f_j - be^{ax_j}]^2$$

we get the conditions for a *and* b

$$\frac{\partial J}{\partial a} = 2 \sum_{j=1}^{N} [f_j - be^{ax_j}](-bx_j e^{ax_j}) = 0, \tag{5.53}$$

$$\frac{\partial J}{\partial b} = 2 \sum_{j=1}^{N} [f_j - be^{ax_j}](-e^{ax_j}) = 0. \tag{5.54}$$

which is a nonlinear system of equations. However, if we take the natural log of $y = be^{ax_k}$ *we have* $\ln y = \ln b + ax$. *Defining* $B = \ln b$ *the problem becomes linear in* B *and* a. *Tabulating* $(x_j, \ln f_j)$ *we can obtain the normal equations*

$$B\sum_{j=1}^{N} 1 + a\sum_{j=1}^{N} x_j = \sum_{j=1}^{N} \ln f_j, \tag{5.55}$$

$$B\sum_{j=1}^{N} x_j + a\sum_{j=1}^{N} x_j^2 = \sum_{j=1}^{N} x_j \ln f_j, \tag{5.56}$$

and solve this linear system for B and a. Then $b = e^B$ and $a = a$.

If a is given and we only need to determine b then the problem is linear. From (5.54) we have

$$b\sum_{j=1}^{N} e^{2ax_j} = \sum_{j=1}^{N} f_j e^{ax_j} \Rightarrow b = \frac{\sum_{j=1}^{N} f_j e^{ax_j}}{\sum_{j=1}^{N} e^{2ax_j}}$$

Example 5.6. *Discrete orthogonal polynomials. Let us construct the first few orthogonal polynomials with respect to the discrete inner product with* $\omega \equiv 1$ *and* $x_j = \frac{j}{N}, j = 1, ..., N$. *Here* $N = 10$ *(the points are equidistributed in* $[0, 1]$*). We have* $\psi_0(x) = 1$ *and* $\psi_1(x) = x - \alpha_0$, *where*

$$\alpha_0 = \frac{\langle x\psi_0, \psi_0\rangle_N}{\langle \psi_0, \psi_0\rangle_N} = \frac{\sum_{j=1}^{N} x_j}{\sum_{j=1}^{N} 1} = 0.55.$$

and hence $\psi_1(x) = x - 0.55$. *Now*

$$\psi_2(x) = (x - \alpha_1)\psi_1(x) - \beta_1\psi_0(x), \tag{5.57}$$

$$\alpha_1 = \frac{\langle x\psi_1, \psi_1\rangle_N}{\langle \psi_1, \psi_1\rangle_N} = \frac{\sum_{j=1}^{N} x_j(x_j - 0.55)^2}{\sum_{j=1}^{N}(x_j - 0.55)^2} = 0.55, \tag{5.58}$$

$$\beta_1 = \frac{\langle \psi_1, \psi_1\rangle_N}{\langle \psi_0, \psi_0\rangle_N} = 0.0825. \tag{5.59}$$

Therefore $\psi_2(x) = (x - 0.55)^2 - 0.0825$. *We can now use these orthogonal polynomials to find the Least Squares Approximation by polynomial of degree*

at most two of a given set of data. Let us take $f_j = x_j^2 + 2x_j + 3$. *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 Least Squares Approximation coefficients are given by*

$$a_0 = \frac{\langle f, \psi_0 \rangle_N}{\langle \psi_0, \psi_0 \rangle_N} = 4.485, \tag{5.60}$$

$$a_1 = \frac{\langle f, \psi_1 \rangle_N}{\langle \psi_1, \psi_1 \rangle_N} = 3.1, \tag{5.61}$$

$$a_2 = \frac{\langle f, \psi_2 \rangle_N}{\langle \psi_2, \psi_2 \rangle_N} = 1, \tag{5.62}$$

which gives, $p_2(x) = (x-0.55)^2 - 0.0825 + (3.1)(x-0.55) + 4.485 = x^2 + 2x + 3$.

5.5 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 $\mathbf{x} \in \mathbb{R}^d$ for $d \geq 1$. Associated with $\mathbf{x}$ there is a scalar quantity f that can be measured or computed so that our data set consists of the points $(\mathbf{x}_j, f_j)$, where $\mathbf{x}_j \in \mathbb{R}^d$ 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 and most commonly used approach is to postulate a linear relation

$$f(\mathbf{x}) = a_0 + \mathbf{a}^T \mathbf{x} \tag{5.63}$$

and determine the *bias coefficient* a_0 and the vector $\mathbf{a} = [a_1, \ldots, a_d]^T$ as a least squares solution, i.e. such that they minimize

$$\sum_{j=1}^{N} [f_j - (a_0 + \mathbf{a}^T \mathbf{x}_j)]^2.$$

We have studied in detail the case $d = 1$. Here we are interested in the case $d >> 1$.

If we add an extra component, equal to 1, to each data vector $\mathbf{x}_j$ so that now $\mathbf{x}_j = [1, x_{j1}, \ldots, x_{jd}]^T$, for $j = 1, \ldots, N$, then we can write (5.63) as

$$f(\mathbf{x}) = \mathbf{a}^T \mathbf{x} \tag{5.64}$$

and the dimension d is increased by one. Then, we are seeking a vector $\mathbf{a} \in \mathbb{R}^d$ that minimizes

$$J(\mathbf{a}) = \sum_{j=1}^{N} [f_j - \mathbf{a}^T \mathbf{x}_j]^2. \tag{5.65}$$

Putting the data $\mathbf{x}_j$ as rows of an $N \times d$ ($N \geq d$) matrix X and f_j as the components of a (column) vector $\mathbf{f}$, i.e.

$$X = \begin{bmatrix} \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \\ \\ \end{bmatrix} \quad \text{and} \quad \mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix} \tag{5.66}$$

we can write (5.65) as

$$J(\mathbf{a}) = (\mathbf{f} - X\mathbf{a})^T (\mathbf{f} - X\mathbf{a}) = \|\mathbf{f} - X\mathbf{a}\|^2. \tag{5.67}$$

The normal equations are given by the condition $\nabla_{\mathbf{a}} J(\mathbf{a}) = 0$. Since $\nabla_{\mathbf{a}} J(\mathbf{a}) = -2X^T\mathbf{f} + 2X^T X\mathbf{a}$, we get the linear system of equations

$$X^T X\mathbf{a} = X^T \mathbf{f}. \tag{5.68}$$

Every solution of the least square problem is necessarily a solution of the normal equations. We will prove that the converse is also true and that the solutions have a geometric characterization.

Let W be the linear space spanned by the columns of X. Clearly, $W \subseteq \mathbb{R}^N$. Then, the least squares problem is equivalent to minimizing $\|\mathbf{f} - \mathbf{w}\|^2$ among all vectors $\mathbf{w}$ in W. There is always at least one solution, which can be obtained by projecting $\mathbf{f}$ onto W, as Fig. 5.2 illustrates. First, note that if $\mathbf{a} \in \mathbb{R}^d$ is a solution of the normal equations (5.68) then the residual $\mathbf{f} - X\mathbf{a}$ is orthogonal to W because

$$X^T(\mathbf{f} - X\mathbf{a}) = X^T\mathbf{f} - X^T X\mathbf{a} = \mathbf{0} \tag{5.69}$$

and a vector $\mathbf{r} \in \mathbb{R}^N$ is orthogonal to W if it is orthogonal to each column of X, i.e. $X^T\mathbf{r} = \mathbf{0}$. Let $\mathbf{a}^*$ be a solution of the normal equations, let $\mathbf{r} = \mathbf{f} - X\mathbf{a}^*$, and for arbitrary $\mathbf{a} \in \mathbb{R}^d$, let $\mathbf{s} = X\mathbf{a} - X\mathbf{a}^*$. Then, we have

$$\|\mathbf{f} - X\mathbf{a}\|^2 = \|\mathbf{f} - X\mathbf{a}^* - (X\mathbf{a} - X\mathbf{a}^*)\|^2 = \|\mathbf{r} - \mathbf{s}\|^2. \tag{5.70}$$

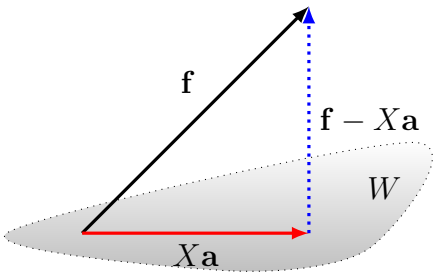

Figure 5.2: Geometric interpretation of the solution $X\mathbf{a}$ of the Least Squares problem as the orthogonal projection of $\mathbf{f}$ on the approximating linear subspace W.

But $\mathbf{r}$ and $\mathbf{s}$ are orthogonal. Therefore,

$$\|\mathbf{r}-\mathbf{s}\|^2 = \|\mathbf{r}\|^2 + \|\mathbf{s}\|^2 \geq \|\mathbf{r}\|^2 \tag{5.71}$$

and so we have proved that

$$\|\mathbf{f}-X\mathbf{a}\|^2 \geq \|\mathbf{f}-X\mathbf{a}^*\|^2 \tag{5.72}$$

for arbitrary $\mathbf{a} \in \mathbb{R}^d$, i.e. $\mathbf{a}^*$ minimizes $\|\mathbf{f}-X\mathbf{a}\|^2$.

If the columns of X are linearly independent, i.e. if for every $\mathbf{a} \neq \mathbf{0}$ we have that $X\mathbf{a} \neq \mathbf{0}$, then the $d \times d$ matrix X^TX is positive definite and hence nonsingular. Therefore, in this case, there is a unique solution to the least squares problem $\min_{\mathbf{a}} \|\mathbf{f}-X\mathbf{a}\|^2$ given by

$$\mathbf{a}^* = (X^TX)^{-1}X^T\mathbf{f}. \tag{5.73}$$

The $d \times N$ matrix

$$X^\dagger = (X^TX)^{-1}X^T \tag{5.74}$$

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}.

As we have done in the other least squares problems we have seen so far, rather than working with the normal equations, whose matrix X^TX may be very sensitive to perturbations in the data, we use an orthogonal basis for the approximating subspace (W in this case) to find a solution. While in principle this can be done by applying the Gram-Schmidt process to the columns of

X, this is a numerically unstable procedure; when two columns are nearly linearly dependent, errors introduced by the finite precision representation of computer numbers can be largely amplified during the Gram-Schmidt process and render vectors which are not orthogonal. A re-orthogonalization step can be introduced in the Gram-Schmidt algorithm to remedy this problem at the price of doubling the computational cost. A more efficient method using a sequence of orthogonal transformations, known as *Householder reflections*, is usually preferred. Once this orthonormalization process is completed we get a QR *factorization* of X

$$X = QR, \tag{5.75}$$

where Q is an $N \times N$ orthogonal matrix, i.e. $Q^T Q = I$, and R is an $N \times d$ upper triangular matrix

$$R = \begin{bmatrix} * & \cdots & * \\ & \ddots & \vdots \\ & & * \\ & & \\ & & \end{bmatrix} = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}. \tag{5.76}$$

Here R_1 is a $d \times d$ upper triangular matrix and the zero stands for a $(N-d) \times d$ zero matrix.

Using $X = QR$ we have

$$\|\mathbf{f} - X\mathbf{a}\|^2 = \|\mathbf{f} - QR\mathbf{a}\|^2 = \|Q^T\mathbf{f} - R\mathbf{a}\|^2. \tag{5.77}$$

Therefore, a least squares solution is obtained by solving the system $R\mathbf{a} = Q^T\mathbf{f}$. Writing R in blocks we have

$$\begin{bmatrix} R_1\mathbf{a} \\ 0 \end{bmatrix} = \begin{bmatrix} (Q^T\mathbf{f})_1 \\ (Q^T\mathbf{f})_2 \end{bmatrix}. \tag{5.78}$$

so that the solution is found by solving upper triangular system $R_1\mathbf{a} = (Q^T\mathbf{f})_1$ (R_1 is nonsingular if the columns of X are linearly independent). The last $N - d$ equations, $(Q^T\mathbf{f})_2 = 0$, may be satisfied or not depending on $\mathbf{f}$ but we have no control on them.

Chapter 6

Computer Arithmetic

6.1 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

$$x = \pm S \times 2^E, \quad 1 \leq S < 2, \tag{6.1}$$

where S is called the *significant* or *mantissa* and E is the exponent. In general S is an infinite expansion of the form

$$S = (1.b_1 b_2 \cdots)_2. \tag{6.2}$$

In a computer, a real number is represented in scientific notation but using a finite number of binary digits (bits). We call these **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* which can be represented in a computer. Moreover, the floating points numbers are not uniformly distributed!

The maximum exponent possible in DP would be $2^{11} = 2048$ but this is shifted to allow representation of small and large numbers so that we actually have $E_{\min} = -1022$, $E_{\max} = 1023$. Consequently, the min and max DP floating point number which can be represented in DP are

$$N_{\min} = \min_{x \in DP} |x| = 2^{-1022} \approx 2.2 \times 10^{-308}, \tag{6.3}$$

$$N_{\max} = \max_{x \in DP} |x| = (1.1.....1)_2 \cdot 2^{1023} = (2 - 2^{-52}) \cdot 2^{1023} \approx 1.8 \times 10^{308}. \tag{6.4}$$

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.

6.2 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.

$$x^* = \text{chop}(x) = \pm(1.b_1 b_2 \cdots b_d)_2 \times 2^E. \tag{6.5}$$

In double precision $d = 52$.

A better way to approximate to 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

$$x^* = \text{round}(x) = \text{chop}(x + 2^{-(d+1)} \times 2^E). \tag{6.6}$$

Definition 6.1. *Given an approximation x^* to x the* **absolute error** *is defined by $|x - x^*|$ and the* **relative error** *by $|\frac{x-x^*}{x}|$, $x \neq 0$.*

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**) is

$$\left|\frac{x - \text{chop}(x)}{x}\right| \leq \frac{2^{-d}2^E}{(1.b_1b_2\cdots)2^E} \leq 2^{-d}, \tag{6.7}$$

$$\left|\frac{x - \text{round}(x)}{x}\right| \leq \frac{1}{2}2^{-d}. \tag{6.8}$$

The number 2^{-d} is called **machine precision** or epsilon (eps). In double precision eps=$2^{-52} \approx 2.22 \times 10^{-16}$. The smallest double precision number greater than 1 is 1+eps. As we will see below, it is more convenient to write (6.8) as

$$\text{round}(x) = x(1+\delta), \quad |\delta| \leq \text{eps}. \tag{6.9}$$

6.3 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

$$x \oplus y = \text{round}(x+y) = (x+y)(1+\delta_+), \quad |\delta_+| \leq \text{eps}, \tag{6.10}$$

and similarly for $\ominus, \otimes, \oslash$.

One important interpretation of (6.10) is the following. Assuming $x+y \neq 0$, write

$$\delta_+ = \frac{1}{x+y}[\delta_x + \delta_y].$$

Then

$$x \oplus y = (x+y)\left[1 + \frac{1}{x+y}(\delta_x + \delta_y)\right] = (x+\delta_x) + (y+\delta_y). \tag{6.11}$$

The computer $\oplus$ 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.

6.4 Propagation of Errors and Cancellation of Digits

Let $fl(x)$ and $fl(y)$ denote the floating point approximation of x and y, respectively, and assume that their product is computed exactly, i.e

$$fl(x)\cdot fl(y) = x(1+\delta_x)\cdot y(1+\delta_y) = x\cdot y(1+\delta_x+\delta_y+\delta_x\delta_y) \approx x\cdot y(1+\delta_x+\delta_y),$$

where $|\delta_x|$,$|\delta_y| \leq$ eps. Therefore, for the relative error we get

$$\left|\frac{x\cdot y - fl(x)\cdot fl(y)}{x\cdot y}\right| \approx |\delta_x+\delta_y|, \tag{6.12}$$

which is acceptable.

Let us now consider addition (or subtraction):

$$\begin{aligned} fl(x)+fl(y) &= x(1+\delta_x)+y(1+\delta_y) = x+y+x\delta_x+y\delta_y \\ &= (x+y)\left(1+\frac{x}{x+y}\delta_x+\frac{y}{x+y}\delta_y\right). \end{aligned}$$

The relative error is

$$\left|\frac{x+y-(fl(x)+fl(y))}{x+y}\right| = \left|\frac{x}{x+y}\delta_x+\frac{y}{x+y}\delta_y\right|. \tag{6.13}$$

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, the error could be largely amplified because $|\frac{x}{x+y}|, |\frac{y}{x+y}|$ can be very large.

Example 6.1. *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 (i.e. garbage) that say were generated in previous floating point computations. Then, in this 10 bit precision arithmetic

$$z = x-y = (1.00********)_2 \times 2^{E-6}. \tag{6.14}$$

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 6.2. *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.

Chapter 7

Numerical Differentiation

7.1 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 $x_1, x_2, ..., x_n$. We could approximate f by its interpolating polynomial p_n at those points and use $f'(x_0) \approx p_n'(x_0)$. There are several other possibilities. For example, we can approximate $f'(x_0)$ by the derivative of the cubic spline of f evaluated at x_0, or by the derivative of the Least Squares Chebyshev expansion of f:

$$f'(x_0) = \sum_{j=1}^{n} a_j T_j'(x_0),$$

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

$$f(x) = p_n(x) + \frac{1}{(n+1)!} f^{(n+1)}(\xi(x))\, \omega_n(x), \tag{7.1}$$

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

$$\omega_n(x) = (x - x_0)(x - x_1) \cdots (x - x_n). \tag{7.2}$$

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

$$f'(x_0) = p_n'(x_0) + \frac{1}{(n+1)!} f^{(n+1)}(\xi(x))(x_0 - x_1)\cdots(x_0 - x_n) \tag{7.3}$$

Example 7.1. *Take $n = 1$ and $x_1 = x_0 + h$ ($h > 0$). In Newton's form*

$$p_1(x) = f(x_0) + \frac{f(x_0 + h) - f(x_0)}{h}(x - x_0), \tag{7.4}$$

and $p_1'(x_0) = \frac{1}{h}[f(x_0+h) - f(x_0)]$. We obtain the so-called Forward Difference Formula for approximating $f'(x_0)$

$$D_h^+ f(x_0) := \frac{f(x_0 + h) - f(x_0)}{h}. \tag{7.5}$$

From (7.3) the error in this approximation is

$$f'(x_0) - D_h^+ f(x_0) = \frac{1}{2!} f''(\xi(x))(x_0 - x_1) = -\frac{1}{2} f''(\xi) h. \tag{7.6}$$

Example 7.2. *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 so-called Backward Difference Formula for approximating $f'(x_0)$*

$$D_h^- f(x_0) := \frac{f(x_0) - f(x_0 - h)}{h}. \tag{7.7}$$

Its error is

$$f'(x_0) - D_h^- f(x_0) = \frac{1}{2} f''(\xi) h. \tag{7.8}$$

Example 7.3. *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 finite difference table:

$$\begin{array}{lll} x_0 - h & f(x_0 - h) & \\ & & \frac{f(x_0) - f(x_0 - h)}{h} \\ x_0 & f(x_0) & \qquad\qquad \frac{f(x_0+h) - 2f(x_0) + f(x_0+h)}{2h^2} \\ & & \frac{f(x_0+h) - f(x_0)}{h} \\ x_0 + h & f(x_0 + h) & \end{array}$$

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

$$p_2'(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h}. \tag{7.9}$$

This defines the Centered Difference Formula to approximate $f'(x_0)$

$$D_h^0 f(x_0) := \frac{f(x_0 + h) - f(x_0 - h)}{2h}. \tag{7.10}$$

Its error is

$$f'(x_0) - D_h^0 f(x_0) = \frac{1}{3!} f'''(\xi)(x_0 - x_1)(x_0 - x_2) = -\frac{1}{6} f'''(\xi) h^2. \tag{7.11}$$

Example 7.4. *Let* $n = 2$ *and* $x_1 = x_0 + h$, $x_2 = x_0 + 2h$. *The table of finite differences is*

$$\begin{array}{cc} x_0 & f(x_0) \\ & \frac{f(x_0+h)-f(x_0)}{h} \\ x_0 + h & f(x_0 + h) \\ & \frac{f(x_0+2h)-f(x_0+h)}{h} \\ x_0 + 2h & f(x_0 + 2h) \end{array} \qquad \frac{f(x_0+2h)-2f(x_0+h)+f(x_0)}{2h^2}$$

and

$$p_2'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} + \frac{f(x_0 + 2h) - 2f(x_0 + h) + f(x_0)}{2h^2} h$$

thus

$$p_2'(x_0) = \frac{-f(x_0 + 2h) + 4f(x_0 + h) - 3f(x_0)}{2h}. \tag{7.12}$$

If we use this sided difference to approximate $f'(x_0)$, *the error is*

$$f'(x_0) - p_2'(x_0) = \frac{1}{3!} f'''(\xi)(x_0 - x_1)(x_0 - x_2) = \frac{1}{3} h^2 f'''(\xi), \tag{7.13}$$

which is twice as large as that in Centered Finite Difference Formula.

7.2 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. Consider for example the centered difference formula. 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 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 $|\delta_-| \leq \text{eps}$. Then

$$\frac{f(x_0+h)(1+\delta_+) - f(x_0-h)(1+\delta_-)}{2h} = \frac{f(x_0+h)-f(x_0-h)}{2h} + r_h,$$

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 $-\frac{1}{6}f'''(\xi)h^2$. Thus, the total error $E(h)$ can be approximately bounded by $\frac{1}{6}h^2M_3 + |f(x_0)|\frac{\text{eps}}{h}$. The minimum error occurs at h_0 such that $E'(h_0)=0$, i.e.

$$h_0 = \left(\frac{3\ \text{eps}\ |f(x_0)|}{M_3}\right)^{\frac{1}{3}} \approx c\ \text{eps}^{\frac{1}{3}} \tag{7.14}$$

and $E(h_0) = O(\text{eps}^{\frac{2}{3}})$. We do not get machine precision!

Higher order finite differences exacerbate the problem of digit cancellation. When f can be extended to an analytic function in the complex plane, Cauchy Integral Theorem can be used to evaluate the derivative:

$$f'(z_0) = \frac{1}{2\pi i}\int_C \frac{f(z)}{(z-z_0)^2}dz, \tag{7.15}$$

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

$$f'(z_0) = \frac{1}{2\pi r}\int_0^{2\pi} f(x_0+re^{it})e^{-it}dt \tag{7.16}$$

The integrand is periodic and smooth so it can be approximated with spectral accuracy with the composite trapezoidal rule.

Another approach to obtain finite difference formulas to approximate derivatives is through Taylor expansions. For example,

$$f(x_0+h) = f(x_0) + f'(x_0)h + \frac{1}{2}f''(x_0)h^2 + \frac{1}{3!}f^{(3)}(x_0)h^3 + \frac{1}{4!}f^{(4)}(\xi_+)h^4, \tag{7.17}$$

$$f(x_0-h) = f(x_0) - f'(x_0)h + \frac{1}{2}f''(x_0)h^2 - \frac{1}{3!}f^{(3)}(x_0)h^3 + \frac{1}{4!}f^{(4)}(\xi_-)h^4, \tag{7.18}$$

where $x_0 < \xi_+ < x_0 + h$ and $x_0 - h < \xi_- < x_0$. Then subtracting (7.17) from (7.18) 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

$$\frac{f(x_0+h) - f(x_0-h)}{2h} = f'(x_0) + c_2h^2 + c_4h^4 + \cdots \tag{7.19}$$

Similarly if we add (7.17) and (7.18) we obtain $f(x_0+h) + f(x_0-h) = 2f(x_0) + f''(x_0)h^2 + ch^4 + \cdots$ and consequently

$$f''(x_0) = \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{h^2} + \tilde{c}h^2 + \cdots \tag{7.20}$$

The finite difference

$$D_h^2 f(x_0) = \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{h^2} \tag{7.21}$$

is thus a second order approximation to $f''(x_0)$, i.e., $f''(x_0) - D_h^2 f(x_0) = O(h^2)$.

7.3 Richardson's Extrapolation

From (7.19) we know that, asymptotically

$$D_h^0 f(x_0) = f'(x_0) + c_2h^2 + c_4h^4 + \cdots \tag{7.22}$$

We can apply Richardson extrapolation once to obtain a fourth order approximation. Evaluating (7.22) at $h/2$ we get

$$D_{h/2}^0 f(x_0) = f'(x_0) + \frac{1}{4}c_2h^2 + \frac{1}{16}c_4h^4 + \cdots \tag{7.23}$$

and multiplying this equation by 4, subtracting (7.22) to the result and dividing by 3 we get

$$D_h^{ext} f(x_0) := \frac{4D_{h/2}^0 f(x_0) - D_h^0 f(x_0)}{3} = f'(x_0) + \tilde{c}_4 h^4 + \cdots \tag{7.24}$$

The method $D_h^{ext} f(x_0)$ has order of convergence 4 for about twice the amount of work of that $D_h^0 f(x_0)$. Round-off errors are still $O(\text{eps}/h)$ and the minimum total error will be when $O(h^4)$ is $O(\text{eps}/h)$, i.e. when $h = \text{eps}^{1/5}$. The minimum error is thus $O(\text{eps}^{4/5})$ for $D_h^{ext} f(x_0)$, about 10^{-14} in double precision with $h = O(10^{-3})$.

Chapter 8

Numerical Integration

We now revisit the problem of numerical integration that we used to introduced some principles of numerical analysis in Chapter 1.

The problem in question is to find accurate and efficient approximations to

$$\int_a^b f(x)dx$$

Numerical formulas to approximate a definite integral are called *quadratures* 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.

8.1 Elementary Simpson Quadrature

The elementary trapezoidal rule quadrature was derived by replacing the integrand f by its linear interpolating polynomial p_1 at a and b, that is

$$f(x) = p_1(x) + \frac{1}{2}f''(\xi)(x-a)(x-b), \tag{8.1}$$

for some ξ between a and b and thus

$$\begin{aligned}\int_a^b f(x)dx &= \int_a^b p_1(x)dx + \frac{1}{2}\int_a^b f''(\xi)(x-a)(x-b)dx \\ &= \frac{1}{2}(b-a)[f(a)+f(b)] - \frac{1}{2}f''(\eta)(b-a)^3\end{aligned} \tag{8.2}$$

Thus, the approximation

$$\int_a^b f(x)dx \approx \frac{1}{2}(b-a)[f(a)+f(b)] \tag{8.3}$$

has an error given by $-\frac{1}{2}f''(\eta)(b-a)^3$.

We can add an intermediate point, say $x_m = (a+b)/2$, and replace f by its quadratic interpolating polynomial p_2 with respect to the nodes a, x_m and b. For simplicity let's take $[a,b] = [-1,1]$. With the simple change of variables

$$x = \frac{1}{2}(a+b) + \frac{1}{2}(b-a)t, \qquad t \in [-1,1] \tag{8.4}$$

we can obtain a quadrature formula for a general interval $[a,b]$.

Let p_2 be the interpolating polynomial of f at $-1, 0, 1$. The corresponding divided difference table is:

$$\begin{array}{lll} -1 & f(-1) & \\ & & f(0)-f(-1) \\ 0 & f(0) & \qquad\qquad \frac{f(1)-2f(0)+f(-1)}{2}. \\ & & f(1)-f(0) \\ 1 & f(1) & \end{array}$$

Thus

$$p_2(x) = f(-1) + [f(0)-f(1)](x+1) + \frac{f(1)-2f(0)+f(-1)}{2}(x+1)x. \tag{8.5}$$

Now using the interpolation formula with remainder expressed in terms of a divided difference (3.56) we have

$$\begin{aligned} f(x) &= p_2(x) + f[-1,0,1,x](x+1)x(x-1) \\ &= p_2(x) + f[-1,0,1,x]x(x^2-1). \end{aligned} \tag{8.6}$$

Therefore

$$\begin{aligned} \int_{-1}^1 f(x)dx &= \int_{-1}^1 p_2(x)dx + \int_{-1}^1 f[-1,0,1,x]x(x^2-1)dx \\ &= 2f(-1) + 2[f(0)-f(-1)] + \frac{1}{3}[f(1)-2f(0)+f(-1)] + E[f] \\ &= \frac{1}{3}[f(-1)+4f(0)+f(1)] + E[f], \end{aligned}$$

where

$$E[f] = \int_{-1}^{1} f[-1,0,1,x]x(x^2-1)dx \tag{8.7}$$

is the error. Note that $x(x^2-1)$ changes sign in $[-1,1]$ so we cannot use the Mean Value Theorem for integrals. However, if we add another node, x_4, we can relate $f[-1,0,1,x]$ to the fourth order divided difference $f[-1,0,1,x_4,x]$, which will make the integral in (8.7) easier to evaluate:

$$f[-1,0,1,x] = f[-1,0,1,x_4] + f[-1,0,1,x_4,x](x-x_4). \tag{8.8}$$

This identity is just an application of Theorem 3.2. Using (8.8)

$$E[f] = f[-1,0,1,x_4]\int_{-1}^{1} x(x^2-1)dx + \int_{-1}^{1} f[-1,0,1,x_4,x]x(x^2-1)(x-x_4)dx.$$

The first integral is zero, because the integrand is odd. Now we choose x_4 symmetrically, $x_4 = 0$, so that $x(x^2-1)(x-x_4)$ does not change sign in $[-1,1]$ and

$$E[f] = \int_{-1}^{1} f[-1,0,1,0,x]x^2(x^2-1)dx = \int_{-1}^{1} f[-1,0,0,1,x]x^2(x^2-1)dx. \tag{8.9}$$

Now, using (3.58), there is $\xi(x) \in (-1,1)$ such that

$$f[-1,0,0,1,x] = \frac{f^{(4)}(\xi(x))}{4!}, \tag{8.10}$$

and assuming $f \in C^4[-1,1]$, by the Mean Value Theorem for integrals, there is $\eta \in (-1,1)$ such that

$$E[f] = \frac{f^{(4)}(\eta)}{4!}\int_{-1}^{1} x^2(x^2-1)dx = -\frac{4}{15}\frac{f^{(4)}(\eta)}{4!} = -\frac{1}{90}f^{(4)}(\eta). \tag{8.11}$$

Summarizing, Simpson's elementary quadrature for the interval $[-1,1]$ is

$$\int_{-1}^{1} f(x)dx = \frac{1}{3}[f(-1)+4f(0)+f(1)] - \frac{1}{90}f^{(4)}(\eta). \tag{8.12}$$

Note that Simpson's elementary quadrature gives the exact value of the integral when f is polynomial of degree 3 or less (the error is proportional to the fourth derivative), even though we used a second order polynomial to approximate the integrand. We gain extra precision because of 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 $-1, 0, 0, 1$.

To obtain the corresponding formula for a general interval $[a, b]$ we use the change of variables (8.4)

$$\int_a^b f(x)dx = \frac{1}{2}(b-a)\int_{-1}^{1} F(t)dt,$$

where

$$F(t) = f\left(\frac{1}{2}(a+b) + \frac{1}{2}(b-a)t\right), \tag{8.13}$$

and noting that $F^{(k)}(t) = (\frac{b-a}{2})^k f^{(k)}(x)$ we obtain Simpson's elementary rule on the interval $[a, b]$:

$$\begin{aligned}\int_a^b f(x)dx = \frac{1}{6}(b-a)\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] \\ - \frac{1}{90}f^{(4)}(\eta)\left(\frac{b-a}{2}\right)^5.\end{aligned} \tag{8.14}$$

8.2 Interpolatory Quadratures

The elementary trapezoidal and Simpson 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 rule is exact for polynomial up to degree 3 and we used p_2 in its construction. The "degree gain" was 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 k possible?

2. What is that k?

Because orthogonal polynomials (Section 5.3) play a central role in the answer to these questions, we will consider the more general problem of approximating

$$I[f] = \int_a^b f(x)w(x)dx, \tag{8.15}$$

where w is an admissible weight function ($w \geq 0$, $\int_a^b w(x)dx > 0$, and $\int_a^b x^k w(x)dx < +\infty$ for $k = 0, 1, \ldots$), $w \equiv 1$ being a particular case. The interval of integration $[a, b]$ can be either finite or infinite (e.g. $[0, +\infty]$, $[-\infty, +\infty]$).

Definition 8.1. *We say that a quadrature $Q[f]$ to approximate $I[f]$ has degree of precision k if it is exact, i.e. $I[P] = Q[P]$, for all polynomials P of degree up to 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 8.1. *The trapezoidal rule quadrature has degree of precision 1 while the Simpson 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)

$$p_n(x) = \sum_{j=0}^{n} f(x_j)l_j(x), \tag{8.16}$$

where

$$l_j(x) = \prod_{k=0, k\neq j}^{n} \frac{(x - x_k)}{(x_j - x_k)}, \quad \text{for } j = 0, 1, ..., n. \tag{8.17}$$

are the elementary Lagrange polynomials. The corresponding interpolatory quadrature $Q_n[f]$ to approximate $I[f]$ is then given by

$$Q_n[f] = \sum_{j=0}^{n} A_j f(x_j), \quad A_j = \int_a^b l_j(x)w(x)dx, \text{ for } j = 0, 1, \ldots, n. \tag{8.18}$$

Theorem 8.1. *Degree of precision of the interpolatory quadrature (8.18) is less than* $2n + 2$

Proof. Suppose the degree of precision k of (8.18) 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

$$\int_a^b f(x)w(x)dx = \sum_{j=0}^{n} A_j f(x_j) = 0. \tag{8.19}$$

and on the other hand

$$\int_a^b f(x)w(x)dx = \int_a^b (x - x_0)^2 \cdots (x - x_n)^2 w(x)dx > 0 \tag{8.20}$$

which is a contradiction. Therefore $k < 2n + 2$. □

8.3 Gaussian Quadratures

We will now show that there is a choice of nodes $x_0, x_1, ..., x_n$ which yields the optimal 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

$$< f, g >= \int_a^b f(x)g(x)w(x)dx, \tag{8.21}$$

if $< \psi_k, q >= 0$ for all polynomials q of degree less than k. Recall also that the zeros of the orthogonal polynomials are real, simple, and contained in $[a, b]$ (see Theorem 5.1).

Definition 8.2. *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 (8.18) with the nodes so chosen is called a Gaussian quadrature.*

Theorem 8.2. *The interpolatory quadrature (8.18) has degree of precision $k = 2n + 1$ if and only if it is a Gaussian quadrature.*

Proof. Let f is a polynomial of degree $\leq 2n + 1$. Then, we can write

$$f(x) = q(x)\psi_{n+1}(x) + r(x), \tag{8.22}$$

where q and r are polynomials of degree $\leq n$. Now

$$\int_a^b f(x)w(x)dx = \int_a^b q(x)\psi_{n+1}(x)w(x)dx + \int_a^b r(x)w(x)dx \tag{8.23}$$

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

$$\int_a^b f(x)w(x)dx = \sum_{j=0}^{n} A_j r(x_j). \tag{8.24}$$

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,

$$\int_a^b f(x)w(x)dx = \sum_{j=0}^{n} A_j f(x_j). \tag{8.25}$$

This proves that the Gaussian quadrature has degree of precision $k = 2n+1$. Now suppose that the interpolatory quadrature (8.18) 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}. □

Example 8.2. *Consider the interval $[-1, 1]$ and the weight function $w \equiv 1$. The corresponding orthogonal the Legendre Polynomials $1, x, x^2 - \frac{1}{3}, x^3 -$*

$\frac{3}{5}x, \cdots$. *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*

$$\int_{-1}^{1} f(x)dx \approx A_0 f\left(-\sqrt{\frac{1}{3}}\right) + A_1 f\left(\sqrt{\frac{1}{3}}\right) \tag{8.26}$$

where

$$A_0 = \int_{-1}^{1} l_0(x)dx, \tag{8.27}$$

$$A_1 = \int_{-1}^{1} l_1(x)dx. \tag{8.28}$$

We can evaluate these integrals directly or use 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*

$$2 = \int_{-1}^{1} 1dx = A_0 + A_1, \tag{8.29}$$

$$0 = \int_{-1}^{1} xdx = -A_0\sqrt{\frac{1}{3}} + A_1\sqrt{\frac{1}{3}}. \tag{8.30}$$

Solving this 2×2 *linear system we get* $A_0 = A_1 = 1$. *So the Gaussian quadrature for* $n = 1$ *in* $[-1, 1]$ *is*

$$Q_1[f] = f\left(-\sqrt{\frac{1}{3}}\right) + f\left(\sqrt{\frac{1}{3}}\right) \tag{8.31}$$

Let us compare this quadrature to the elementary trapezoidal rule. Take $f(x) = x^2$. *The trapezoidal rule,* $T[f]$, *gives*

$$T[x^2] = \frac{2}{2}[f(-1) + f(1)] = 2 \tag{8.32}$$

whereas the Gaussian quadrature $Q_1[f]$ *yields the exact result:*

$$Q_1[x^2] = \left(-\sqrt{\frac{1}{3}}\right)^2 + \left(\sqrt{\frac{1}{3}}\right)^2 = \frac{2}{3}. \tag{8.33}$$

Example 8.3. *Let us take again the interval* $[-1,1]$ *but now* $w(x)=\frac{1}{\sqrt{1-x^2}}$. *As we know (see 2.4),* $\psi_{n+1}=T_{n+1}$, *i.e.the Chebyshev polynomial of degree* $n+1$. *Its zeros are* $x_j=\cos[\frac{2j+1}{2(n+1)}\pi]$ *for* $j=0,\ldots,n$. *For* $n=1$ *we have*

$$\cos\left(\frac{\pi}{4}\right)=\sqrt{\frac{1}{2}},\quad \cos\left(\frac{5\pi}{4}\right)=-\sqrt{\frac{1}{2}}. \tag{8.34}$$

We can use again the method of undetermined coefficients to find A_0 *and* A_1*:*

$$\pi=\int_{-1}^{1}1\frac{1}{\sqrt{1-x^2}}dx=A_0+A_1, \tag{8.35}$$

$$0=\int_{-1}^{1}x\frac{1}{\sqrt{1-x^2}}dx=-A_0\sqrt{\frac{1}{2}}+A_1\sqrt{\frac{1}{2}}, \tag{8.36}$$

which give $A_0=A_1=\frac{\pi}{2}$. *Thus, the corresponding Gaussian quadrature to approximate* $\int_{-1}^{1}f(x)\frac{1}{\sqrt{1-x^2}}dx$ *is*

$$Q_1[f]=\frac{\pi}{2}\left[f\left(-\sqrt{\frac{1}{2}}\right)+f\left(\sqrt{\frac{1}{2}}\right)\right]. \tag{8.37}$$

8.3.1 Convergence of Gaussian Quadratures

Let $f\in C[a,b]$ and consider the interpolation quadrature (8.18). Can we guarantee that the error converges to zero as $n\to\infty$, i.e.,

$$\int_a^b f(x)w(x)dx-\sum_{j=0}^{n}A_jf(x_j)\to 0,\ \text{as } n\to\infty\ ?$$

The answer is no. As we know, 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 a Gaussian the answer is yes. This follows from the following special properties of the quadrature weights $A_0,A_1,\ldots,A_n$ in the Gaussian quadrature.

Theorem 8.3. *For a Gaussian quadrature all the quadrature weights are positive and sum up to* $\|w\|_1$, *i.e.,*

(a) $A_j>0$ *for all* $j=0,1,\ldots,n$.

(b) $\sum_{j=0}^{n} A_j = \int_a^b w(x)dx.$

Proof. (a) Let $p_k = l_k^2$ for $k = 0, 1, \ldots, n$. These are polynomials of degree exactly equal to $2n$ and $p_k(x_j) = \delta_{kj}$. Thus,

$$0 < \int_a^b l_k^2(x)w(x)dx = \sum_{j=0}^{n} A_j l_k^2(x_j) = A_k \tag{8.38}$$

for $k = 0, 1, \ldots, n$.
(b) Take $f(x) \equiv 1$ then

$$\int_a^b w(x)dx = \sum_{j=0}^{n} A_j. \tag{8.39}$$

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 $f \in C[a, b]$:

Theorem 8.4. *Let*

$$Q_n[f] = \sum_{j=0}^{n} A_j f(x_j) \tag{8.40}$$

be the Gaussian quadrature. Then

$$E_n[f] := \int_a^b f(x)w(x)dx - Q_n[f] \to 0, \text{ as } n \to \infty\ . \tag{8.41}$$

Proof. Let p_{2n+1}^* be the best uniform approximation to f (in the max norm, $\|f\|_\infty = \max_{x\in[a,b]} |f(x)|$) by polynomials of degree $\leq 2n + 1$. Then,

$$E_n[f - p_{2n+1}^*] = E_n[f] - E_n[p_{2n+1}^*] = E_n[f] \tag{8.42}$$

and therefore

$$E_n[f] = E_n[f - p_{2n+1}^*] = \int_a^b [f(x) - p_{2n+1}^*(x)]w(x)dx - \sum_{j=0}^{n} A_j[f(x_j) - p_{2n+1}^*(x_j)].$$

Taking the absolute value, using the triangle inequality, and the fact that the quadrature 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 $E_n[f] \to 0$ as $n \to \infty$. □

Moreover, one can prove (using one of the Jackson Theorems) that if $f \in C^m[a,b]$

$$|E_n[f]| \leq C(2n)^{-m}\|f^{(m)}\|_\infty. \tag{8.43}$$

That is, the rate of convergence is not fixed; it depends on the number of derivatives the integrand has. We say in this case 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 $1/(2n)$.

8.4 Computing the Gaussian Nodes and Weights

Orthogonal polynomials satisfy a three-term relation:

$$\psi_{k+1}(x) = (x - \alpha_k)\psi_k(x) - \beta_k\psi_{k-1}(x), \quad \text{for } k = 0, 1, \ldots, n, \tag{8.44}$$

where β_0 is defined by $\int_a^b w(x)dx$, $\psi_0(x) = 1$ and $\psi_{-1}(x) = 0$. Equivalently

$$x\psi_k(x) = \beta_k\psi_{k-1}(x) + \alpha_k\psi_k(x) + \psi_{k+1}(x), \quad \text{for } k = 0, 1, \ldots, n. \tag{8.45}$$

If we use the normalized orthogonal polynomials

$$\tilde{\psi}_k(x) = \frac{\psi_k(x)}{\sqrt{< \psi_k, \psi_k >}} \tag{8.46}$$

and recalling that

$$\beta_k = \frac{< \psi_k, \psi_k >}{< \psi_{k-1}, \psi_{k-1} >}$$

then (8.45) can be written as

$$x\tilde{\psi}_k(x) = \sqrt{\beta_k}\tilde{\psi}_{k-1}(x) + \alpha_k\tilde{\psi}_k(x) + \sqrt{\beta_{k+1}}\tilde{\psi}_{k+1}(x), \quad \text{for } k = 0, 1, \ldots, n. \tag{8.47}$$

Now evaluating this at a root x_j of ψ_{n+1} we get the eigenvalue problem

$$x_j\mathbf{v}_j = J_n\mathbf{v}_j, \tag{8.48}$$

where

$$J_n = \begin{bmatrix} \alpha_0 & \sqrt{\beta_1} & 0 & \cdots & 0 \\ \sqrt{\beta_1} & \alpha_0 & \sqrt{\beta_2} & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \\ & & & & \sqrt{\beta_n} \\ 0 & 0 & 0 & \sqrt{\beta_n} & \alpha_n \end{bmatrix}, \qquad \mathbf{v}_j = \begin{bmatrix} \tilde{\psi}_0(x_j) \\ \tilde{\psi}_1(x_j) \\ \vdots \\ \tilde{\psi}_{n-1}(x_j) \\ \tilde{\psi}_n(x_j) \end{bmatrix}. \tag{8.49}$$

That is, the Gaussian nodes x_j, $j = 0, 1, \ldots, n$ are the eigenvalues of the *Jacobi Matrix* J_n. One can show that the Gaussian weights A_j, are given in terms of the first component $v_{j,0}$ of the (normalized) eigenvector $\mathbf{v}_j$ ($\mathbf{v}_j^T\mathbf{v}_j = 1$):

$$A_j = \beta_0 v_{j,0}^2. \tag{8.50}$$

There are efficient numerical methods (e.g. the QR method) to solve the eigenvalue problem for a symmetric triadiagonal matrix and this is one of most popular approached to compute the Gaussian nodes.

8.5 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.

Suppose f is a smooth function in $[-1, 1]$ and we are interested in an accurate approximation of the integral

$$\int_{-1}^{1} f(x)dx.$$

The idea is to use the extrema of the n Chebyshev polynomial T_n, $x_j = \cos(\frac{j\pi}{n})$, $j = 0, 1, ..., n$ as the nodes of the corresponding interpolatory quadrature. The degree of precision is only n (not $2n + 1$!). However, as we know, for smooth functions the approximation by polynomial interpolation using the Chebyshev nodes converges very rapidly. Hence, for smooth integrands this particular interpolatory quadrature can be expected to converge fast to the exact value of the integral.

Let p_n be the interpolating polynomial of f at $x_j = \cos(\frac{j\pi}{n})$, $j = 0, 1, ..., n$. We can write p_n as

$$p_n(x) = \frac{a_0}{2} + \sum_{k=1}^{n-1} a_k T_k(x) + \frac{a_n}{2} T_n(x) \tag{8.51}$$

Under the change of variable $x = \cos\theta$, for $\theta \in [0, \pi]$ we get

$$p_n(\cos\theta) = \frac{a_0}{2} + \sum_{k=1}^{n-1} a_k \cos k\theta + \frac{1}{2} a_n \cos n\theta. \tag{8.52}$$

Let $\Pi_n(\theta) = p_n(\cos\theta)$ and $F(\theta) = f(\cos\theta)$. By extending F evenly over $[-\pi, 0]$ (or over $[\pi, 2\pi]$) and using Theorem 4.2, we conclude that $\Pi_n(\theta)$ interpolates $F(\theta) = f(\cos\theta)$ at the equally spaced points $\theta_j = \frac{j\pi}{n}$, $j = 0, 1, ...n$ if and only if

$$a_k = \frac{2}{n} \sum_{j=0}^{n}{}'' F(\theta_j) \cos k\theta_j, \qquad k = 0, 1, .., n. \tag{8.53}$$

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 FFT.

Now, using the change of variable $x = \cos\theta$ we have

$$\int_{-1}^{1} f(x)dx = \int_{0}^{\pi} f(\cos\theta) \sin\theta d\theta = \int_{0}^{\pi} F(\theta) \sin\theta d\theta, \tag{8.54}$$

and approximating $F(\theta)$ by its interpolant $\Pi_n(\theta) = P_n(\cos\theta)$, we obtain the corresponding quadradure

$$\int_{-1}^{1} f(x)dx \approx \int_{0}^{\pi} \Pi_n(\theta) \sin\theta d\theta. \tag{8.55}$$

Substituting (8.52) we have

$$\int_0^\pi \Pi_n(\theta)\sin\theta d\theta = \frac{a_0}{2}\int_0^\pi \sin\theta d\theta + \sum_{k=1}^{n-1} a_k \int_0^\pi \cos k\theta \sin\theta d\theta + \frac{a_n}{2}\int_0^\pi \cos n\theta \sin\theta d\theta. \tag{8.56}$$

Assuming n even and using $\cos k\theta \sin\theta = \frac{1}{2}[\sin(1+k)\theta + \sin(1-k)\theta]$ we get the Clenshaw-Curtis quadrature:

$$\int_{-1}^1 f(x)dx \approx a_0 + \sum_{\substack{k=2 \\ k \text{ even}}}^{n-2} \frac{2a_k}{1-k^2} + \frac{a_n}{1-n^2}. \tag{8.57}$$

For a general interval $[a,b]$ we simply use the change of variables

$$x = \frac{a+b}{2} + \frac{b-a}{2}\cos\theta \tag{8.58}$$

for $\theta \in [0,\pi]$ and thus

$$\int_b^a f(x)dx = \frac{b-a}{2}\int_0^\pi F(\theta)\sin\theta d\theta, \tag{8.59}$$

where $F(\theta) = f(\frac{a+b}{2} + \frac{b-a}{2}\cos\theta)$ and so the formula (8.57) gets an extra factor of $(b-a)/2$.

8.6 Composite Quadratures

We saw in Section 1.2.2 that one strategy to improve the accuracy of a 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:

$$\int_a^b f(x)dx = h\left[\frac{1}{2}f(a) + \sum_{j=1}^{N-1} f(x_j) + \frac{1}{2}f(b)\right] - \frac{1}{12}(b-a)h^2 f''(\eta), \tag{8.60}$$

where η is some point in (a, b).

To derive a corresponding *composite* Simpson quadrature we take N even and apply the elementary Simpson quadrature in each of the $N/2$ intervals $[x_j, x_{j+2}]$, $j = 0, ..., N-2$. That is:

$$\int_a^b f(x)dx = \int_{x_0}^{x_2} f(x)dx + \int_{x_2}^{x_4} f(x)dx + \cdots + \int_{x_{N-2}}^{x_N} f(x)dx \tag{8.61}$$

and since the elementary Simpson quadrature applied to $[x_j, x_{j+2}]$ reads:

$$\int_{x_j}^{x_{j+2}} f(x)dx = \frac{h}{3}\left[f(x_j) + 4f(x_{j+1}) + f(x_{j+2})\right] - \frac{1}{90}f^{(4)}(\eta_j)h^5 \tag{8.62}$$

for some $\eta_j \in (x_j, x_{j+2})$, summing up all the $N/2$ contributions we get the composite Simpson quadrature:

$$\int_a^b f(x)dx = \frac{h}{3}\left[f(a) + 2\sum_{j=1}^{N/2-1} f(x_{2j}) + 4\sum_{j=1}^{N/2} f(x_{2j-1}) + f(b)\right] - \frac{1}{180}(b-a)h^4 f^{(4)}(\eta),$$

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

8.7 Modified Trapezoidal Rule

We are going to consider here a modification to the trapezoidal rule that will yield a quadrature with an error of the same order as Simpson's rule. Moreover, this modified quadrature will give us some insight to the the asymptotic form of the trapezoidal rule error.

To simplify the derivation let us consider the interval $[0, 1]$ and let p_3 be the polynomial interpolating $f(0), f'(0), f(1), f'(1)$. Newton's divided differences representation of p_3 is

$$p_3(x) = f(0) + f[0,0]x + f[0,0,1]x^2 + f[0,0,1,1]x^2(x-1), \tag{8.63}$$

and thus

$$\int_0^1 p_3(x)dx = f(0) + \frac{1}{2}f'(0) + \frac{1}{3}f[0,0,1] - \frac{1}{12}f[0,0,1,1]. \tag{8.64}$$

Th divided differences are obtained in the tableau:

$$\begin{array}{c|cccc}
0 & f(0) & & & \\
 & & f'(0) & & \\
0 & f(0) & & f(1)-f(0)-f'(0) & \\
 & & f(1)-f(0) & & f'(1)+f'(0)+2(f(0)-f(1)) \\
1 & f(1) & & f'(1)-f(1)+f(0) & \\
 & & f'(1) & & \\
1 & f(1) & & &
\end{array}$$

Thus,

$$\int_0^1 p_3(x)dx = f(0)+\frac{1}{2}f'(0)+\frac{1}{3}[f(1)-f(0)-f'(0)]-\frac{1}{12}[f'(0)+f'(1)+2(f(0)-f(1))]$$

and simplifying the right hand side we get

$$\int_0^1 p_3(x)dx = \frac{1}{2}[f(0)+f(1)]+\frac{1}{12}[f'(0)-f'(1)], \tag{8.65}$$

which is the simple trapezoidal rule plus a correction involving the derivative of the integrand at the end points.

We can obtain an expression for the error of this quadrature formula by recalling that the Cauchy remainder in the interpolation is

$$f(x)-p_3(x) = \frac{1}{4!}f^{(4)}(\xi(x))x^2(x-1)^2 \tag{8.66}$$

and since $x^2(x-1)^2$ does not change sign in $[0,1]$ we can use the mean value Theorem for integrals to get

$$E[f] = \int_0^1 [f(x)-p_3(x)]dx = \frac{1}{4!}f^{(4)}(\eta)\int_0^1 x^2(x-1)^2dx = \frac{1}{720}f^{(4)}(\eta) \tag{8.67}$$

for some $\eta \in (0,1)$.

To obtain the quadrature in a general finite interval $[a,b]$ we use the change of variables $x = a+(b-a)t,\ t\in[0,1]$

$$\int_a^b f(x)dx = (b-a)\int_0^1 F(t)dt, \tag{8.68}$$

where $F(t) = f(a + (b-a)t)$. Thus,

$$\int_a^b f(x)dx = \frac{b-a}{2}[f(a)+f(b)] + \frac{(b-a)^2}{12}[f'(a)-f'(b)] + \frac{1}{720}f^{(4))}(\eta)(b-a)^5, \tag{8.69}$$

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

We can get a composite modified trapezoidal rule by subdividing $[a,b]$ in N subintervals of equal length $h = \frac{b-a}{N}$, applying the simple rule in each subinterval and adding up all the contributions:

$$\begin{aligned}\int_a^b f(x)dx = h\left[\frac{1}{2}f(x_0) + \sum_{j=1}^{N-1} f(x_j) + \frac{1}{2}f(x_N)\right] - \frac{h^2}{12}[f'(b)-f'(a)] \\ + \frac{1}{720}f^{(4)}(\eta)h^4.\end{aligned} \tag{8.70}$$

8.8 The Euler-Maclaurin Formula

We are now going to obtain a more general formula for the asymptotic form of the error in the trapezoidal rule quadrature. The idea is to use integration by parts with the aid of suitable polynomials. Let us consider again the interval $[0,1]$ and define $B_0(x) = 1, B_1(x) = x - \frac{1}{2}$, then

$$\begin{aligned}\int_0^1 f(x)dx &= \int_0^1 f(x)B_0(x)dx = \int_0^1 f(x)B_1'(x)dx \\ &= f(x)B_1(x)|_0^1 - \int_0^1 f'(x)B_1(x)dx \\ &= \frac{1}{2}[f(0)+f(1)] - \int_0^1 f'(x)B_1(x)dx\end{aligned} \tag{8.71}$$

We can continue the integration by parts using the *Bernoulli Polynomials* which satisfy

$$B_{k+1}'(x) = (k+1)B_k(x), \qquad k = 1,2,\ldots \tag{8.72}$$

Since we start with $B_1(x) = x - \frac{1}{2}$ it is clear that $B_k(x)$ is a polynomial of degree exactly k with leading order coefficient 1, i.e. monic. These polynomials are determined by the recurrence relation (8.72) up to a constant. The

constant is fixed by requiring that

$$B_k(0) = B_k(1) = 0, \qquad k = 3, 5, 7, \ldots \tag{8.73}$$

Indeed,

$$B''_{k+1}(x) = (k+1)B'_k(x) = (k+1)kB_{k-1}(x) \tag{8.74}$$

and $B_{k-1}(x)$ has the form

$$B_{k-1}(x) = x^{k-1} + a_{k-2}x^{k-2} + \ldots a_1 x + a_0. \tag{8.75}$$

Integrating (8.74) twice we get

$$B_{k+1}(x) = k(k+1)\left[\frac{1}{k(k+1)}x^{k+1} + \frac{a_{k-2}}{(k-1)k}x^k + \ldots + \frac{1}{2}a_0x^2 + bx + c\right] \tag{8.76}$$

For $k+1$ odd, the two constants of integration b and c are determined by the condition (8.73). The $B_k(x)$ for k even are then given by $B_k(x) = B'_{k+1}(x)/(k+1)$.

We are going to need a few properties of the Bernoulli polynomials. Because of construction, $B_k(x)$ is an even (odd) polynomial in $x - \frac{1}{2}$ is k is even (odd). Equivalently, they satisfy the identity

$$(-1)^k B_k(1-x) = B_k(x). \tag{8.77}$$

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 have are the same. From (8.77) and (8.73) we get that

$$B_k(0) = B_k(1), \qquad k = 2, 3, , \ldots \tag{8.78}$$

We define *Bernoulli numbers* as $B_k = B_k(0) = B_k(1)$. This together with the recurrence relation (8.72) implies that

$$\int_0^1 B_k(x)dx = \frac{1}{k+1}\int_0^1 B'_{k+1}(x)dx = \frac{1}{k+1}\left[B_{k+1}(1) - B_{k+1}(0)\right] = 0 \tag{8.79}$$

for $k = 1, 2, \ldots$.

Lemma 3. *The polynomials $C_{2m}(x) = B_{2m}(x) - B_{2m}$, $m = 1, 2, \ldots$ do not change sign in $[0, 1]$.*

Proof. We will prove it by contradiction. Let us suppose that $C_{2M}(x)$ changes sign. Then it has at least 3 zeros and, by Rolle's theorem, $C'_{2m}(x) = B'_{2m}(x)$ 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. We then 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

$$B_0(x) = 1 \tag{8.80}$$

$$B_1(x) = x - \frac{1}{2} \tag{8.81}$$

$$B_2(x) = \left(x - \frac{1}{2}\right)^2 - \frac{1}{12} = x^2 - x + \frac{1}{6} \tag{8.82}$$

$$B_3(x) = \left(x - \frac{1}{2}\right)^3 - \frac{1}{4}\left(x - \frac{1}{2}\right) = x^3 - \frac{3}{2}x^2 + \frac{1}{2}x \tag{8.83}$$

$$B_4(x) = \left(x - \frac{1}{2}\right)^4 - \frac{1}{2}\left(x - \frac{1}{2}\right)^2 + \frac{7}{5 \cdot 48} = x^4 - 2x^3 + x^2 - \frac{1}{30}. \tag{8.84}$$

Let us retake the idea of integration by parts that we started in (8.71)

$$\begin{aligned} -\int_0^1 f'(x)B_1(x)dx &= -\frac{1}{2}\int_0^1 f'(x)B'_2(x)dx \\ &= \frac{1}{2}B_2[f'(0) - f'(1)] + \frac{1}{2}\int_0^1 f''(x)B_2(x)dx \end{aligned} \tag{8.85}$$

and

$$\begin{aligned}\frac{1}{2}\int_0^1 f''(x)B_2(x)dx &= \frac{1}{2\cdot 3}\int_0^1 f''(x)B_3'(x)dx \\ &= \frac{1}{2\cdot 3}\left[f''(x)B_3(x)\Big|_0^1 - \int_0^1 f'''(x)B_3(x)dx\right] \\ &= -\frac{1}{2\cdot 3}\int_0^1 f'''(x)B_3(x)dx = -\frac{1}{2\cdot 3\cdot 4}\int_0^1 f'''(x)B_4'(x)dx \\ &= \frac{B_4}{4!}[f'''(0) - f'''(1)] + \frac{1}{4!}\int_0^1 f^{(4)}(x)B_4(x)dx.\end{aligned} \tag{8.86}$$

Continuing this way we arrive at the Euler-Maclaurin formula for the simple trapezoidal rule in $[0, 1]$:

Theorem 8.5.

$$\int_0^1 f(x)dx = \frac{1}{2}[f(0) + f(1)] + \sum_{k=1}^{m}\frac{B_{2k}}{(2k)!}[f^{(2k-1)}(0) - f^{(2k-1)}(1)] + R_m \tag{8.87}$$

where

$$R_m = \frac{1}{(2m+2)!}\int_0^1 f^{(2m+2)}(x)[B_{2m+2}(x) - B_{2m+2}]dx \tag{8.88}$$

and using (8.79), the Mean Value theorem for integrals, and Lemma 3

$$R_m = \frac{1}{(2m+2)!}\int_0^1 f^{(2m+2)}(\eta)[B_{2m+2}(x) - B_{2m+2}]dx = -\frac{B_{2m+2}}{(2m+2)!}f^{(2m+2)}(\eta) \tag{8.89}$$

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 8.6. *(The Euler-Maclaurin Summation Formula) Let m be a positive integer and $f \in C^{(2m+2)}[a,b]$, $h = \frac{b-a}{N}$ then*

$$\begin{aligned}\int_a^b f(x)dx = h\left[\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{j=1}^{N-1} f(a+jh)\right] \\ + \sum_{k=1}^{m} \frac{B_{2k}}{(2k)!}h^{2k}[f^{2k-1}(a) - f^{2k-1}(b)] \\ - \frac{B_{2m+2}}{(2m+2)!}(b-a)h^{2m+2}f^{(2m+2)}(\eta). \qquad \eta \in (0,1)\end{aligned} \tag{8.90}$$

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 $h \to 0$.

8.9 Romberg Integration

We are now going to apply successively Richardson's Extrapolation to the trapezoidal rule. Again, we consider equally spaced nodes, $x_j = a + jh$, $j = 0, 1, \ldots, N$, $h = (b-a)/N$, and assume N is even

$$T_h[f] = h\left[\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{j=1}^{n-1} f(a+jh)\right] := h\sum_{j=0}^{N}{}'' f(a+jh) \tag{8.91}$$

where $\sum''$ means that first and last terms have a $\frac{1}{2}$ factor.

We know from the Euler-Maclaurin formula that for a smooth integrand

$$\int_a^b f(x)dx = T_h[f] + c_2h^2 + c_4h^4 + \cdots \tag{8.92}$$

for some constants c_2, c_4, etc. We can do Richardson extrapolation to obtain a quadrature with a leading order error $O(h^4)$. If we have computed $T_{2h}[f]$ we can combine it with $T_h[f]$ to achieve this by noting that

$$\int_a^b f(x)dx = T_{2h}[f] + c_2(2h)^2 + c_4(2h)^4 + \cdots \tag{8.93}$$

we have

$$\int_a^b f(x)dx = \frac{4T_h[f] - T_{2h}[f]}{3} + \tilde{c_4}h^4 + \tilde{c_6}h^6 + \cdots \tag{8.94}$$

We can continue the Richardson extrapolation process but we can do this more efficiently if we reuse the work we have done to compute $T_{2h}[f]$ to evaluate $T_h[f]$. To this end, we note that

$$T_h[f] - \frac{1}{2}T_{2h}[f] = h\sum_{j=0}^{N}{}'' f(a+jh) - h\sum_{j=0}^{\frac{N}{2}}{}'' f(a+2jh) = h\sum_{j=1}^{\frac{N}{2}} f(a+(2j-1)h)$$

If we let $h_l = \frac{b-a}{2^l}$ then

$$T_{h_l} = \frac{1}{2}T_{h_{l-1}}[f] + h_l\sum_{j=1}^{2^{l-1}} f(a+(2j-1)h_l). \tag{8.95}$$

Beginning with the simple trapezoidal rule (two points) we can successively double the number of points in the quadrature by using (8.95) and immediately do extrapolation.

Let

$$R(0,0) = T_{h_0}[f] = \frac{b-a}{2}\left[f(a)+f(b)\right] \tag{8.96}$$

and for $l = 1, 2, ..., M$ define

$$R(l,0) = \frac{1}{2}R(l-1,0) + h_l\sum_{j=1}^{2^{l-1}} f(a+2j-1)h_l). \tag{8.97}$$

From $R(0,0)$ and $R(1,0)$ we can extrapolate to obtain

$$R(1,1) = R(1,0) + \frac{1}{4-1}[R(1,0) - R(0,0)] \tag{8.98}$$

We can generate a tableau of approximations like the following, for $M = 4$

$$\begin{array}{lllll} R(0,0) & & & & \\ R(1,0) & R(1,1) & & & \\ R(2,0) & R(2,1) & R(2,2) & & \\ R(3,0) & R(3,1) & R(3,2) & R(3,3) & \\ R(4,0) & R(4,1) & R(4,2) & R(4,3) & R(4,4) \end{array}$$

Each of the $R(l,m)$ is obtained by extrapolation

$$R(l,m) = R(l,m-1) + \frac{1}{4^m - 1}[R(l,m-1) - R(l-1,m-1)]. \qquad (8.99)$$

and R(4,4) would be the most accurate approximation (neglecting round off errors). This is the Romberg algorithm and can be written as:

$h = b - a$;
$R(0,0) = \frac{1}{2}(b-a)[f(a) + f(b)]$;
for $l = 1 : M$
    $h = h/2$;
    $R(1,0) = \frac{1}{2}R(l-1,0) + h\sum_{j=1}^{2^{l-1}} f(a + (2j-1)h)$;
    for $m = 1 : M$
        $R(l,m) = R(l,m-1) + \frac{1}{4^m-1}[R(l,m-l) - R(l-1,m-1)]$;
    end
end

Chapter 9

Linear Algebra

9.1 The Three Main Problems

There are three main problems in Numerical Linear Algebra:

1. Solving large linear systems of equations.
2. Finding eigenvalues and eigenvectors.
3. Computing the Singular Value Decomposition (SVD) of a large matrix.

The first problem appears in a wide variety of applications and is 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 on the order of millions or billions, we would like to find the unique solution x, satisfying

$$Ax = b \tag{9.1}$$

or an accurate approximation $\tilde{x}$ to 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 of the solution $x^{(0)}$ produce subsequent approximations $x^{(1)}, x^{(2)}, \ldots$ from a given recipe

$$x^{(k+1)} = G(x^{(k)}, A, b), \quad k = 0, 1, \ldots \tag{9.2}$$

where G is a continuous function of the first variable. Consequently, if the iterations converge, $x^{(k)} \to x$ as $k \to \infty$, to the solution x of the linear system $Ax = b$, then

$$x = G(x, A, b). \tag{9.3}$$

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.

The eigenvalue problem for an $n \times n$ matrix A consists of finding each or some of the scalars (the eigenvalues) λ and the corresponding eigenvectors $v \neq 0$ such that

$$Av = \lambda v. \tag{9.4}$$

Equivalently, $(A - \lambda I)v = 0$ and so the eigenvalues are the roots of the characteristic polynomial of A

$$p(\lambda) = \det(A - \lambda I). \tag{9.5}$$

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 of the eigenvalues of a matrix is useful concept in numerical linear algebra.

Definition 9.1. *Let A be an $n \times n$ matrix. The spectral radius ρ of A is defined as*

$$\rho(A) = \max\{|\lambda_1|, \ldots, |\lambda_n|\}, \tag{9.6}$$

where λ_i, $i = 1, \ldots, n$ are the eigenvalues (not necessarily distinct) of A.

Large eigenvalue (or more appropriately eigenvector) problems arise in the study of the 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 original Google's PageRank search algorithm is a prominent example of the latter. 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.

The third problem is related to the second one 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 $A^T A$ (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)

$$U^T A V = \Sigma, \tag{9.7}$$

where U is an orthogonal $m \times m$ matrix, V is an orthogonal $n \times n$ matrix, and Σ is a "diagonal" matrix of the form

$$\Sigma = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}, \qquad D = diag(\sigma_1, \sigma_2, \ldots, \sigma_r), \tag{9.8}$$

where $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_r > 0$ are the nonzero singular values of A.

9.2 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

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}. \tag{9.9}$$

The canonical vectors, corresponding to the standard basis in $\mathbb{R}^n$, will be denoted by $e_1, e_2, \ldots, e_n$, where e_k is the n-vector with all entries equal to zero except the j-th one, which is equal to one.

The inner product of two real vectors x and y in $\mathbb{R}^n$ is

$$\langle x, y\rangle = \sum_{i=1}^{n} x_i y_i = x^T y. \tag{9.10}$$

If the vectors are complex, i.e. x and y in $\mathbb{C}^n$ we define their inner product as

$$\langle x, y\rangle = \sum_{i=1}^{n} \bar{x}_i y_i, \tag{9.11}$$

where $\bar{x}_i$ denotes the complex conjugate of x_i.

With the inner product (9.10) in the real case or (9.11) in the complex case, we can define the Euclidean norm

$$\|x\|_2 = \sqrt{\langle x, x\rangle}. \tag{9.12}$$

Note that if A is an $n \times n$ real matrix and $x, y \in \mathbb{R}^n$ then

$$\begin{aligned} \langle x, Ay\rangle &= \sum_{i=1}^{n} x_i \left(\sum_{k=1}^{n} a_{ik} y_k \right) = \sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik} x_i y_k \\ &= \sum_{k=1}^{n} \left(\sum_{i=1}^{n} a_{ik} x_i \right) y_k = \sum_{k=1}^{n} \left(\sum_{i=1}^{n} a_{ki}^T x_i \right) y_k, \end{aligned} \tag{9.13}$$

that is

$$\langle x, Ay\rangle = \langle A^T x, y\rangle. \tag{9.14}$$

Similarly in the complex case we have

$$\langle x, Ay\rangle = \langle A^* x, y\rangle, \tag{9.15}$$

where A^* is the conjugate transpose of A, i.e. $A^* = (\overline{a_{ji}})$.

9.3 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

$$\langle Ax, Ax\rangle = \langle A^T Ax, x\rangle = \langle x, x\rangle \tag{9.16}$$

and consequently $A^T A = I$.

Definition 9.2. *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 9.3. *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

$$\langle v, Av\rangle = \langle v, \lambda v\rangle = \lambda \langle v, v\rangle = \lambda. \tag{9.17}$$

But if $A^T = A$ then

$$\lambda = \langle v, Av\rangle = \langle Av, v\rangle = \langle \lambda v, v\rangle = \overline{\lambda}\langle v, v\rangle = \overline{\lambda}, \tag{9.18}$$

and $\lambda = \overline{\lambda}$ if and only if $\lambda \in \mathbb{R}$.

Definition 9.4. *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

$$\langle x, Ax\rangle = \sum_{i=1}^{n} \lambda_i y_i^2, \tag{9.19}$$

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 9.5. *Let $A = (a_{ij})$ be an $n \times n$ matrix. Its leading principal submatrices are the square matrices*

$$A_k = \begin{bmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & & \\ a_{k1} & \cdots & a_{kk} \end{bmatrix}, \quad k = 1, \ldots, n. \tag{9.20}$$

Theorem 9.1. *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

$$x = \begin{bmatrix} y_1 \\ \vdots \\ y_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{9.21}$$

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. □

The converse of Theorem 9.1 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 $j = 1, \ldots, n$.

9.4 Schur Theorem

Theorem 9.2. *(Schur) Let A be an $n \times n$ matrix, then there exists a unitary matrix T ($T^*T = I$) such that*

$$T^*AT = \begin{bmatrix} \lambda_1 & b_{12} & b_{13} & \cdots & b_{1n} \\ & \lambda_2 & b_{23} & \cdots & b_{2n} \\ & & \ddots & & \vdots \\ & & & & b_{n-1,n} \\ & & & & \lambda_n \end{bmatrix}, \tag{9.22}$$

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

$$T^*AT = \begin{bmatrix} u^* \\ v^* \end{bmatrix} \begin{bmatrix} \lambda_1 u & Av \end{bmatrix} = \begin{bmatrix} \lambda_1 & u^*Av \\ 0 & v^*Av \end{bmatrix}. \tag{9.23}$$

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,

$$T_1^*AT_1 = \begin{bmatrix} \lambda_1 & c_{12} & c_{13} & \cdots & c_{1n} \\ 0 & & & & \\ \vdots & & A_k & & \\ & & & & \\ 0 & & & & \end{bmatrix}, \tag{9.24}$$

where A_k is a $k \times k$ matrix. Now, the eigenvalues of the matrix on the right hand side of (9.24) 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

$$T_{k+1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & & & & \\ \vdots & & T_k & & \\ & & & & \\ 0 & & & & \end{bmatrix} \tag{9.25}$$

and define $T = T_1 T_{k+1}$. Then

$$T^* A T = T_{k+1}^* T_1^* A T_1 T_{k+1} = T_{k+1}^* (T_1^* A T_1) T_{k+1} \tag{9.26}$$

and using (9.24) and (9.25) we get

$$\begin{aligned} T^* A T &= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & & & & \\ \vdots & & T_k^* & & \\ & & & & \\ 0 & & & & \end{bmatrix} \begin{bmatrix} \lambda_1 & c_{12} & c_{13} & \cdots & c_{1n} \\ 0 & & & & \\ \vdots & & A_k & & \\ & & & & \\ 0 & & & & \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & & & & \\ \vdots & & T_k & & \\ & & & & \\ 0 & & & & \end{bmatrix} \\ &= \begin{bmatrix} \lambda_1 & b_{12} & b_{13} & \cdots & b_{1n} \\ & \lambda_2 & b_{23} & \cdots & b_{2n} \\ & & \ddots & & \vdots \\ & & & & b_{n-1,n} \\ & & & & \lambda_n \end{bmatrix}. \end{aligned}$$

□

9.5 Norms

A norm on a vector space V (for example $\mathbb{R}^n$ or $\mathbb{C}^n$) over $K = \mathbb{R}$ (or $\mathbb{C}$) 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) $\|\lambda x\| = |\lambda| \, \|x\| \; \forall x \in V, \, \lambda \in K.$

Example 9.1.

$$\|x\|_1 = |x_1| + \ldots + |x_n|, \tag{9.27}$$

$$\|x\|_2 = \sqrt{\langle x, x \rangle >} = \sqrt{|x_1|^2 + \ldots + |x_n|^2}, \tag{9.28}$$

$$\|x\|_\infty = \max\{|x_1|, \ldots, |x_n|\}. \tag{9.29}$$

Lemma 4. *Let $\| \cdot \|$ be a norm on a vector space V then*

$$| \, \|x\| - \|y\| \, | \leq \|x - y\|. \tag{9.30}$$

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

$$\|x\| - \|y\| \leq \|x - y\|. \tag{9.31}$$

By reversing the roles of x and y we also get

$$\|y\| - \|x\| \leq \|x - y\|. \tag{9.32}$$

□

We will also need 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

$$\|A\| = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|^2}. \tag{9.33}$$

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

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}, \tag{9.34}$$

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 9.6. *The matrix norm defined by (11.1) is called the subordinate or natural norm induced by the vector norm* $\|\cdot\|$.

Example 9.2.

$$\|A\|_1 = \max_{x\neq 0} \frac{\|Ax\|_1}{\|x\|_1}, \tag{9.35}$$

$$\|A\|_\infty = \max_{x\neq 0} \frac{\|Ax\|_\infty}{\|x\|_\infty}, \tag{9.36}$$

$$\|A\|_2 = \max_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2}. \tag{9.37}$$

Theorem 9.3. *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 (11.1) implies

$$\frac{\|Ax\|}{\|x\|} \leq \|A\| \tag{9.38}$$

that is $\|Ax\| \leq \|A\|\|x\|$.
(b) Take $x \neq 0$. By (a) $\|ABx\| \leq \|A\|\|Bx\| \leq \|A\|\|B\|\|x\|$ and thus

$$\frac{\|ABx\|}{\|x\|} \leq \|A\|\|B\|. \tag{9.39}$$

Taking the max it we get that $\|AB\| \leq \|A\|\|B\|$. □

The following theorem offers a more concrete way to compute the matrix norms (9.35)-(9.37).

Theorem 9.4. *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) $\|A\|_2 = \sqrt{\rho(A^T A)}$,

where $\rho(A^T A)$ *is the spectral radius of* $A^T A$*, as defined in (9.6).*

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

$$\sum_{i=1}^{n} |a_{ij^*}| = \max_j \sum_{i=1}^{n} |a_{ij}| \tag{9.40}$$

and take x to be given by $x_i = 0$ for $i \neq j^*$ and $x_{j^*} = 1$. Then, $\|x\|_1 = 1$ and

$$\|Ax\|_1 = \sum_{i=1}^{n} \left| \sum_{j=1}^{n} a_{ij} x_j \right| = \sum_{i=1}^{n} |a_{ij^*}| = \max_j \sum_{i=1}^{n} |a_{ij}|. \tag{9.41}$$

(b) Analogously to (a) we have

$$\|Ax\|_\infty = \max_i \left| \sum_{j=1}^{n} a_{ij} x_j \right| \leq \left(\max_i \sum_{j=1}^{n} |a_{ij}| \right) \|x\|_\infty. \tag{9.42}$$

Let i^* be the index such that

$$\sum_{j=1}^{n} |a_{i^* j}| = \max_i \sum_{j=1}^{n} |a_{ij}| \tag{9.43}$$

and take x given by

$$x_j = \begin{cases} \frac{a_{i^* j}}{|a_{i^* j}|} & \text{if } a_{i^* j} \neq 0, \\ 1 & \text{if } a_{i^* j} = 0. \end{cases} \tag{9.44}$$

Then, $|x_j| = 1$ for all j and $\|x\|_\infty = 1$. Hence

$$\|Ax\|_\infty = \max_i \left| \sum_{j=1}^n a_{ij} x_j \right| = \sum_{j=1}^n |a_{i^* j}| = \max_i \sum_{i=1}^n |a_{ij}|. \tag{9.45}$$

(c) By definition

$$\|A\|_2^2 = \max_{x \neq 0} \frac{\|Ax\|_2^2}{\|x\|_2^2} = \max_{x \neq 0} \frac{x^T A^T A x}{x^T x} \tag{9.46}$$

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

$$\frac{x^T A^T A x}{x^T x} = \frac{y^T D y}{y^T y} = \frac{\lambda_1 y_1^2 + \cdots + \lambda_n y_n^2}{y_1^2 + \cdots + y_n^2} \leq \lambda_n. \tag{9.47}$$

Now take the vector y such that $y_j = 0$ for $j \neq n$ and $y_n = 1$ and the equality holds. Thus,

$$\|A\|_2 = \sqrt{\max_{x \neq 0} \frac{\|Ax\|_2^2}{\|x\|_2^2}} = \sqrt{\lambda_n} = \sqrt{\rho(A^T A)}. \tag{9.48}$$

□

Note that if $A^T = A$ then

$$\|A\|_2 = \sqrt{\rho(A^T A)} = \sqrt{\rho(A^2)} = \rho(A). \tag{9.49}$$

Let λ be an eigenvalue of the matrix A with eigenvector x, normalized so that $\|x\| = 1$. Then,

$$|\lambda| = |\lambda| \|x\| = \|\lambda x\| = \|Ax\| \leq \|A\| \|x\| = \|A\| \tag{9.50}$$

for any matrix norm with the property $\|Ax\| \leq \|A\| \|x\|$. Thus,

$$\rho(A) \leq \|A\| \tag{9.51}$$

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 9.5. *Let A be an $n \times n$ matrix. Given $\epsilon > 0$ there is at least one induced matrix norm $\|\cdot\|$ such that*

$$\rho(A) \le \|A\| \le \rho(A) + \epsilon. \tag{9.52}$$

Proof. By Schur's Theorem, there is a unitary matrix T such that

$$T^*AT = \begin{bmatrix} \lambda_1 & b_{12} & b_{13} & \cdots & b_{1n} \\ & \lambda_2 & b_{23} & \cdots & b_{2n} \\ & & \ddots & & \vdots \\ & & & & b_{n-1,n} \\ & & & & \lambda_n \end{bmatrix} = U, \tag{9.53}$$

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

$$D_\delta^{-1} U D_\delta = \begin{bmatrix} \lambda_1 & \delta b_{12} & \delta^2 b_{13} & \cdots & \delta^{n-1} b_{1n} \\ & \lambda_2 & \delta b_{23} & \cdots & \delta^{n-2} b_{2n} \\ & & \ddots & & \vdots \\ & & & & \delta b_{n-1,n} \\ & & & & \lambda_n \end{bmatrix}. \tag{9.54}$$

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:

$$\sum_{j=i+1}^{n} \left| \delta^{j-i} b_{ij} \right| \le \epsilon \qquad \text{for } i = 1, \ldots, n. \tag{9.55}$$

Now,

$$D_\delta^{-1} U D_\delta = D_\delta^{-1} T^* A T D_\delta = (T D_\delta)^{-1} A (T D_\delta) \tag{9.56}$$

Given a nonsingular matrix S and a matrix norm $\|\cdot\|$ then

$$\|A\|' = \|S^{-1} A S\| \tag{9.57}$$

is also a norm. Taking $S = TD_\delta$ and using the infinity norm we get

$$\|A\|' = \|(TD_\delta)^{-1}A(TD_\delta)\|_\infty$$

$$\leq \left\| \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \right\|_\infty + \left\| \begin{bmatrix} 0 & \delta b_{12} & \delta^2 b_{13} & \cdots & \delta^{n-1} b_{1n} \\ & 0 & \delta b_{23} & \cdots & \delta^{n-2} b_{2n} \\ & & \ddots & & \vdots \\ & & & & \delta b_{n-1,n} \\ & & & & 0 \end{bmatrix} \right\|_\infty$$

$$\leq \rho(A) + \epsilon.$$

□

9.6 Condition Number of a Matrix

Consider the 5×5 Hilbert matrix

$$H_5 = \begin{bmatrix} \frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} \end{bmatrix} \tag{9.58}$$

and the linear system $H_5 x = b$ where

$$b = \begin{bmatrix} 137/60 \\ 87/60 \\ 153/140 \\ 743/840 \\ 1879/2520 \end{bmatrix}. \tag{9.59}$$

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)

$$b + \delta b = \begin{bmatrix} 2.28 \\ 1.46 \\ 1.10 \\ 0.89 \\ 0.75 \end{bmatrix} \tag{9.60}$$

The solution of the perturbed system (up to rounding at 12 digits of accuracy) is

$$x + \delta x = \begin{bmatrix} 0.5 \\ 7.2 \\ -21.0 \\ 30.8 \\ -12.6 \end{bmatrix}. \tag{9.61}$$

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

$$\|\delta x\| \leq \|A^{-1}\|\|\delta b\| \tag{9.62}$$

for any induced norm. But also $\|b\| = \|Ax\| \leq \|A\|\|x\|$ or

$$\frac{1}{\|x\|} \leq \|A\|\frac{1}{\|b\|}. \tag{9.63}$$

Combining (9.62) and (9.63) we obtain

$$\frac{\|\delta x\|}{\|x\|} \leq \|A\|\|A^{-1}\|\frac{\|\delta b\|}{\|b\|}. \tag{9.64}$$

The right hand side of this inequality is actually a least upper bound, there are b and δb for which the equality holds.

Definition 9.7. *Given a matrix norm $\|\cdot\|$, the* **condition number** *of a matrix A, denoted by $\kappa(A)$ is defined by*

$$\kappa(A) = \|A\|\|A^{-1}\|. \tag{9.65}$$

Example 9.3. *The condition number of the 5×5 Hilbert matrix H_5, (9.58), 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

$$Ax + A\delta x + \delta A(x + \delta x) = b \tag{9.66}$$

that is, $A\delta x = -\delta A(x + \delta x)$, which implies that

$$\|\delta x\| \leq \|A^{-1}\|\|\delta A\|\|x + \delta x\| \tag{9.67}$$

for any induced matrix norm and consequently

$$\frac{\|\delta x\|}{\|x + \delta x\|} \leq \|A^{-1}\|\|A\|\frac{\|\delta A\|}{\|A\|} = \kappa(A)\frac{\|\delta A\|}{\|A\|}. \tag{9.68}$$

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 9.4. *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 $\kappa(H_6) = 1.4951 \times 10^7$, $\kappa(H_{10}) = 1.6025 \times 10^{13}$.*

9.6.1 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 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 by for example by premultiplying by a matrix P^{-1} such that we have $P^{-1}Ax = P^{-1}b$. Obviously, taking $P = A$ gives us the smallest possible condition number but this choice is not practical so a compromise is made between P approximating A and the cost of solving linear systems with the matrix P to be low. This very useful technique, also employed to accelerate the convergence of some iterative methods, is called **preconditioning**.

Chapter 10

Linear Systems of Equations I

In this chapter we focus on a problem which is central to many applications: find the solution to a large linear system of n linear equations in n unknowns $x_1, x_2, \ldots, x_n$

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2, \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n, \end{aligned} \tag{10.1}$$

or written in matrix form

$$Ax = b \tag{10.2}$$

where A is the $n \times n$ matrix of coefficients

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \tag{10.3}$$

x is a column vector whose components are the unknowns, and b is the given right hand side of the linear system

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}. \tag{10.4}$$

We will assume, unless stated otherwise, that A is a nonsingular, real matrix. That is, the linear system (10.2) 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.

10.1 Easy to Solve Systems

When A **is diagonal**, i.e.

$$A = \begin{bmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{bmatrix} \tag{10.5}$$

(all the entries outside the diagonal are zero and since A is assumed nonsingular $a_{ii} \neq 0$ for all i), then each equation can be solved with just one division:

$$x_i = b_i/a_{ii}, \qquad \text{for } i = 1, 2, \ldots, n. \tag{10.6}$$

If A **is lower triangular** and nonsingular,

$$A = \begin{bmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ \vdots & \vdots & \ddots & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \tag{10.7}$$

the solution can also be obtained easily by the process of *forward substitution*:

$$\begin{aligned} x_1 &= \frac{b_1}{a_{11}} \\ x_2 &= \frac{b_2 - a_{21}x_1}{a_{22}} \\ x_3 &= \frac{b_3 - [a_{31}x_1 + a_{32}x_2]}{a_{33}} \vdots \\ x_n &= \frac{b_n - [a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{n,n-1}x_{n-1}]}{a_{nn}}, \end{aligned} \tag{10.8}$$

or in pseudo-code:

Algorithm 10.1 Forward Subsitution

1: **for** $i = 1, \ldots, n$ **do**

2: $\quad x_i \leftarrow \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j \right) / a_{ii}$

3: **end for**

Note that the assumption that A is nonsingular implies that $a_{ii} \neq 0$ for all $i = 1, 2, \ldots, n$ since $\det(A) = a_{11} a_{22} \cdots a_{nn}$. Also observe that (10.8) shows that 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 $W(n)$ to do forward substitution is

$$W(n) = 2 \sum_{i=1}^{n} (i-1) + n = n^2 - 2n, \tag{10.9}$$

where we have used that

$$\sum_{i=1}^{n} i = \frac{n(n-1)}{2}. \tag{10.10}$$

That is, $W(n) = O(n^2)$ to solve a lower triangular linear system.

If A is nonsingular and upper triangular

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix} \tag{10.11}$$

we solve the linear system $Ax = b$ starting from x_n, then we solve for x_{n-1},

etc. This is called *backward substitution*

$$\begin{aligned} x_n &= \frac{b_n}{a_{nn}}, \\ x_{n-1} &= \frac{b_{n-1} - a_{n-1,n}x_n}{a_{n-1,n-1}}, \\ x_{n-2} &= \frac{b_{n-2} - [a_{n-2,n-1}x_{n-1} + a_{n-2,n}x_n]}{a_{n-2,n-2}}, \\ &\vdots \\ x_1 &= \frac{b_1 - [a_{12}x_2 + a_{13}x_3 + \cdots a_{1n}x_n]}{a_{11}}. \end{aligned} \tag{10.12}$$

From this 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. In pseudo-code, we have

Algorithm 10.2 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, $W(n) = O(n^2)$.

10.2 Gaussian Elimination

The central idea of Gaussian elimination is to reduce the linear system $Ax = b$ to an equivalent upper triangular system, which 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:

$$\begin{aligned} x_1 + 2x_2 - x_3 + x_4 &= 0, \\ 2x_1 + 4x_2 - x_4 &= -3, \\ 3x_1 + x_2 - x_3 + x_4 &= 3, \\ x_1 - x_2 + 2x_3 + x_4 &= 3. \end{aligned} \tag{10.13}$$

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:

$$A_b = \begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 2 & 4 & 0 & -1 & -3 \\ 3 & 1 & -1 & 1 & 3 \\ 1 & -1 & 2 & 1 & 3 \end{bmatrix}. \tag{10.14}$$

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:

$$\begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 2 & 4 & 0 & -1 & -3 \\ 3 & 1 & -1 & 1 & 3 \\ 1 & -1 & 2 & 1 & 3 \end{bmatrix} \xrightarrow[\substack{R_2 \leftarrow R_2 - 2R_1 \\ R_3 \leftarrow R_3 - 3R1 \\ R_4 \leftarrow R_4 - 1R_1}]{} \begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & -3 & 3 & 0 & 3 \end{bmatrix}, \tag{10.15}$$

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 (10.15) 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

$$\begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & -3 & 3 & 0 & 3 \end{bmatrix} \xrightarrow[R_2 \leftrightarrow R_3]{} \begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & -3 & 3 & 0 & 3 \end{bmatrix}. \tag{10.16}$$

We can now use -5 as a pivot and do the second round of elimination:

$$\begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & -3 & 3 & 0 & 3 \end{bmatrix} \xrightarrow[\substack{R_3 \leftarrow R_3 - 0R_2 \\ R_4 \leftarrow R_4 - \frac{3}{5}R_2}]{} \begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & 0 & \frac{9}{5} & \frac{6}{5} & \frac{6}{5} \end{bmatrix}. \tag{10.17}$$

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

$$\begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & 0 & \frac{9}{5} & \frac{6}{5} & \frac{6}{5} \end{bmatrix} \xrightarrow[R_4 \leftarrow R_4 - \frac{9}{10}R_3]{} \begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & 0 & 0 & \frac{39}{10} & \frac{39}{10} \end{bmatrix}, \tag{10.18}$$

The last matrix, let us call it U_b, corresponds to the upper triangular system

$$\begin{bmatrix} 1 & 2 & -1 & 1 \\ 0 & -5 & 2 & -2 \\ 0 & 0 & 2 & -3 \\ 0 & 0 & 0 & \frac{39}{10} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ -3 \\ \frac{39}{10} \end{bmatrix}, \tag{10.19}$$

which we can solve with backward substitution to obtain the solution

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 1 \end{bmatrix}. \tag{10.20}$$

Each of the steps in the Gaussian elimination process are linear transformations and hence we can represent these transformations 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 (10.15) is equivalent to multiplying (from the left) A_b by the lower triangular matrix

$$E_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ -3 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}, \tag{10.21}$$

that is

$$E_1 A_b = \begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & -3 & 3 & 0 & 3 \end{bmatrix}. \tag{10.22}$$

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 a *permutation* matrix

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{10.23}$$

which is obtained by exchanging the second and third rows in the 4×4 identity matrix,

$$PE_1A_b = \begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & -3 & 3 & 0 & 3 \end{bmatrix}. \tag{10.24}$$

To construct the matrix associated with the second round of elimination we have to take 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:

$$E_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -\frac{3}{5} & 0 & 1 \end{bmatrix}, \tag{10.25}$$

and we get

$$E_2PE_1A_b = \begin{bmatrix} 1 & 2 & -1 & 1 & 0 \\ 0 & -5 & 2 & -2 & 3 \\ 0 & 0 & 2 & -3 & -3 \\ 0 & 0 & \frac{9}{5} & \frac{6}{5} & \frac{6}{5} \end{bmatrix}. \tag{10.26}$$

Finally, for the last elimination we have

$$E_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{9}{10} & 1 \end{bmatrix}, \tag{10.27}$$

and $E_3E_2PE_1A_b = U_b$.

Observe that $PE_1A_b = E_1'PA_b$, where

$$E_1' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ -2 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}, \tag{10.28}$$

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

$$E_3E_2E_1'PA = U, \tag{10.29}$$

where U is the upper triangular matrix

$$U = \begin{bmatrix} 1 & 2 & -1 & 1 \\ 0 & -5 & 2 & -2 \\ 0 & 0 & 2 & -3 \\ 0 & 0 & 0 & \frac{39}{10} \end{bmatrix}. \tag{10.30}$$

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

$$PA = LU, \tag{10.31}$$

where $L = (E_3E_2E_1')^{-1} = E_1'^{-1}E_2^{-1}E_3^{-1}$ 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

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 1 & \frac{3}{5} & \frac{9}{10} & 1 \end{bmatrix}. \tag{10.32}$$

Note that L has all the multipliers below the diagonal and U has all the pivots on the diagonal. We will see that a factorization $PA = LU$ is always possible for any nonsingular $n \times n$ matrix A and can be very useful.

We now consider the general linear system (10.1). The matrix of coefficients and the right hand size are

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}, \tag{10.33}$$

respectively. We form the augmented matrix A_b by appending b to A as the last column:

$$A_b = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{bmatrix}. \tag{10.34}$$

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 possible to then 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

$$A_b^{(1)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & b_1' \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{bmatrix}. \tag{10.35}$$

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 to obtain the elimination of the entries below the first element in the first column. The same three steps above can now be applied to the smaller $(n-1) \times n$ matrix

$$\tilde{A}_b^{(1)} = \begin{bmatrix} a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{bmatrix}, \tag{10.36}$$

and so on. Doing 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 correspond to $A_b^{(k)} = E_k P_k A_b^{(k-1)}$, for $k = 1, 2, \ldots, n-1$ ($A_b^{(0)} = A_b$), where the 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

$$U_b \equiv A_b^{(n-1)} = E_{n-1} P_{n-1} E_{n-2} P_{n-2} \cdots E_1 P_1 A_b. \tag{10.37}$$

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

$$U_b \equiv A_b^{(n-1)} = E'_{n-1} E'_{n-2} \cdots E'_1 P A_b. \tag{10.38}$$

Since the inverse of $E'_{n-1}E'_{n-2}\cdots E'_1$ is the lower triangular matrix

$$L = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{n,n-1} & 1 \end{bmatrix}, \tag{10.39}$$

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

$$\det(PA) = \pm \det(A) = \det(LU) = \det(U) = a_{11}^{(1)} a_{22}^{(2)} \cdots a_{nn}^{(n)} \tag{10.40}$$

and so $\det(A)$ is plus or minus 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 with partial pivoting (assuming $a_{i,n+1} = b_i$, $i = 1, \ldots, n$) is presented in Algorithm 10.3.

10.2.1 The Cost of Gaussian Elimination

We now do an operation count of Gaussian elimination to solve an $n \times n$ linear system $Ax = b$.

We focus on the elimination as we already know that the work for the step of 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 number of operations is

$$W(n) = \sum_{j=1}^{n-1} \left[2(n-j)(n-j+1) + (n-j)\right] = \sum_{j=1}^{n-1} \left[2(n-j)^2 + 3(n-j)\right] \tag{10.41}$$

and using (10.10) and

$$\sum_{i=1}^{m} i^2 = \frac{m(m+1)(2m+1)}{6}, \tag{10.42}$$

Algorithm 10.3 Gaussian Elimination with Partial Pivoting

for $j = 1, \ldots, n-1$ **do**
  Find m such that $|a_{mj}| = \max_{j \le i \le n} |a_{ij}|$
  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
  $x_i \leftarrow \left(a_{i,n+1} - \sum_{j=i+1}^{n} a_{ij} x_j \right) / a_{ii}$
end for

we get

$$W(n) = \frac{2}{3} n^3 + O(n^2). \tag{10.43}$$

Thus, Gaussian elimination is computationally rather expensive for large systems of equations.

10.3 LU and Choleski Factorizations

If Gaussian elimination can be performed without row interchanges, then we obtain an LU factorization of A, i.e. $A = LU$. This factorization can be advantageous when solving many linear systems with the same $n \times n$ matrix A but different right hand sides 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

$$Ly = b, \tag{10.44}$$

$$Ux = y. \tag{10.45}$$

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 right hand sides can be done in $O(n^2)$ operations.

When can we obtain the factorization $A = LU$? the following result provides a useful sufficient condition.

Theorem 10.1. *Let A be an $n\times n$ matrix whose leading principal submatrices $A_1, \ldots, A_n$ are all nonsingular. Then, there exists 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 $2, \ldots, k-1$ so that $A^{(k-1)} = E_{k-1} \cdots E_2 E_1 A$, that is

$$\left[\begin{array}{ccc|ccc} a_{11} & \cdots & a_{1k} & \cdots & a_{1n} \\ & \ddots & & \cdots & \\ & & & \cdots & \\ & & a_{kk}^{(k-1)} & \cdots & a_{kn}^{(k-1)} \\ \hline & & \vdots & & \vdots \\ & & a_{nk}^{(k-1)} & \cdots & a_{nn}^{(k-1)} \end{array}\right] = \left[\begin{array}{ccc|c} 1 & & & \\ -m_{21} & \ddots & & \\ \vdots & & & \\ -m_{k1} & & 1 & \\ \hline \vdots & & & \\ -m_{n1} & \cdots & & 1 \end{array}\right] \left[\begin{array}{ccc|ccc} a_{11} & \cdots & a_{1k} & \cdots & a_{1n} \\ \vdots & \ddots & & \cdots & \vdots \\ & & & \cdots & \\ a_{k1} & & a_{kk} & \cdots & a_{kn} \\ \hline \vdots & & \vdots & & \vdots \\ a_{n1} & \cdots & a_{nk} & \cdots & a_{nn} \end{array}\right].$$

The determinant of the boxed $k \times k$ leading principal submatrix on the left is $a_{11}a_{22}^{(2)} \cdots a_{kk}^{(k-1)}$ 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

$$a_{11}a_{22}^{(2)} \cdots a_{kk}^{(k-1)} = \det(A_k) \neq 0, \tag{10.46}$$

which implies that $a_{kk}^{(k-1)} \neq 0$ and so $P_k = I$ and 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

$$L_2^{-1}L_1 = U_2U_1^{-1}. \tag{10.47}$$

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 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 non-singular in both cases.

Corollary 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 to obtain a symmetric factorization $A = BB^T$, where B is a lower triangular matrix with positive entries in its diagonal. This representation is called Choleski factorization of the symmetric positive definite matrix A.

Theorem 10.2. *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 1 A has an LU factorization. Moreover, from (10.46) 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 $D = diag(\sqrt{u_{11}}, \ldots, \sqrt{u_{nn}})$ 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 $C^TB^T = BC$, which implies

$$B^{-1}C^T = C(B^T)^{-1}. \tag{10.48}$$

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 Choleski factorization is unique we go back to the LU factorization, which we now 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 $A = BD_B^{-1}D_BB^T$, where $D_B = diag(b_{11}, \ldots, b_{nn})$. Then $L = BD_B^{-1}$ and $U = D_BB^T$ 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

$$L = BD_B^{-1} = CD_C^{-1}, \tag{10.49}$$

$$U = D_BB^T = D_CC^T, \tag{10.50}$$

where $D_C = diag(c_{11}, \ldots, c_{nn})$. Equation (10.50) 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

$$a_{ij} = \sum_{k=1}^{n} l_{ik}l_{jk} = \sum_{k=1}^{\min(i,j)} l_{ik}l_{jk}. \tag{10.51}$$

Now, because $A^T = A$ we only need a_{ij} for $i \leq j$, that is

$$a_{ij} = \sum_{k=1}^{i} l_{ik}l_{jk} \qquad 1 \leq i \leq j \leq n. \tag{10.52}$$

We can solve equations (10.52) 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 (10.52) 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 10.4 gives the pseudo code for the Choleski factorization.

Algorithm 10.4 Choleski factorization

```
for i = 1, ..., n do                          ▷ Compute column i of L for i = 1, ..., n
    l_ii ← sqrt( a_ii − Σ_{k=1}^{i−1} l_ik^2 )
    for j = i + 1, ..., n do
        l_ji ← (a_ij − Σ_{k=1}^{i−1} l_ik l_jk) / l_ii
    end for
end for
```

10.4 Tridiagonal Linear Systems

If the matrix of coefficients A has a triadiagonal structure

$$A = \begin{bmatrix} a_1 & b_1 & & & & \\ c_1 & a_2 & b_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & & b_{n-1} & \\ & & & c_{n-1} & a_n & \end{bmatrix} \tag{10.53}$$

its LU factorization can be computed at an $O(n)$ cost and the corresponding linear system can thus be solved efficiently.

Theorem 10.3. *If A is triadiagonal and all of its leading principal subma-*

trices are nonsingular then

$$\begin{bmatrix} a_1 & b_1 & & & \\ c_1 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & & & b_{n-1} \\ & & & c_{n-1} & a_n \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ l_1 & 1 & & & \\ & \ddots & \ddots & & \\ & & & l_{n-1} & 1 \end{bmatrix} \begin{bmatrix} m_1 & b_1 & & & \\ & m_2 & b_2 & & \\ & & \ddots & \ddots & \\ & & & & b_{n-1} \\ & & & & m_n \end{bmatrix}, \tag{10.54}$$

where

$$m_1 = a, \tag{10.55}$$

$$l_j = c_j/m_j, \quad m_{j+1} = a_{j+1} - l_j b_j, \quad \text{for } j = 1, \ldots, n-1, \tag{10.56}$$

and this factorization is unique.

Proof. By Theorem 10.1 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 (10.54) holds. Equating the matrix product on the right hand side of (10.54), 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 (10.55)-(10.56) follows. Of course, we need the m_j's to be nonzero to use (10.56). 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

$$m_{j+1} m_j = a_{j+1} m_j - b_j c_j, \quad \text{for } j = 1, \ldots, n-1. \tag{10.57}$$

Thus,

$$\det(A_1) = a_1 = m_1, \tag{10.58}$$

$$\det(A_2) = a_2 a_1 - c_1 b_1 = a_2 m_1 - b_1 c_1 = m_1 m_2. \tag{10.59}$$

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

$$\det(A_k) = a_k \det(A_{k-1}) - b_{k-1} c_{k-1} \det(A_{k-2}) \tag{10.60}$$

and using the induction hypothesis and (10.57) it follows that

$$\det(A_k) = m_1 m_2 \cdots m_{k-2} \left[a_k m_{k-1} - b_{k-1} c_{k-1} \right] = m_1 \cdots m_k, \tag{10.61}$$

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. □

Algorithm 10.5 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

10.5 A 1D BVP: Deformation of an Elastic Beam

We saw in Section 5.5 an example of a very large system of equations in connection with the least squares problem for fitting high dimensional data. We now consider another example which 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

$$-u''(x) + c(x)u(x) = f(x), \quad 0 < x < 1, \tag{10.62}$$

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

$$u(0) = u(1) = 0. \tag{10.63}$$

The system (10.62)-(10.63) is called a *boundary value problem* (BVP). That is, we need to find a function u that satisfies the ordinary differential equation (10.62) and the boundary conditions (10.63) for any given, continuous f and c. The condition $c(x) \geq 0$ guarantees existence and uniqueness of 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 end points (boundary). So we choose a positive integer N and define the nodes or grid points

$$x_0 = 0, x_1 = h, x_2 = 2h, \ldots, x_N = Nh, x_{N+1} = 1, \tag{10.64}$$

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,

$$u''(x_j) = \frac{u(x_{j+1}) - 2u(x_j) + u(x_{j-1})}{h^2} + O(h^2). \tag{10.65}$$

Neglecting the $O(h^2)$ error and denoting the approximation of $u(x_j)$ by v_j (i.e. $v_j \approx u(x_j)$) $f_j = f(x_j)$ and $c_j = c(x_j)$, for $j = 1, \ldots, N$, then at each interior node

$$-\frac{v_{j-1} + 2v_j + v_{j+1}}{h^2} + c_j v_j \quad = \quad f_j, \quad j = 1, 2, \ldots, N \tag{10.66}$$

and at the boundary nodes, applying (10.63), we have

$$v_0 = v_{N+1} = 0. \tag{10.67}$$

Thus, (10.66) is a linear system of N equations in N unknowns $v_1, \ldots, v_N$, which we can write in matrix form as

$$\frac{1}{h^2}\begin{bmatrix} 2+c_1h^2 & -1 & 0 & \cdots & & \cdots & 0 \\ -1 & 2+c_2h^2 & -1 & \ddots & & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & 0 \\ & & & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & & & 0 & -1 & 2+c_Nh^2 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ v_N \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ f_N \end{bmatrix}. \tag{10.68}$$

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 $\mathbf{v} = [v_1, \ldots, v_N]^T$

$$\mathbf{v}^T A\mathbf{v} = \sum_{j=1}^{N}\left[\left(\frac{v_j + c_j v_j}{h}\right)^2 + c_j v_j^2\right] > 0, \quad \forall\, \mathbf{v} \neq 0 \tag{10.69}$$

and therefore, since $c_j \geq 0$ for all j, A is positive definite. Thus, there is a unique solution to (10.68) and can be efficiently found with our tridiagonal solver, Algorithm 10.5. Since the expected numerical error is $O(h^2) = O(1/(N+1)^2)$, even a modest accuracy of $O(10^{-4})$ requires $N \approx 100$.

10.6 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

$$-\Delta u(x,y) = f(x,y), \quad \text{for } (x,y) \in \Omega \tag{10.70}$$

and

$$u(x,y) = 0. \quad \text{for } (x,y) \in \partial\Omega. \tag{10.71}$$

In (10.70), Δu is the Laplacian of u, also denoted as $\nabla^2 u$, and is given by

$$\Delta u = \nabla^2 u = u_{xx} + u_{yy} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}. \tag{10.72}$$

Equation (10.70) is Poisson's equation (in 2D) and together with (10.71) specify a (homogeneous) Dirichlet problem because the value of u is given at the boundary.

To construct a numerical approximation to (10.70)-(10.71), 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

$$(x_i, x_j) = (ih, jh), \quad \text{for } i,j = 0, \ldots, N+1, \tag{10.73}$$

where $h = 1/(N+1)$. The interior nodes correspond to $1 \le i,j \le N$ and the boundary nodes are those corresponding to the remaining values of indices i and j (i or j equal 0 and i or j 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*

$$\nabla^2 u(x_i, y_j) = \frac{u(x_{i-1}, x_j) + u(x_{i+1}, y_j) + u(x_i, y_{j-1}) + u(x_i, y_{j+1}) - 4u(x_i, y_j)}{h^2} + O(h^2). \tag{10.74}$$

Neglecting the $O(h^2)$ discretization error and denoting by v_{ij} the approximation to $u(x_i, y_j)$ we get:

$$-\frac{v_{i-1,j} + v_{i+1,j} + v_{i,j-1} + v_{i,j+1} - 4v_{ij}}{h^2} = f_{ij}, \text{ for } 1 \le i,j \le N. \tag{10.75}$$

This is a linear system of N^2 equations for the N^2 unknowns, v_{ij}, $1 \leq i, j \leq N$. 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 ordering of the unknowns!.

The most common labeling is the so-called *lexicographical order*, which proceeds from the bottom row to top one, left to right, $v_{11}, v_{12}, \ldots, v_{1N}, v_{21}, \ldots$, etc. Denoting by $\mathbf{v}_1 = [v_{11}, v_{12}, \ldots, v_{1N}]^T$, $\mathbf{v}_2 = [v_{21}, v_{22}, \ldots, v_{2N}]^T$, etc., and similarly for the right hand side f, the linear system (10.75) can be written in matrix form as

$$\begin{bmatrix} T & -I & 0 & & & & 0 \\ -I & T & -I & \ddots & & & \\ 0 & \ddots & \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & 0 \\ & & & \ddots & \ddots & \ddots & -I \\ 0 & & & & 0 & -I & T \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \mathbf{v}_N \end{bmatrix} = h^2 \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \mathbf{f}_N \end{bmatrix}. \tag{10.76}$$

Here, I is the $N \times N$ identity matrix and T is the $N \times N$ tridiagonal matrix

$$T = \begin{bmatrix} 4 & -1 & 0 & & & & 0 \\ -1 & 4 & -1 & \ddots & & & \\ 0 & \ddots & \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & 0 \\ & & & \ddots & \ddots & \ddots & -1 \\ 0 & & & & 0 & -1 & 4 \end{bmatrix}. \tag{10.77}$$

Thus, the matrix of coefficients in (10.76), 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 ($n > 100$) 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.

10.7 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 non-singular 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

$$Mx = Nx + b. \tag{10.78}$$

Starting with an initial guess $x^{(0)}$, (10.78) defines a sequence of approximations generated by

$$Mx^{(k+1)} = Nx^{(k)} + b, \quad k = 0, 1, \ldots \tag{10.79}$$

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 (10.79). Unless otherwise stated, A is assumed to be a non-singular $n \times n$ matrix and b a given n-column vector.

10.8 Jacobi, Gauss-Seidel, and S.O.R.

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 (10.79) can be easily solved to obtain the next iterate $x^{(k+1)}$. Note that we do not need to compute M^{-1} nor do we need to do 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 just solve the first equation for the first unknown, the second equation for the second unknown, etc., and we obtain the so-called *Jacobi* iterative method:

$$x_i^{(k+1)} = \frac{-\sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k+1)} + b}{a_{ii}}, \ i = 1, 2, ..., n, \text{ and } k = 0, 1, ... \tag{10.80}$$

The iteration could be stopped when

$$\frac{\|x^{(k+1)} - x^{(k)}\|_\infty}{\|x^{(k+1)}\|_\infty} \leq \text{Tolerance}. \tag{10.81}$$

Example 10.1. *Consider the* 4×4 *linear system*

$$\begin{aligned} 10x_1 - x_2 + 2x_3 &= 6, \\ -x_1 + 11x_2 - x_3 + 3x_4 &= 25, \\ 2x_1 - x_2 + 10x_3 - x_4 &= -11, \\ 3x_2 - x_3 + 8x_4 &= 15. \end{aligned} \tag{10.82}$$

It has the unique solution (1,2,-1,1). Jacobi's iteration for this system is

$$\begin{aligned} x_1^{(k+1)} &= \frac{1}{10}x_2^{(k)} - \frac{1}{5}x_3^{(k)} + \frac{3}{5}, \\ x_2^{(k+1)} &= \frac{1}{11}x_1^{(k)} + \frac{1}{11}x_3^{(k)} - \frac{3}{11}x_4^{(k)} + \frac{25}{11}, \\ x_3^{(k+1)} &= -\frac{1}{5}x_1^{(k)} + \frac{1}{10}x_2^{(k)} + \frac{1}{10}x_4^{(k)} - \frac{11}{10}, \\ x_4^{(k+1)} &= -\frac{3}{8}x_2^{(k)} + \frac{1}{8}x_3^{(k)} + \frac{15}{8}. \end{aligned} \tag{10.83}$$

Starting with $x^{(0)} = [0, 0, 0, 0]^T$ *we obtain*

$$x^{(1)} = \begin{bmatrix} 0.60000000 \\ 2.27272727 \\ -1.10000000 \\ 1.87500000 \end{bmatrix}, \quad x^{(2)} = \begin{bmatrix} 1.04727273 \\ 1.71590909 \\ -0.80522727 \\ 0.88522727 \end{bmatrix}, \quad x^{(3)} = \begin{bmatrix} 0.93263636 \\ 2.05330579 \\ -1.04934091 \\ 1.13088068 \end{bmatrix}. \tag{10.84}$$

In the Jacobi 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 the Jacobi iteration with the already computed components of $x^{(k+1)}$ we obtained the *Gauss-Seidel* iteration:

$$x_i^{(k+1)} = \frac{-\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} + b}{a_{ii}}, \quad i = 1, 2, ..., n, \quad k = 0, 1, ... \tag{10.85}$$

The Gauss-Seidel iteration is equivalent to the iteration obtained by taking M as the lower triangular part of the matrix A, including its diagonal.

Example 10.2. *For the system (10.82), starting again with the initial guess* $[0, 0, 0, 0]^T$*, Gauss-Seidel produces the following approximations*

$$x^{(1)} = \begin{bmatrix} 0.60000000 \\ 2.32727273 \\ -0.98727273 \\ 0.87886364 \end{bmatrix}, \quad x^{(2)} = \begin{bmatrix} 1.03018182 \\ 2.03693802 \\ -1.0144562 \\ 0.98434122 \end{bmatrix}, \quad x^{(3)} = \begin{bmatrix} 1.00658504 \\ 2.00355502 \\ -1.00252738 \\ 0.99835095 \end{bmatrix}. \tag{10.86}$$

In an attempt to accelerate convergence of the Gauss-Seidel iteration, one could also put some weight in diagonal part of A, and split this into the matrices M and N of the iterative method (10.79). Specifically, we can write

$$\text{diag}(A) = \frac{1}{\omega}\text{diag}(A) - \frac{1-\omega}{\omega}\text{diag}(A), \tag{10.87}$$

where the first term of the right hand side goes into M and the last into N. The Gauss-Seidel method then becomes

$$x_i^{(k+1)} = \frac{a_{ii}x_i^{(k)} - \omega\left[\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} + \sum_{j=i}^{n} a_{ij}x_j^{(k)} - b\right]}{a_{ii}},\ i = 1, 2, ..., n,\ k = 0, 1, \ldots \tag{10.88}$$

Note that $\omega = 1$ corresponds to Gauss-Seidel. This iteration is generically *S.O.R.* (successive over-relaxation), even though we refer to over-relaxation only when $\omega > 1$ and under-relaxation when $\omega < 1$. It can be proved that a necessary condition for convergence is that $0 < \omega < 2$.

10.9 Convergence of Linear Iterative Methods

To study the convergence of iterative methods of the form $Mx^{(k+1)} = Nx^{(k)} + b$, for $k = 0, 1, \ldots$ we use the equivalent iteration

$$x^{(k+1)} = Tx^{(k)} + c,\ k = 0, 1, \ldots \tag{10.89}$$

where

$$T = M^{-1}N = I - M^{-1}A \tag{10.90}$$

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 (10.89) 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 inducedinduced matrix norm we have

$$\|F(x) - F(y)\| = \|Tx - Ty\| \leq \|T\|\ \|x - y\|. \tag{10.91}$$

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 (10.89) is given by the spectral radius $\rho(T)$ of the iteration matrix T. These conclusions will follow from the following result.

Theorem 10.4. *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

$$\|T^k x\| \leq \|T^k\| \, \|x\| \tag{10.92}$$

and so if $T^k \to 0$ as $k \to \infty$ then $\|T^k x\| \to 0$, 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 9.5, 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 10.5. *The iterative method (10.89) 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

$$x - x^{(1)} = Tx + c - (Tx^{(0)} + c) = T(x - x^{(0)}), \tag{10.93}$$

from which it follows that the error of the k iterate, $e_k = x^{(k)} - x$, satisfies

$$e_k = T^k e_0, \tag{10.94}$$

for $k = 1, 2, \ldots$ and where $e_0 = x - x^{(0)}$ is the error of the initial guess. The conclusion now follows immediately from Theorem 10.4. □

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 (10.94) we get

$$\|e_k\|_2 \leq \rho(T)^k \|e_0\|_2. \tag{10.95}$$

But 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 results shows that, asymptotically $\|T^k\| \approx \rho(T)^k$, for any matrix norm.

Theorem 10.6. *Let T be any $n \times n$ matrix. Then, for any matrix norm $\|\cdot\|$*

$$\lim_{k\to\infty} \|T^k\|^{1/k} = \rho(T). \tag{10.96}$$

Proof. We know that $\rho(T^k) = \rho(T)^k$ and that $\rho(T) \leq \|T\|$. Therefore

$$\rho(T) \leq \|T^k\|^{1/k} \tag{10.97}$$

Now, for any given $\epsilon > 0$ construct the matrix $T_\epsilon = T/(\rho(T) + \epsilon)$. Then $\lim_{k\to\infty} T_\epsilon^k = 0$ as $\rho(T_\epsilon) < 1$. Therefore, there is an integer K_ϵ such that

$$\|T_\epsilon^k\| = \frac{\|T^k\|}{(\rho(T) + \epsilon)^k} \leq 1, \text{ for all } k \geq K_\epsilon. \tag{10.98}$$

Thus, for all $k \geq K_\epsilon$ we have

$$\rho(T) \leq \|T^k\|^{1/k} \leq \rho(T) + \epsilon \tag{10.99}$$

from which the results follows. □

Theorem 10.7. *Let A 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,

$$\|T\|_\infty = \max_{1\leq i\leq n} \sum_{\substack{j=1\\ j\neq i}}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| = \max_{1\leq i\leq n} \frac{1}{|a_{ii}|} \sum_{j=1, j\neq i}^{n} |a_{ij}| < 1. \tag{10.100}$$

(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 $T = I - M^{-1}A$. Then, $Tx = \lambda x$ implies $Mx - Ax = \lambda Mx$ from which we get

$$-\sum_{j=i+1}^{n} a_{ij}x_j = \lambda \sum_{j=1}^{i} a_{ij}x_j = \lambda a_{ii}x_i + \lambda \sum_{j=1}^{i-1} a_{ij}x_j. \tag{10.101}$$

Now choose i such that $\|x\|_\infty = |x_i| = 1$ then

$$|\lambda|\,|a_{ii}| \leq |\lambda| \sum_{j=1}^{i-1} |a_{ij}| + \sum_{j=i+1}^{n} |a_{ij}|$$

$$|\lambda| \leq \frac{\displaystyle\sum_{j=i+1}^{n} |a_{ij}|}{|a_{ii}| - \displaystyle\sum_{j=1}^{i-1} |a_{ij}|} < \frac{\displaystyle\sum_{j=i+1}^{n} |a_{ij}|}{\displaystyle\sum_{j=i+1}^{n} |a_{ij}|} = 1.$$

where the last inequality was obtained by using that A is SDD. Thus, $|\lambda| < 1$ and so $\rho(T) < 1$. □

Theorem 10.8. *A necessary condition for the S.O.R. 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

$$\rho(T) \geq |1 - \omega|. \tag{10.102}$$

Since $\rho(T) < 1$ is required for convergence, the conclusion follows. Now, $T = M^{-1}[M - A]$ and $\det(T) = \det(M^{-1})\det(M - A)$. From the definition of the S.O.R. iteration (10.88) we get that

$$a_{ii}x_i^{(k+1)} + \omega\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} = a_{ii}x_i^{(k)} - \omega\sum_{j=i}^{n} a_{ij}x_j^{(k)} + \omega b. \tag{10.103}$$

Therefore, M is lower triangular with a diagonal equal to that of A. Consequently, $\det(M^{-1}) = \det(\text{diag}(A)^{-1})$. Similarly, $\det(M - A) = \det((1 - \omega)\text{diag}(A))$. Thus,

$$\begin{aligned}\det(T) &= \det(M^{-1})\det(M - A) = \det(\text{diag}(A)^{-1})\det((1 - \omega)\text{diag}(A)) \\ &= \det(\text{diag}(A)^{-1}(1 - \omega)\text{diag}(A)) = \det((1 - \omega)I) = (1 - \omega)^n.\end{aligned} \tag{10.104}$$

□

If A is positive definite S.O.R. converges for any initial guess. However, as we will see, there are more efficient iterative methods for positive definite linear systems.

Chapter 11

Linear Systems of Equations II

In this chapter we focus on some numerical methods for the solution of large linear systems $Ax = b$ where A is a *sparse*, symmetric positive definite matrix. We also look briefly at the non-symmetric case.

11.1 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 $Ax = b$. Let $\bar{x}$ be the unique, exact solution of $Ax = b$. Since A is positive definite, we can define the norm

$$\|x\|_A = \sqrt{x^T A x}. \tag{11.1}$$

Henceforth we are going to denote the inner product of two vector x, y in $\mathbb{R}^n$ by $\langle x, y \rangle$, i.e

$$\langle x, y \rangle = x^T y = \sum_{i=1}^{n} x_i y_i. \tag{11.2}$$

Consider now the quadratic function of $x \in \mathbb{R}^n$ defined by

$$J(x) = \frac{1}{2}\|x - \bar{x}\|_A^2. \tag{11.3}$$

Note that $J(x) \geq 0$ and $J(x) = 0$ if and only if $x = \bar{x}$ because A is positive definite. Therefore, x minimizes J if and only if $x = \bar{x}$. 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 J along a fixed direction. For given $x, v \in \mathbb{R}^n$ we consider the so called *line minimization* problem consisting in minimizing J along the line that passes through x and is in the direction of v, i.e

$$\min_{t\in\mathbb{R}} J(x+tv). \tag{11.4}$$

Denoting $g(t) = J(x+tv)$ and using the definition (11.3) of J we get

$$\begin{aligned} g(t) &= \frac{1}{2}\langle x-\bar{x}+tv, A(x-\bar{x}+tv)\rangle \\ &= J(x) + \langle x-\bar{x}, Av\rangle\, t + \frac{1}{2}\langle v, Av\rangle\, t^2 \\ &= J(x) + \langle Ax-b, v\rangle\, t + \frac{1}{2}\langle v, Av\rangle\, t^2. \end{aligned} \tag{11.5}$$

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

$$0 = g'(t^*) = -\langle v, b-Ax\rangle + t^*\langle v, Av\rangle, \tag{11.6}$$

that is

$$t^* = \frac{\langle v, b-Ax\rangle}{\langle v, Av\rangle} \tag{11.7}$$

and the minimum of J along the line $x+tv$, $t \in \mathbb{R}$ is

$$g(t^*) = J(x) - \frac{1}{2}\frac{\langle v, b-Ax\rangle^2}{\langle v, Av\rangle}. \tag{11.8}$$

Finally, using the definition of $\|\cdot\|_A$ and $A\bar{x} = b$, we have

$$\frac{1}{2}\|x-\bar{x}\|_A^2 = \frac{1}{2}\|x\|_A^2 - \langle b, x\rangle + \frac{1}{2}\|\bar{x}\|_A^2 \tag{11.9}$$

and so it follows that

$$\nabla J(x) = Ax - b. \tag{11.10}$$

11.2 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 $J(x) = \|x - \bar{x}\|_A^2$. 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, in the form

$$x^{(k+1)} = x^{(k)} + t_k v^{(k)}, \quad k = 0, 1, \ldots, \tag{11.11}$$

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. The question then is how to select the search directions and the step lengths to converge to the minimizer. Most line search methods are of *descent* type because they required that the value of J is decreased with each iteration. Going back to (11.5) this means that descent line search methods have the condition $\langle v^{(k)}, \nabla J(x^{(k)})\rangle < 0$.

Starting with an initial guess $x^{(0)}$, line search methods generate

$$x^{(1)} = x^{(0)} + t_0 v^{(0)} \tag{11.12}$$

$$x^{(2)} = x^{(1)} + t_1 v^{(0)} = x^{(0)} + t_0 v^{(0)} + t_1 v^{(1)}, \tag{11.13}$$

etc., so that the k-th element of the sequence is $x^{(0)}$ plus a linear combination of $v^{(0)}, v^{(1)}, \ldots, v^{(k-1)}$:

$$x^{(k)} = x^{(0)} + t_0 v^{(0)} + t_1 v^{(0)} + \cdots + t_{k-1} v^{(k-1)}. \tag{11.14}$$

That is,

$$\left(x^{(k)} - x^{(0)}\right) \in \text{span}\{v^{(0)}, v^{(1)}, \ldots, v^{(k-1)}\}. \tag{11.15}$$

Unless otherwise noted, we will take the step length t_k to be given by the one-dimensional minimizer (11.7) evaluated at the k-step, i.e.

$$t_k = \frac{\langle v^{(k)}, r^{(k)}\rangle}{\langle v^{(k)}, Av^{(k)}\rangle}, \tag{11.16}$$

where

$$r^{(k)} = b - Ax^{(k)} \tag{11.17}$$

is the *residual* of the linear equation $Ax = b$ associated with the approximation $x^{(k)}$.

11.2.1 Steepest Descent

One way to guarantee a decrease of $J(x) = \|x - \bar{x}\|_A^2$ at every step of a line search method 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 (11.16) 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 $x^{(0)}$, is given by

$$t_k = \frac{\langle r^{(k)}, r^{(k)} \rangle}{\langle r^{(k)}, Ar^{(k)} \rangle}, \tag{11.18}$$

$$x^{(k+1)} = x^{(k)} + t_k r^{(k)}, \tag{11.19}$$

$$r^{(k+1)} = r^{(k)} - t_k A r^{(k)}, \tag{11.20}$$

for $k = 0, 1, \ldots$. Formula (11.20), which comes from subtracting A times (11.19) to b, is preferable to using the definition of the residual, i.e. $r^{(k+1)} = b - Ax^{(k)}$, due to round-off errors.

If A is an $n \times n$ diagonal, positive definite matrix, the steepest descent method finds the minimum in at most n steps. This is easy to visualize for $n = 2$ as the level sets of J are ellipses with their principal axes aligned with the coordinate axes. For a general, non-diagonal positive definite matrix A, convergence of the steepest descent sequence to the minimizer of J and hence to the solution of $Ax = b$ is guaranteed but it may not be reached in a finite number of steps.

11.3 The Conjugate Gradient Method

The steepest descent method uses an optimal search direction *locally* but not *globally* and as a results it converges in general very slowly to the minimizer.

A key strategy to accelerate convergence in line search methods is to widen our search space by considering the previous search directions, not just the current one. Obviously, we would like the $v^{(k)}$'s to be linear independent.

Recall that $\left(x^{(k)} - x^{(0)}\right) \in \text{span}\{v^{(0)}, v^{(1)}, \ldots, v^{(k-1)}\}$. We are going to denote

$$V_k = \text{span}\{v^{(0)}, v^{(1)}, \ldots, v^{(k-1)}\} \tag{11.21}$$

and write $x \in x^{(0)} + V_k$ to mean that $x = x^{(0)} + v$ with $v \in V_k$.

The idea is to select $v^{(0)}, v^{(1)}, \ldots, v^{(k-1)}$ such that

$$x^{(k)} = \min_{x \in x^{(0)}+V_k} \|x - \bar{x}\|_A^2. \tag{11.22}$$

If the search directions are linearly independent, as k increases our search space grows so the minimizer would be found in at most n steps, when $V_n = \mathbb{R}^n$.

Let us derive a condition for the minimizer of $J(x) = \frac{1}{2}\|x - \bar{x}\|_A^2$ in $x^{(0)} + V_k$. Suppose $x \in x^{(0)} + V_k$. Then, there are scalars $c_0, c_1, \ldots, c_{k-1}$ such that

$$x^{(k)} = x_0 + c_0 v^{(0)} + c_1 v^{(1)} + \cdots + c_{k-1} v^{(k-1)}. \tag{11.23}$$

For fixed $v^{(0)}, v^{(1)}, \ldots, v^{(k-1)}$, define the following function of $c_0, c_1, \ldots, c_{k-1}$

$$G(c_0, c_1, ..., c_{k-1}) := J\left(x_0 + c_0 v^{(0)} + c_1 v^{(1)} + \cdots + c_{k-1} v^{(k-1)}\right) \tag{11.24}$$

Because J is a quadratic function, the minimizer of G is the critical point $c_0^*, c_1^*, ..., c_{k-1}^*$

$$\frac{\partial G}{\partial c_j}(c_0^*, c_1^*, ..., c_{k-1}^*) = 0, \quad j = 0, \ldots, k-1. \tag{11.25}$$

But by the Chain Rule

$$0 = \frac{\partial G}{\partial c_j} = \nabla J \cdot v^{(j)} = -\langle r^{(k)}, v^{(j)} \rangle, \quad j = 0, 1, \ldots, k-1. \tag{11.26}$$

We have proved the following theorem.

Theorem 11.1. *The vector $x^{(k)} \in x^{(0)} + V^k$ minimizes $\|x - \bar{x}\|_A^2$ over $x^{(0)} + V_k$, for $k = 0, 1, \ldots$ if and only if*

$$\langle r^{(k)}, v^{(j)} \rangle = 0, \quad j = 0, 1, \ldots, k-1. \tag{11.27}$$

That is, the residual $r^{(k)} = b - Ax^{(k)}$ is orthogonal to all the 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 (11.16). As we have done in the Steepest Descent method, we find that 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,

$$x^{(1)} = x^{(0)} + t_0 v^{(0)}, \tag{11.28}$$

$$r^{(1)} = r^{(0)} - t_0 A v^{(0)} \tag{11.29}$$

and

$$\langle r^{(1)}, v^{(0)} \rangle = \langle r^{(0)}, v^{(0)} \rangle - t_0 \langle v^{(0)}, A v^{(0)} \rangle = 0 \tag{11.30}$$

where the last equality follows from the definition (11.16) of t_0. Now,

$$r^{(2)} = r^{(1)} - t_1 A v^{(1)} \tag{11.31}$$

and consequently

$$\langle r^{(2)}, v^{(0)} \rangle = \langle r^{(1)}, v^{(0)} \rangle - t_1 \langle v^{(0)}, A v^{(1)} \rangle = -t_1 \langle v^{(0)}, A v^{(1)} \rangle. \tag{11.32}$$

Thus if

$$\langle v^{(0)}, A v^{(1)} \rangle = 0 \tag{11.33}$$

then $\langle r^{(2)}, v^{(0)} \rangle = 0$. Moreover, $r^{(2)} = r^{(1)} - t_1 A v^{(1)}$ from which it follows that

$$\langle r^{(2)}, v^{(1)} \rangle = \langle r^{(1)}, v^{(1)} \rangle - t_1 \langle v^{(1)}, A v^{(1)} \rangle = 0, \tag{11.34}$$

where in the last equality we have used the definition of t_1, (11.16). Thus, if condition (11.33) holds we can guarantee that $\langle r^{(1)}, v^{(0)} \rangle = 0$ and $\langle r^{(2)}, v^{(j)} \rangle = 0, j = 0, 1$, i.e. we satisfy the conditions of Theorem 11.1 for $k = 1, 2$.

Definition 11.1. *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*

$$\langle x, Ay \rangle = 0. \tag{11.35}$$

We can now proceed by induction to prove the following theorem.

Theorem 11.2. *Suppose $v^{(0)}, ..., v^{(k-1)}$ are conjugate with respect to A, then for $k = 1, 2, \ldots$*

$$\langle r^{(k)}, v^{(j)} \rangle = 0, \quad j = 0, 1, \ldots, k-1.$$

Proof. Let us do induction. We know the statement is true for $k = 1$. Suppose

$$\langle r^{(k-1)}, v^{(j)} \rangle = 0, \quad j = 0, 1, ..., k-2. \tag{11.36}$$

Recall that $r^{(k)} = r^{(k-1)} - t_{k-1} A v^{(k-1)}$ and so

$$\langle r^{(k)}, v^{(k-1)} \rangle = \langle r^{(k-1)}, v^{(k-1)} \rangle - t_{k-1} \langle v^{(k-1)}, A v^{(k-1)} \rangle = 0 \tag{11.37}$$

because of the choice (11.16) of t_{k-1}. Now, for $j = 0, 1, \ldots, k-2$

$$\langle r^{(k)}, v^{(j)} \rangle = \langle r^{(k-1)}, v^{(j)} \rangle - t_{k-1} \langle v^{(j)}, A v^{(k-1)} \rangle = 0, \tag{11.38}$$

where the first term is zero because of the induction hypothesis and the second term is zero because the search directions are conjugate. □

Combining Theorems 11.1 and 11.2 we get the following important conclusion.

Theorem 11.3. *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 $\|x - \bar{x}\|_A^2$ over $x^{(0)} + V_k$.*

11.3.1 Generating the Conjugate Search Directions

The conjugate gradient method, due to Hestenes and Stiefel, is an ingenious approach to generating efficiently the set of conjugate search directions. The idea is to modify the negative gradient direction, $r^{(k)}$, by adding information about the previous search direction, $v^{(k-1)}$. Specifically, we start with

$$v^{(k)} = r^{(k)} + s_k v^{(k-1)}, \tag{11.39}$$

where the scalar s_k is chosen so that $v^{(k)}$ is conjugate to $v^{(k-1)}$ with respect to A, i.e.

$$0 = \langle v^{(k)}, A v^{(k-1)} \rangle = \langle r^{(k)}, A v^{(k-1)} \rangle + s_k \langle v^{(k-1)}, A v^{k-1)} \rangle \tag{11.40}$$

which gives

$$s_k = -\frac{\langle r^{(k)}, A v^{(k-1)} \rangle}{\langle v^{(k-1)}, A v^{(k-1)} \rangle}. \tag{11.41}$$

Magically this simple construction renders all the search directions conjugate and the residuals orthogonal!

Theorem 11.4.

$$(a) \quad \langle r^{(i)}, r^{(j)} \rangle = 0, \quad i \neq j.$$
$$(b) \quad \langle v^{(i)}, Av^{(j)} \rangle = 0, \quad i \neq j.$$

Proof. By the choice of t_k and s_k it follows that

$$\langle r^{(k+1)}, r^{(k)} \rangle = 0, \tag{11.42}$$
$$\langle v^{(k+1)}, v^{(k)} \rangle = 0, \tag{11.43}$$

for $k = 0, 1, \ldots$ Let us now proceed by induction. We know $\langle r^{(1)}, r^{(0)} \rangle = 0$ and $\langle v^{(1)}, v^{(0)} \rangle = 0$. Suppose $\langle r^{(i)}, r^{(j)} \rangle = 0$ and $\langle v^{(i)}, Av^{(j)} \rangle = 0$ holds for $0 \leq j < i \leq k$. We need to prove that this holds also for $0 \leq j < i \leq k+1$. In view of (11.42) and (11.43) we can assume $j < k$. Now,

$$\begin{aligned} \langle r^{(k+1)}, r^{(j)} \rangle &= \langle r^{(k)} - t_k Av^{(k)}, r^{(j)} \rangle \\ &= \langle r^{(k)}, r^{(j)} \rangle - t_k \langle r^{(j)}, Av^{(k)} \rangle \\ &= -t_k \langle r^{(j)}, Av^{(k)} \rangle, \end{aligned} \tag{11.44}$$

where we have used the induction hypothesis on the orthogonality of the residuals for the last equality. B $v^{(j)} = r^{(j)} + s_{j-1} v^{(j-1)}$ and so $r^{(j)} = v^{(j)} - s_{j-1} v^{(j-1)}$. Thus,

$$\begin{aligned} \langle r^{(k+1)}, r^{(j)} \rangle &= -t_k \langle v^{(j)} - s_{j-1} v^{(j-1)}, Av^{(k)} \rangle \\ &= -t_k \langle v^{(j)}, Av^{(k)} \rangle + t_k s_j \langle v^{(j-1)}, Av^{(k)} \rangle = 0. \end{aligned} \tag{11.45}$$

Also for $j < k$

$$\begin{aligned} \langle v^{(k+1)}, Av^{(i)} \rangle &= \langle r^{(k+1)} + s_k v^{(k)}, Av^{(i)} \rangle \\ &= \langle r^{(k+1)}, Av^{(i)} \rangle + s_k \langle v^{(k)}, Av^{(i)} \rangle \\ &= \langle r^{(k+1)}, \frac{1}{t_j} (r^{(j)} - r^{(j+1)}) \rangle \\ &= \frac{1}{t_j} \langle r^{(k+1)}, r^{(j)} \rangle - \frac{1}{t_j} \langle r^{(k+1)}, r^{(j+1)} \rangle = 0. \end{aligned} \tag{11.46}$$

□

The conjugate gradient method is completely specified by (11.11), (11.16), (11.39), (11.41). We are now going to do some algebra to get computationally better formulas for t_k and s_k.

Recall that

$$t_k = \frac{\langle v^{(k)}, r^{(k)} \rangle}{\langle v^{(k)}, Av^{(k)} \rangle}.$$

Now,

$$\begin{aligned} \langle v^{(k)}, r^{(k)} \rangle &= \langle r^{(k)} + s_k v^{(k-1)}, r^{(k)} \rangle \\ &= \langle r^{(k)}, r^{(k)} \rangle + s_k \langle v^{(k-1)}, r^{(k)} \rangle = \langle r^{(k)}, r^{(k)} \rangle. \end{aligned} \tag{11.47}$$

Therefore

$$t_k = \frac{\langle r^{(k)}, r^{(k)} \rangle}{\langle r^{(k)}, Ar^{(k)} \rangle}. \tag{11.48}$$

Let us now work with the numerator of s_{k+1}, the inner product $\langle r^{(k+1)}, Av^{(k)} \rangle$. First recall that $r^{(k+1)} = r^{(k)} - t_k Av^{(k)}$ and so $t_k Av^{(k)} = r^{(k)} - r^{(k+1)}$. Therefore,

$$-\langle r^{(k+1)}, Av^{(k)} \rangle = \frac{1}{t_k} \langle r^{(k+1)}, r^{(k+1)} - r^{(k)} \rangle = \frac{1}{t_k} \langle r^{(k+1)}, r^{(k+1)} \rangle. \tag{11.49}$$

And for the denominator, we have

$$\begin{aligned} \langle v^{(k)}, Av^{(k)} \rangle &= \frac{1}{t_k} \langle v^{(k)}, r^{(k)} - r^{(k+1)} \rangle \\ &= \frac{1}{t_k} \langle v^{(k)}, r^{(k)} \rangle - \frac{1}{t_k} \langle v^{(k)}, r^{(k+1)} \rangle \\ &= \frac{1}{t_k} \langle r^{(k)} + s_k v^{(k-1)}, r^{(k)} \rangle \\ &= \frac{1}{t_k} \langle r^{(k)}, r^{(k)} \rangle + \frac{s_k}{t_k} \langle v^{(k+1)}, r^{(k)} \rangle = \frac{1}{t_k} \langle r^{(k)}, r^{(k)} \rangle. \end{aligned} \tag{11.50}$$

Thus, we can write

$$s_{k+1} = \frac{\langle r^{(k+1)}, r^{(k+1)} \rangle}{\langle r^{(k)}, r^{(k)} \rangle}. \tag{11.51}$$

A pseudo-code for the conjugate gradient method is given in Algorithm 11.1. The main cost per iteration of the conjugate gradient method is the evaluation of $Av^{(k)}$. If A is sparse then this product of a and a vector can be done

cheaply by avoiding to operate with the zeros of A. For example, for matrix in the solution of Poisson's equation in 2D (10.76), the cost of computing $Av^{(k)}$ is just $O(n)$, where $n = N^2$ is the total number of unknowns.

Algorithm 11.1 The conjugate gradient Method

1: Given $x^{(0)}$ and TOL, set $r^{(0)} = b - Ax^{(0)}$, $v^{(0)} = r^{(0)}$, and $k = 0$.
2: **while** $\|r^{(k)}\|_2 > TOL$ **do**
3:
$$t_k \leftarrow \frac{\langle r^{(k)}, r^{(k)} \rangle}{\langle v^{(k)}, Av^{(k)} \rangle}$$
4:
$$x^{(k+1)} \leftarrow x^{(k)} + t_k v^{(k)}$$
5:
$$r^{(k+1)} \leftarrow r^{(k)} - t_k A v^{(k)}$$
6:
$$s_{k+1} \leftarrow \frac{\langle r^{(k+1)}, r^{(k+1)} \rangle}{\langle r^{(k)}, r^{(k)} \rangle}$$
7:
$$v^{(k+1)} \leftarrow r^{(k+1)} + s_{k+1} v^{(k)}$$
8: $k \leftarrow k + 1$
9: **end while**

Theorem 11.5. *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 11.4, 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. □

11.4 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 $x^{(1)} = x^{(0)} + t_0 r^{(0)}$

and evaluate the residual $r^{(1)}$, etc. If we use the definition of the residual we have

$$r^{(1)} = b - Ax^{(1)} = b - Ax^{(0)} + t_0 Ar^{(0)} = r^{(0)} + t_0 Ar^{(0)} \tag{11.52}$$

so that $r^{(1)}$ is a linear combination of $r^{(0)}$ and $Ar^{(0)}$. Similarly,

$$\begin{aligned} x^{(2)} &= x^{(1)} + t_1 v^{(1)} \\ &= x^{(0)} + t_0 r^{(0)} + t_1 r^{(1)} + t_1 s_0 r^{(0)} \\ &= x^{(0)} + (t_0 + t_1 s_0) r^{(0)} + t_1 r^{(1)} \end{aligned} \tag{11.53}$$

so that $r^{(2)} = b - Ax^{(2)}$ is a linear combination of $r^{(0)}$, $Ar^{(0)}$, and $A^2 r^{(0)}$ and so on.

Definition 11.2. *The set* $\mathcal{K}_k(r^{(0)}, A) = span\{r^{(0)}, Ar^{(0)}, ..., A^{k-1}r^{(0)}\}$ *is called the Krylov subspace of degree* k *for* $r^{(0)}$.

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 $\|x - \bar{x}\|_A^2$ over $\mathcal{K}_k(r^{(0)}, A)$.

Theorem 11.6. $\mathcal{K}_k(r^{(0)}, A) = span\{r^{(0)}, ..., r^{(k-1)}\} = span\{v^{(0)}, ..., v^{(k-1)}\}$.

Proof. We will proof it by induction. The case $k = 1$ 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 $r^{(k)}, v^{(k-1)} \in \mathcal{K}_k(r^{(0)}; A)$ 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

$$r^{(k)} \in \mathcal{K}_{k+1}(r^{(0)}, A).$$

Consequently,

$$\text{span}\{r^{(0)}, ..., r^{(k)}\} \subseteq \mathcal{K}_{k+1}(r^{(0)}, A).$$

We now prove the reverse inclusion,

$$\text{span}\{r^{(0)}, ..., r^{(k)}\} \supseteq \mathcal{K}_{k+1}(r^{(0)}, A).$$

Note that $A^k r^{(0)} = A(A^{k-1} r^{(0)})$. But by the induction hypothesis

$$\text{span}\{r^{(0)}, Ar^{(0)}, ..., A^{k-1} r^{(0)}\} = \text{span}\{v^{(0)}, ..., v^{(k-1)}\}.$$

Given that

$$A^k r^{(0)} = A(A^{k-1} r^{(0)}) \in \text{span}\{Av^{(0)}, ..., Av^{(k-1)}\}$$

and since

$$Av^{(j)} = \frac{1}{t_j}(r^{(j)} - r^{(j+1)})$$

it follows that

$$A^k r^{(0)} \in \text{span}\{r^{(0)}, r^{(1)}, ..., r^{(k)}\}.$$

Thus,

$$\text{span}\{r^{(0)}, ..., r^{(k)}\} = \mathcal{K}_{k+1}(r^{(0)}, A).$$

For the last equality we observe that $\text{span}\{v^{(0)}, ..., v^{(k)}\} = \text{span}(\{v^{(0)}, ..., v^{(k)}, r^{(k)}\}$ because $v^{(k)} = r^{(k)} + s_k v^{(k-1)}$ and by the induction hypothesis

$$\begin{aligned} \text{span}\{v^{(0)}, ..., v^{(k)}, r^{(k)}\} &= \text{span}\{r^{(0)}, Ar^{(0)}, ..., A^k r^{(0)}, r^{(k)}\} \\ &= \text{span}\{r^{(0)}, r^{(1)}, ..., r^{(k)}, r^{(k)}\} \\ &= \mathcal{K}_{k+1}(r^{(0)}, A). \end{aligned} \tag{11.54}$$

□

11.5 Convergence of the Conjugate Gradient Method

Let us define the initial error as $e^{(0)} = x^{(0)} - \bar{x}$. Then $Ae^{(0)} = Ax^{(0)} - A\bar{x}$ implies that

$$r^{(0)} = -Ae^{(0)}. \tag{11.55}$$

For the conjugate gradient method $x^{(k)} \in x^{(0)} + \mathcal{K}_k(r^{(0)}, A)$ and in view of (11.55) we have that

$$x^{(k)} - \bar{x} = e^{(0)} + c_1 Ae^{(0)} + c_2 A^2 e^{(0)} + \cdots + c_k A^k e^{(0)}, \tag{11.56}$$

for some real constants $c_1, \ldots, c_k$. In fact,

$$\|x^{(k)} - \bar{x}\|_A = \min_{p \in \tilde{\mathbb{P}}_k} \|p(A)e^{(0)}\|_A, \tag{11.57}$$

where $\tilde{\mathbb{P}}_k$ is the set of all polynomials of degree $\leq k$ and that are equal to one at 0. Since A is symmetric positive definite all its eigenvalues are real and positive. Let's 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

$$p(A)e^{(0)} = \sum_{j=1}^{n} p(\lambda_j)\alpha_j v_j. \tag{11.58}$$

Therefore,

$$\begin{aligned} \|p(A)e^{(0)}\|_A^2 = \langle p(A)e^{(0)}, Ap(A)e^{(0)} \rangle &= \sum_{j=1}^{n} p^2(\lambda_j)\lambda_j \alpha_j^2 \\ &\leq \left(\max_j p^2(\lambda_j) \right) \sum_{j=1}^{n} \lambda_j \alpha_j^2 \end{aligned} \tag{11.59}$$

and since

$$\|e^{(0)}\|_A^2 = \sum_{j=1}^{n} \lambda_j \alpha_j^2 \tag{11.60}$$

we get

$$\|e^{(k)}\|_A \leq \min_{p \in \tilde{\mathbb{P}}_k} \max_j |p(\lambda_j)| \, \|e^{(0)}\|_A. \tag{11.61}$$

The min max term can be estimated using the Chebyshev polynomial T_k with the change of variables

$$f(\lambda) = \frac{2\lambda - \lambda_1 - \lambda_n}{\lambda_n - \lambda_1} \tag{11.62}$$

to map $[\lambda_1, \lambda_n]$ to $[-1, 1]$. The polynomial

$$p(\lambda) = \frac{1}{T_k(f(0))} T_k(f(\lambda)) \tag{11.63}$$

is in $\tilde{\mathbb{P}}_k$ and since $|Tk(f(\lambda))| \leq 1$

$$\max_j |p(\lambda_j)| = \frac{1}{|T_k(f(0))|}. \tag{11.64}$$

Now

$$|T_k(f(0))| = \left|T_k\left(\frac{\lambda_1+\lambda_n}{\lambda_n-\lambda_1}\right)\right| = \left|T_k\left(\frac{\lambda_n/\lambda_1+1}{\lambda_n/\lambda_1-1}\right)\right| = \left|T_k\left(\frac{\kappa_2(A)+1}{\kappa_2(A)-1}\right)\right| \tag{11.65}$$

because $\kappa_2(A) = \lambda_n/\lambda_1$ is the condition number of A in the 2-norm. Now we use an identity of Chebyshev polynomials, namely if $x = (z+1/z)/2$ then $T_k(x) = (z^k + 1/z^k)/2$. Noting that

$$\frac{\kappa_2(A)+1}{\kappa_2(A)-1} = \frac{1}{2}(z+1/z) \tag{11.66}$$

for

$$z = (\sqrt{\kappa_2(A)}+1)/(\sqrt{\kappa_2(A)}-1) \tag{11.67}$$

we obtain

$$\left|T_k\left(\frac{\kappa_2(A)+1}{\kappa_2(A)-1}\right)\right|^{-1} \leq 2\left(\frac{\sqrt{\kappa_2(A)}-1}{\sqrt{\kappa_2(A)}+1}\right)^k. \tag{11.68}$$

Thus we get the upper bound error estimate for the error in the conjugate gradient method:

$$\|e^{(k)}\|_A \leq 2\left(\frac{\sqrt{\kappa_2(A)}-1}{\sqrt{\kappa_2(A)}+1}\right)^k \|e^{(0)}\|_A. \tag{11.69}$$

Chapter 12

Eigenvalue Problems

In this chapter we take a brief look at some numerical methods for the standard eigenvalue problem, i.e. of finding eigenvalues λ and eigenvectors v of an $n \times n$ matrix A.

12.1 The Power Method

Suppose that A has a dominant eigenvalue:

$$|\lambda_1| > |\lambda_2| \geq\geq \cdots \geq |\lambda_n| \tag{12.1}$$

and a complete set of eigenvector $v_1, \ldots, v_n$ associated to $\lambda_1, \ldots, \lambda_n$, respectively. Then, each vector $v \in \mathbb{R}^n$ can be written in terms of the eigenvectors as

$$v = c_1 v_1 + \cdots + c_n v_n \tag{12.2}$$

and

$$A^k v = c_1 \lambda_1^k v_1 + \cdots + c_n \lambda_1^k v_n = c_1 \lambda_1^k \left[v_1 + \sum_{j=2}^{n} \frac{c_j}{c_1} \left(\frac{\lambda_j}{\lambda_1} \right)^k v_j \right]. \tag{12.3}$$

Therefore $A^k v / c_1 \lambda_1^k \to v_1$ and we get a method to determine v_1 and λ_1. To avoid overflow we normalize the approximating vector at each iteration as Algorithm 12.1 shows. The rate of convergence of the power method is determined by the ratio λ_2/λ_1. From (12.3), $|\lambda^{(k)} - \lambda| = O(|\lambda_2/\lambda_1|^k)$, where $\lambda^{(k)}$ is the approximation to λ_1 at the k-th iteration.

Algorithm 12.1 The Power Method

1: Set $k = 0$, $\|r\|_2 >> 1$.
2: **while** $\|r\|_2 > TOL$ **do**
3: $v \leftarrow Av$
4: $v \leftarrow v/\|v\|_2$
5: $\lambda \leftarrow v^T Av$
6: $r = Av - \lambda v$
7: $k \leftarrow k + 1$
8: **end while**

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, and when $|\lambda_2/\lambda_1| << 1$.

One 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 accelerated convergence. For example, suppose A is symmetric and has eigenvalues $100, 90, 50, 40, 30, 30$ the matrix $A - 60I$ has eigenvalues $40, 30, -10, -20, -30, -30$ and the power method would converge at a rate of $30/40 = 0.75$ instead of a rate of $90/100 = 0.9$.

A variant of the shift power method is the inverse power method, which applies the iteration to the matrix $(A - sI)^{-1}$. The inverse is not actually computed; instead the linear system $(A - sI)v^{(k)} = v^{(k-1)}$ is solved at every iteration. The method will converge to the eigenvalue λ_j for which $|\lambda_j - s$ is the smallest and so with an appropriate choice for s it is possible to converge to each of the eigenpairs of A.

12.2 Methods Based on Similarity Transformations

For a general $n \times n$ matrix, numerical methods for eigenvalues are typically based on a sequence of similarity transformations

$$A_k = P_k^{-1} A P_k, \quad k = 1, 2, \ldots \tag{12.4}$$

to attempt to converge to either a diagonal matrix or to an upper triangular matrix.

12.2.1 The QR method

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 of a matrix. Here Q is a unitary (orthogonal in the real case) matrix and R is upper triangular.

Given an $n \times n$ matrix A, we set $A_1 = A$, obtain its QR factorization

$$A_1 = Q_1 R_1 \tag{12.5}$$

and define $A_2 = R_1 Q_1$ so that

$$A_2 = R_1 Q_1 = Q_1^* A Q_1, \tag{12.6}$$

etc. The $k+1$-st similar matrix is generated by

$$A_{k+1} = R_k Q_k = Q_k^* A_k Q_k = (Q_1 \cdots Q_k)^* A (Q_1 \cdots Q_k). \tag{12.7}$$

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.

The QR factorization is expensive, $O(n^3)$ operations, so the QR method is usually applied only after the original matrix A has been reduced to a tridiagonal matrix if A is symmetric or to an upper Hessenberg form if A is not symmetric. This reduction is done with a sequence of orthogonal transformations known as Householder reflections.

Example 12.1. *Consider the matrix the* 5×5 *matrix*

$$A = \begin{bmatrix} 12 & 13 & 10 & 7 & 7 \\ 13 & 18 & 9 & 8 & 15 \\ 10 & 9 & 10 & 4 & 12 \\ 7 & 8 & 4 & 4 & 6 \\ 7 & 15 & 12 & 6 & 18 \end{bmatrix} \tag{12.8}$$

the $A_{20} = R_{19} Q_{19}$ *produced by the* QR *method gives the eigenvalues of* A

within 4 digits of accuracy

$$A_{20} = \begin{bmatrix} 51.7281 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 8.2771 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 4.6405 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & -2.8486 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.2028 \end{bmatrix}. \tag{12.9}$$

Chapter 13

Non-Linear Equations

13.1 Introduction

In this chapter we consider the problem of finding zeros of a continuous function f, i.e. solving $f(x) = 0$ for example $e^x - x = 0$ or a system of nonlinear equations:

$$\begin{aligned} f_1(x_1, x_2, \cdots, x_n) &= 0, \\ f_2(x_1, x_2, \cdots, x_n) &= 0, \\ &\vdots \\ f_n(x_1, x_2, \cdots, x_n) &= 0. \end{aligned} \tag{13.1}$$

We are going to write this generic system in vector form as

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \tag{13.2}$$

where $\mathbf{f} : U \subseteq \mathbb{R}^n \to \mathbb{R}^n$. Unless otherwise noted the function $\mathbf{f}$ is assumed to be smooth in its domain U.

We are going to start with the scalar case, $n = 1$ and look a very simple but robust method that relies only on the continuity of the function and the existence of a zero.

13.2 Bisection

Suppose we are interested in solving a nonlinear equation in one unknown

$$f(x) = 0, \tag{13.3}$$

where f is a continuous function on an interval $[a, b]$ and has at least one zero there.

Suppose that f has values of different sign at the end points of the interval, i.e.

$$f(a)f(b) < 0. \tag{13.4}$$

By the Intermediate Value Theorem, f has at least one zero in (a, b). To locate a zero we bisect the interval and check on which subinterval f changes sign. We repeat the process until we bracket a zero within a desired accuracy. The Bisection algorithm to find a zero x^* is shown below.

Algorithm 13.1 The Bisection Method

Given f, a and b $(a < b)$, TOL, and N_{max}, set $k = 1$ and do:
while $(b - a) > TOL$ and $k \leq N_{max}$ **do**
    $c = (a + b)/2$
    if $f(c) == 0$ **then**
        $x^* = c$ ▷ This is the solution
        stop
    end if
    if sign$(f(c))$ == sign$(f(a))$ **then**
        $a \leftarrow c$
    else
        $b \leftarrow c$
    end if
    $k \leftarrow k + 1$
end while
$x^* \leftarrow (a + b)/2$

13.2.1 Convergence of the Bisection Method

With the bisection method we generate a sequence

$$c_k = \frac{a_k + b_k}{2}, \quad k = 1, 2, \ldots \tag{13.5}$$

where each a_k and b_k are the endpoints of the subinterval we select at each bisection step (because f changes sign there). Since

$$b_k - a_k = \frac{b - a}{2^{k-1}}, \quad k = 1, 2, \ldots \tag{13.6}$$

and $c_k = \frac{a_k+b_k}{2}$ is the midpoint of the interval then

$$|c_k - x^*| \leq \frac{1}{2}(b_k - a_k) = \frac{b-a}{2^k} \tag{13.7}$$

and consequently $c_k \to x^*$, a zero of f in $[a, b]$.

13.3 Rate of Convergence

We now define in precise terms the rate of convergence of a sequence of approximations to a value x^*.

Definition 13.1. *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*

$$|x_{n+1} - x^*| \leq C\,|x_n - x^*|^p\,, \quad \textit{for all } n \geq N. \tag{13.8}$$

or equivalently

$$\lim_{n\to\infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|^p} = C. \tag{13.9}$$

$C < 1$ for $p = 1$.

Example 13.1. *The sequence generated by the bisection method converges linearly to x^* because*

$$\frac{|c_{n+1} - x^*|}{|c_n - x^*|} \leq \frac{\frac{b-a}{2^{n+1}}}{\frac{b-a}{2^n}} = \frac{1}{2}.$$

Let's examine the significance of the rate of convergence. Consider first, $p = 1$, linear convergence. Suppose

$$|x_{n+1} - x^*| \approx C|x_n - x^*|, \quad n \geq N. \tag{13.10}$$

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

$$|x_{N+k} - x^*| \approx C^k |x_N - x^*|, \quad k = 0, 1, \ldots \tag{13.11}$$

and this is the reason of the requirement $C < 1$ for $p = 1$. If the error at the N step, $|x_N - x^*|$, is small enough it will be reduced by a factor of C^k after k more steps. Setting $C^k = 10^{-d_k}$, then the error $|x_N - x^*|$ will be reduced approximately

$$d_k = \left(\log_{10} \frac{1}{C} \right) k \tag{13.12}$$

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

$$|x_{N+k} - x^*| \approx C^{2^k - 1} |x_N - x^*|^{2^k}, \quad k = 0, 1, \ldots \tag{13.13}$$

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

$$d_k = \left(\log_{10} \frac{1}{C} + \log_{10} \frac{1}{|x_N - x^*|} \right) (2^k - 1). \tag{13.14}$$

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 $\alpha_p = \frac{1}{p-1} \log_{10} \frac{1}{C} + \log_{10} \frac{1}{|x_N - x^*|}$.

13.4 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 $c = \frac{a+b}{2}$ to subdivide the interval in question we could use the root of linear polynomial interpolating $(a, f(a))$ 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 approach to use interpolation to obtain numerical methods for $f(x) = 0$ is to proceed as follows: Given $m+1$ approximations to the zero of f, $x_0, \ldots, x_m$, 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 to the zero of f. 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**.

13.5 Newton's Method

If the function f is smooth, say at least $C^2[a, b]$, and we have already a good approximation x_0 to 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.

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0). \tag{13.15}$$

Then we can define the next approximation as the zero of that tangent line, i.e.

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}, \tag{13.16}$$

etc. At the k step or iteration we get the new approximation x_{k+1} according to:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, \ldots \tag{13.17}$$

This iteration is called Newton's method or Newton-Raphson's method. There are some conditions for this method to work and converge. But when it does converge it does it at least quadratically. Indeed, a Taylor expansion of f around x_k gives

$$f(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(\xi_k)(x - x_k)^2, \tag{13.18}$$

where ξ_k is a point between x and x_k. Evaluating at $x = x*$ and using that $f(x^*) = 0$ we get

$$0 = f(x_k) + f'(x_k)(x^* - x_k) + \frac{1}{2}f''(\xi_k)(x^* - x_k)^2, \tag{13.19}$$

which we can recast as

$$x^* = x_k - \frac{f(x_k)}{f'(x_k)} - \frac{\frac{1}{2}f''(\xi_k)}{f'(x_k)}(x^* - x_k)^2 = x_{k+1} - \frac{\frac{1}{2}f''(\xi_k)}{f'(x_k)}(x^* - x_k)^2. \tag{13.20}$$

Thus,

$$|x_{k+1} - x^*| = \frac{\frac{1}{2}|f''(\xi_k)|}{|f'(x_k)|}|x_k - x^*|^2. \tag{13.21}$$

So if the sequence $\{x_k\}_{k=0}^{\infty}$ generated by Newton's method converges then it does so at least quadratically.

Theorem 13.1. *Let x^* be a simple zero of f (i.e. $f(x^*) = 0$ and $f'(x^*) \neq 0$) and suppose $f \in c^2$. Then there's a neighborthood I_ϵ of x^* such that Newton's method converges to x^* for any initial guess in I_ϵ.*

Proof. Since f' is continuous and $f'(x^*) \neq 0$ we can choose $\epsilon > 0$, sufficiently small so that $f'(x) \neq 0$ for all x such that $|x - x^*| \leq \epsilon$ (this is I_ϵ) and that $\epsilon M(\epsilon) < 1$ where

$$M(\epsilon) = \frac{\frac{1}{2}\max_{x\in I_\epsilon}|f''(x)|}{\min_{x\in I_\epsilon}|f'(x)|}.$$

This is possible because $\lim_{\epsilon\to 0} M(\epsilon) = \frac{\frac{1}{2}|f''(x^*)|}{|f'(x^*)|} < +\infty$.

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

$$\begin{aligned} f(x) &= f(x^*) + f'(x^*)(x - x^*) + \frac{1}{2}f''(\xi)(x - x^*)^3 \\ &= f'(x^*)(x - x^*)\left(1 + (x - x^*)\frac{\frac{1}{2}f''(\xi)}{f'(x^*)}\right), \end{aligned} \tag{13.22}$$

and since

$$\left|(x - x^*)\frac{\frac{1}{2}f''(\xi)}{f'(x^*)}\right| = |x - x^*|\frac{\frac{1}{2}|f''(\xi)|}{|f'(x^*)|} \leq \epsilon M(\epsilon) < 1 \tag{13.23}$$

$f(x) \neq 0$ 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 (13.21) with $k = 0$ it follow that $x_1 \in I_\epsilon$ as

$$|x_1 - x^*| = |x_0 - x|^2 \left| \frac{\frac{1}{2} f''(\xi_0)}{f'(x_0)} \right| \leq \epsilon^2 M(\epsilon) \leq \epsilon. \tag{13.24}$$

Now assume that $x_k \in I_\epsilon$ then again from (13.21)

$$|x_{k+1} - x^*| = |x_k - x|^2 \left| \frac{\frac{1}{2} f''(\xi_k)}{f'(x_k)} \right| \leq \epsilon^2 M(\epsilon) < \epsilon \tag{13.25}$$

so $x_{k+1} \in I_\epsilon$. Now,

$$\begin{aligned} |x_{k+1} - x^*| &\leq |x_k - x^*|^2 M(\epsilon) = |x_k - x^*| \epsilon M(\epsilon) \\ &\leq |x_{k-1} - x^*| (\epsilon M(\epsilon))^2 \\ &\vdots \\ &\leq |x_0 - x^*| (\epsilon M(\epsilon))^{k+1} \end{aligned}$$

and since $\epsilon M(\epsilon) < 1$ it follows that $x_k \to x^*$ as $k \to \infty$. □

The need for a good initial guess x_0 for Newton's method should be emphasized. In practice, this is obtained with another method, like bisection.

13.6 The Secant Method

Sometimes it could be computationally expensive or not possible to evaluate the derivative of f. The following method, known as the secant method, replaces the derivative by the secant:

$$x_{k+1} = x_k - \frac{f(x_k)}{\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}}, \quad k = 1, 2, \ldots \tag{13.26}$$

Note that since $f(x^*) = 0$

$$\begin{aligned}
x_{k+1} - x^* &= x_k - x^* - \frac{f(x_k) - f(x^*)}{\frac{f(x_k)-f(x_{k-1})}{x_k - x_{k-1}}}, \\
&= x_k - x^* - \frac{f(x_k) - f(x^*)}{f[x_k, x_{k-1}]} \\
&= (x_k - x^*)\left(1 - \frac{\frac{f(x_k)-f(x^*)}{x_k - x^*}}{f[x_k, x_{k-1}]}\right) \\
&= (x_k - x^*)\left(1 - \frac{f[x_k, x^*]}{f[x_k, x_{k-1}]}\right) \\
&= (x_k - x^*)\left(\frac{f[x_k, x_{k-1}] - f[x_k, x^*]}{f[x_k, x_{k-1}]}\right) \\
&= (x_k - x^*)(x_{k-1} - x^*)\left(\frac{\frac{f[x_k, x_{k-1}]-f[x_k, x^*]}{x_{k-1} - x^*}}{f[x_k, x_{k-1}]}\right) \\
&= (x_k - x^*)(x_{k-1} - x^*)\frac{f[x_{k-1}, x_k, x^*]}{f[x_k, x_{k-1}]}
\end{aligned}$$

If $x_k \to x^*$, then $\frac{f[x_{k-1},x_k,x^*]}{f[x_k,x_{k-1}]} \to \frac{\frac{1}{2}f''(x^*)}{f'(x^*)}$ and $\lim_{k\to\infty} \frac{x_{k+1}-x^*}{x_k - x^*} = 0$, i.e. the sequence generated by the secant method would converge faster than linear.

Defining $e_k = |x_k - x^*|$, the calculation above suggests

$$e_{k+1} \approx c e_k e_{k-1}. \tag{13.27}$$

Let's 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

$$e_{k+1} \approx c e_k e_{k-1} \approx c e_k \left(\frac{1}{A} e_k\right)^{\frac{1}{p}},$$

which implies

$$\frac{A^{1+\frac{1}{p}}}{c} \approx e_k^{1-p+\frac{1}{p}}. \tag{13.28}$$

Since the left hand side is a constant we must have $1 - p + \frac{1}{p} = 0$ which gives $p = \frac{1 \pm \sqrt{5}}{2}$, thus

$$p = \frac{1 + \sqrt{5}}{2} = \approx 1.61803 \tag{13.29}$$

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 in Newton's method.

13.7 Fixed Point Iteration

Newton's method is a particular example of a functional iteration of the form

$$x_{k+1} = g(x_k), \quad k = 0, 1, \ldots$$

with the particular choice of $g(x) = x - \frac{f(x)}{f'(x)}$. Clearly, if x^* is a zero of f then x^* is a fixed point of g, i.e. $g(x^*) = x^*$. We will look at fixed point iterations as a tool for solving $f(x) = 0$.

Example 13.2. *Suppose we want to solve $x - e^{-x} = 0$ in $[0, 1]$. Then if we take $g(x) = e^{-x}$, a fixed point of g corresponds to a zero of f.*

Definition 13.2. *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*

$$|g(x) - g(y)| \leq L|x - y|, \quad \textit{for all } x, y \in [a, b]. \tag{13.30}$$

If x^* is a fixed point of g in $[a, b]$ then

$$\begin{aligned} & |x_k - x^*| \\ = \; & |g(x_{k-1}) - g(x^*)| \\ \leq \; & L|x_{k-1} - x^*| \\ \leq \; & L^2|x_{k-2} - x^*| \\ \leq \; & \cdots \\ \leq \; & L^k|x_0 - x^*| \to 0, \text{ as } k \to \infty. \end{aligned}$$

Theorem 13.2. *If g is 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 converges to it for any $[a,b]$. Moreover*
(a)

$$|x_k - x^*| \leq \frac{L^*}{1-L}|x_1 - x_0|$$

(b)

$$|x_k - x^*| \leq L^k |x_0 - x^*|$$

Proof. With proved (b) already. Since $g : [a,b] \to [a,b]$, the fixed point iteration $x_{k+1} = g(x_k), k = 0, 1, ...$ is well-defined and

$$\begin{aligned} |x_{k+1} - x_k| &= |g(x_k) - g(x_{k-1})| \\ &\leq L|x_k - x_{k-1}| \\ &\leq \cdots \\ &\leq L^k |x_1 - x_0|. \end{aligned}$$

Now, for $n \geq m$

$$x_n - x_m = x_n - x_{n-1} + x_{n-1} - x_{n-2} + \ldots + x_{m+1} - x_m \tag{13.31}$$

and so

$$\begin{aligned} |x_n - x_m| &\leq |x_n - x_{n-1}| + |x_{n-1} - x_{n-2}| + \ldots + |x_{m+1} - x_m| \\ &\leq L^{n-1}|x_1 - x_0| + L^{n-2}|x_1 - x_0| + \ldots + L^m |x_1 - x_0| \\ &\leq L^m |x_1 - x_0| \left(1 + L + L^2 + \ldots L^{n-1-m}\right) \\ &\leq L^m |x_1 - x_0| \sum_{j=0}^{\infty} L^j = \frac{L^m}{1-L}|x_1 - x_0|. \end{aligned}$$

Thus given $\epsilon > 0$ there is N such

$$\frac{L^N}{1-L}|x_1 - x_0| \leq \epsilon \tag{13.32}$$

and thus for $n \geq m \geq N$, $|x_n - x_m| \leq \epsilon$, i.e. $\{x_n\}_{n=0}^{\infty}$ is a Cauchy sequence in $[a,b]$ so it converges to a point $x^* \in [a.b]$. But

$$|x_k - g(x^*)| = |g(x_{k-1}) - g(x^*)| \leq L|x_{k-1} - x^*|, \tag{13.33}$$

and so $x_k \to g(x^*)$ as $k \to \infty$ i.e. x^* is a fixed point of g.

Suppose that there are two fixed points, $x_1, x_2 \in [a,b]$. $|x_1 - x_2| = |g(x_1) - g(x_2)| \leq L|x_1 - x_2| \Rightarrow (1-L)|x_1 - x_2) \leq 0$ but $0 \leq L < 1 \Rightarrow |x_1 - x_2| = 0 \Rightarrow x_1 = x_2$ i.e the fixed point is unique. □

If g is differentiable in (a,b), then by the mean value theorem

$$g(x) - g(y) = g'(\xi)(x-y), \quad \text{for some } \xi \in [a,b]$$

and if the derivative is bounded by a constant L less than 1, i.e. $|g'(x)| \leq L$ for all $x \in (a,b)$, then $|g(x) - g(y)| \leq L|x-y|$ with $0 \leq L < 1$, i.e. g is contractive in $[a,b]$.

Example 13.3. *Let $g(x) = \frac{1}{4}(x^2+3)$ for $x \in [0,1]$. Then $0 \leq g(x) \leq 1$ and $|g'(x)| \leq \frac{1}{2}$ for all $x \in [0,1]$. So 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

$$\begin{aligned} x_{k+1} - x^* &= g(x_k) - g(x^*) \\ &= g'(\xi_k)(x_k - x^*), \quad \text{for some } \xi_k \in [x_k, x^*]. \end{aligned}$$

Thus,

$$\frac{x_{k+1} - x^*}{x_k - x^*} = g'(\xi_k) \tag{13.34}$$

and unless $g'(x^*) = 0$, the fixed point iteration converges linearly, when it does converge.

13.8 Systems of Nonlinear Equations

We now look at the problem of finding numerical approximation to the solution(s) of a nonlinear system of equations $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, where $\mathbf{f} : U \subseteq \mathbb{R}^n \to \mathbb{R}^n$.

The main approach to solve a nonlinear system is fixed point iteration

$$\mathbf{x}_{k+1} = \mathbf{G}(\mathbf{x}_k), \quad k = 0, 1, \ldots \tag{13.35}$$

where we assume that $\mathbf{G}$ is defined on a closed set $B \subseteq \mathbb{R}^n$ and $\mathbf{G} : B \to B$.

The map $\mathbf{G}$ is a contraction (with respect to some norm,$\|\cdot\|$) if there is a constant L with $0 \leq L < 1$ and

$$\|\mathbf{G}(\mathbf{x}) - \mathbf{G}(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \quad \text{for all } \mathbf{x}, \mathbf{y} \in B. \tag{13.36}$$

Then, as we know, by the contraction map principle, $\mathbf{G}$ has a unique fixed point and the sequence generated by the fixed point iteration (13.35) converges to it.

Suppose that $\mathbf{G}$ is C^1 on some convex set $B \subseteq \mathbb{R}^n$, for example a ball. Consider the linear segment $\mathbf{x} + t(\mathbf{y} - \mathbf{x})$ for $t \in [0, 1]$ with $\mathbf{x}, \mathbf{y}$ fixed in B. Define the one-variable function

$$\mathbf{h}(t) = \mathbf{G}(\mathbf{x} + t(\mathbf{y} - \mathbf{x})). \tag{13.37}$$

Then, by the Chain Rule, $\mathbf{h}'(t) = \mathbf{DG}(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))(\mathbf{y} - \mathbf{x})$, where $\mathbf{DG}$ stands for the derivative matrix of $\mathbf{G}$. Then, using the definition of $\mathbf{h}$ and the Fundamental Theorem of Calculus we have

$$\begin{aligned} \mathbf{G}(\mathbf{y}) - \mathbf{G}(\mathbf{x}) = \mathbf{h}(1) - \mathbf{h}(0) &= \int_0^1 \mathbf{h}'(t)dt \\ &= \left[\int_0^1 \mathbf{DG}(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))dt\right](\mathbf{y} - \mathbf{x}). \end{aligned} \tag{13.38}$$

Thus if there is $0 \leq L < 1$ such that

$$\|\mathbf{DG}(\mathbf{x})\| \leq L, \quad \text{for all } \mathbf{x} \in B, \tag{13.39}$$

for some subordinate norm $\|\cdot\|$. Then

$$\|\mathbf{G}(\mathbf{y}) - \mathbf{G}(\mathbf{x})\| \leq L\|\mathbf{y} - \mathbf{x}\| \tag{13.40}$$

and $\mathbf{G}$ is a contraction (in that norm). The spectral radius of $\mathbf{DG}$, $\rho(\mathbf{DG})$ willdetermine the rate of convergence of the corresponding fixed point iteration.

13.8.1 Newton's Method

By Taylor theorem

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\mathbf{x}_0) + \mathbf{Df}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) \tag{13.41}$$

so if we take $\mathbf{x}_1$ as the zero of the right ahnd side of (13.41) we get

$$\mathbf{x}_1 = \mathbf{x}_0 - [\mathbf{Df}(x_0)]^{-1}\mathbf{f}(\mathbf{x}_0). \tag{13.42}$$

Continuing this way, Newton's method for the system of equations $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ can be written as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{Df}(x_k)]^{-1}\mathbf{f}(\mathbf{x}_k). \tag{13.43}$$

In the implementation of Newton's method for a system of equations we solve the linear system $\mathbf{Df}(\mathbf{x}_k)\mathbf{w} = -\mathbf{f}(\mathbf{x}_k)$ at each iteration and update $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{w}$.

Chapter 14

Numerical Methods for ODEs

14.1 Introduction

In this chapter we will be concerned with numerical methods for the initial value problem:

$$\frac{dy(t)}{dt} = f(t, y(t)), \quad t_0 < t \leq T, \tag{14.1}$$

$$y(t_0) = \alpha. \tag{14.2}$$

The independent variable t often represents time but does not have to. Without loss of generality we will take $t_0 = 0$. The time derivative is also frequently denoted with a dot (especially in physics) or an apostrophe

$$\frac{dy}{dt} = \dot{y} = y'. \tag{14.3}$$

In (14.1)-(14.2), y and f may be vector-valued in which case we have an initial value for a system of ordinary differential equations (ODEs). The constant α in (14.2) is the given initial condition, which is a constant vector in the case of a system.

Example 14.1.

$$y'(t) = t \sin y(t), \quad 0 < t < 2\pi \tag{14.4}$$

$$y(0) = \alpha \tag{14.5}$$

Example 14.2.

$$\begin{aligned} y_1'(t) &= y_1(t)y_2(t) - y_1^2, \qquad 0 < t \le T \\ y_2'(t) &= -y_2(t) + t^2 \cos y_1(t), \quad 0 < t \le T \end{aligned} \tag{14.6}$$

$$\begin{aligned} y_1(0) &= \alpha_1, \\ y_2(0) &= \alpha_2. \end{aligned} \tag{14.7}$$

These two are examples of first order ODEs, which is the type of initial value problems we will focus on. Higher order ODEs can be written as first order systems by introducing new variables for the derivatives from the first up to one order less.

Example 14.3. *The Harmonic Oscillator.*

$$y''(t) + k^2 y(t) = 0. \tag{14.8}$$

If we define $y_1 = y$ and $y_2 = y'$ we get

$$\begin{aligned} y_1'(t) &= y_2(t), \\ y_2'(t) &= -k^2 y_1(t). \end{aligned} \tag{14.9}$$

Example 14.4.

$$y'''(t) + 2y(t)y''(t) + \cos y'(t) + e^t = 0. \tag{14.10}$$

Introducing the variables $y_1 = y$, $y_2 = y'$, and $y_3 = y''$ we obtain the first order system:

$$\begin{aligned} y_1'(t) &= y_2(t), \\ y_2'(t) &= y_3(t), \\ y_3'(t) &= -2y_1(t)y_3(t) - \cos y_2(t) - e^t. \end{aligned} \tag{14.11}$$

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 14.5. *Consider the ODE*

$$y'(t) = \sin t - y^2(t) \tag{14.12}$$

If we define $y_1 = y$ and $y_2 = t$ we can write this ODE as the autonomous system

$$\begin{aligned} y_1'(t) &= \sin y_2(t) - y_1^2(t), \\ y_2'(t) &= 1. \end{aligned} \tag{14.13}$$

We now state a fundamental theorem of existence and uniqueness of solutions to the initial value problem (14.1)-(14.2).

Theorem 14.1. *Existence and Uniqueness.*
Let

$$D = \{(t, y) : 0 \le t \le T,\, y \in \mathbb{R}^n\}\,. \tag{14.14}$$

If f is continuous in D and uniformly Lipschitz in y, i.e. there is a constant $L \ge 0$ such that

$$\|f(t, y_2) - f(t, y_1)\| \le L\|y_2 - y_1\| \tag{14.15}$$

for all $t \in [0, T]$ and all $y_1, y_2 \in \mathbb{R}^n$, then the initial value problem (14.1)-(14.2) has a unique solution for each $\alpha \in \mathbb{R}^n$.

Note that if there is $L' \ge 0$ such that

$$\left|\frac{\partial f_i}{\partial y_j}(t, y)\right| \le L' \tag{14.16}$$

for all y, $t \in [0, T]$, and $i, j = 1, \ldots, n$ then f is uniformly Lipschitz in y (equivalently, if the given norm of the derivative matrix of f is bounded by L, see Section 13.8).

Example 14.6.

$$\begin{aligned} y' &= y^{2/3}, \quad 0 < t \\ y(0) &= 0. \end{aligned} \tag{14.17}$$

The partial derivative

$$\frac{\partial f}{\partial y} = \frac{2}{3}\, y^{-1/3} \tag{14.18}$$

is not continuous around 0. *Clearly, $y \equiv 0$ is a solution of this initial value problem but so is $y(t) = \frac{1}{27}t^3$. There is no uniqueness of solution for this initial value problem.*

Example 14.7.

$$\begin{aligned} y' &= y^2, \quad 1 < t \le 3 \\ y(1) &= 3. \end{aligned} \tag{14.19}$$

We can integrate to obtain

$$y(t) = \frac{1}{2-t}, \tag{14.20}$$

which becomes unbounded as $\to 2$. *Consequently, there is no solution in* $[1,3]$. *Note that*

$$\frac{\partial f}{\partial y} = 2y \tag{14.21}$$

is unbounded (because y *is so) in* $[1,3]$. *The function* f *is not uniformly Lipschitz in* y *for* $t \in [1,3]$.

The initial value problem (14.1)-(14.2) can be reformulated as

$$y(t) = y(0) + \int_0^T f(s, y(s))ds. \tag{14.22}$$

This is an integral equation for the unknown function y. In particular, if f does not depend on y then the initial value problem (14.1)-(14.2) is reduced to the approximation of the definite integral

$$\int_0^T f(s)ds. \tag{14.23}$$

for which a numerical quadrature can be applied.

The numerical methods that we will study next deal with the more general and important case when f depends on the unknown y. They will produce an approximation to the exact solution of the initial value problem (assuming uniqueness) at a set of discrete points $0 = t_0 < t_1 < \ldots < t_N \leq T$. For simplicity in the presentation of the ideas we will assume that these points are equispaced

$$t_n = n\Delta t, \quad n = 0, 1, \ldots, N \text{ and } \Delta t = T/N. \tag{14.24}$$

Δt is called the time step or simply the step size. A numerical method for an initial value problem will be written as an algorithm to go from one discrete time t_n to the next t_{n+1}. With that in mind, it is useful to transform the ODE (or ODE system) into a integral equation by integrating from t_n to t_{n+1}:

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{tn+1} f(t, y(t))dt. \tag{14.25}$$

This equation provides a useful framework for the construction of numerical methods with the aid of quadratures.

14.2 A First Look at Numerical Methods

Let us denote by y_n the approximation to the exact solution at t_n, i.e.

$$y_n \approx y(t_n). \tag{14.26}$$

Starting from the integral formulation of the problem, eq. (14.25), if we approximate the integral using f evaluated at the lower integration limit

$$\int_{t_n}^{tn+1} f(t, y(t))dt \approx f(t_n, y(t_n))\,\Delta t \tag{14.27}$$

and replace the exact derivative of y at t_n by $f(t_n, y_n)$, we obtain the so called **forward Euler's method**:

$$y_0 = \alpha \tag{14.28}$$

$$y_{n+1} = y_n + \Delta t f(t_n, y_n), \quad n = 0, 1, \ldots N-1. \tag{14.29}$$

This provides an *explicit* formula to advance from one time step to the next. The approximation at the future step, y_{n+1}, only depends on the approximation at the current step, y_n. The forward Euler method is an example of an explict **one-step method**.

If on the other hand we approximate the integral in (14.25) using 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 get the **backward Euler method**:

$$y_0 = \alpha \tag{14.30}$$

$$y_{n+1} = y_n + \Delta t f(t_{n+1}, y_{n+1}), \quad n = 0, 1, \ldots N-1. \tag{14.31}$$

Note that now, y_{n+1} is defined *implicitly* in (14.31). Thus, to update the approximation, i.e. to obtain y_{n+1} for each $n = 0, 1, \ldots, N-1$, we need to solve a nonlinear equation (or a system of nonlinear equations if the initial value problem is for a system of ODEs). Since we expect the approximate solution not to change much in one time step (for sufficiently small Δt), y_n can be a good initial guess for a Newton iteration to find y_{n+1}. The backward Euler method is an implicit one-step method.

We can start with more accurate quadratures as the basis for our numerical methods. For example if we use the trapezoidal rule

$$\int_{t_n}^{tn+1} f(t, y(t))dt \approx \frac{\Delta t}{2}\left[f(t_n, y(t_n)) + f(t_{n+1}, y(t_{n+1}))\right] \tag{14.32}$$

and proceed as before we get the trapezoidal rule method:

$$y_0 = \alpha \tag{14.33}$$

$$y_{n+1} = y_n + \frac{\Delta t}{2}\left[f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right], \quad n = 0, 1, \ldots N-1. \tag{14.34}$$

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, that use intermediate approximations to the derivative (i.e. approximations to f) and a corresponding quadrature. For example, we can use the midpoint rule quadrature and the approximation

$$f\left(t_{n+1/2}, y(t_{n+1/2})\right) \approx f\left(t_{n+1/2}, y_n + \frac{\Delta t}{2} f(t_n, y_n)\right) \tag{14.35}$$

to obtain the explicit midpoint Runge -Kutta method

$$y_{n+1} = y_n + \Delta t f\left(t_{n+1/2}, y_n + \frac{\Delta t}{2} f(t_n, y_n)\right). \tag{14.36}$$

Another possibility is to approximate the integrand f in (14.25) by an interpolating polynomial using f evaluated at previous approximations y_n, y_{n-1}, ..., y_{n-m}. To simplify the notation, let us write

$$f_n = f(t_n, y_n). \tag{14.37}$$

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}), i.e

$$p_1(t) = \frac{(t - t_{n-1})}{\Delta t} f_n - \frac{(t - t_n)}{\Delta t} f_{n-1} \tag{14.38}$$

we get

$$\int_{t_n}^{tn+1} f(t, y(t))dt \approx \int_{t_n}^{tn+1} p_1(t)dt = \frac{\Delta t}{2}[3f_n - f_{n-1}] \tag{14.39}$$

and the corresponding numerical method is

$$y_{n+1} = y_n + \frac{\Delta t}{2}\left[3f_n - f_{n-1}\right], \quad n = 1, 2, \ldots N-1. \tag{14.40}$$

This is an example of a **multistep method**. To obtain the approximation at the future step we need the approximations of more than one step; in this particular case we need y_n and y_{n-1} so this is a two-step method. Note that to start using (14.40), i.e. $n = 1$, we need y_0 and y_1. For y_0 we can use the initial condition, $y_0 = \alpha$, and we can get y_1 by using a one-step method. All multistep methods require this *initialization process* when approximations to $y_1, \ldots, y_{m-1}$ have to be generated with one-step methods to begin using the multistep formula.

Numerical methods can also be constructed by approximating the derivative y' using finite differences or interpolation. For example, the central difference approximation

$$y'(t_n) \approx \frac{y(t_n + \Delta t) - y(t_n - \Delta t)}{2\Delta t} \approx \frac{y_{n+1} - y_{n-1}}{2\Delta t} \tag{14.41}$$

produces the two-step method

$$y_{n+1} = y_{n-1} + 2\Delta t f_n. \tag{14.42}$$

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 method known as **backward differentiation formula (BDF)** methods. For example, let p_2 be the polynomial of degree at most 2 that interpolates (t_{n-1}, y_{n-1}), (t_n, y_n), and (t_{n+1}, y_{n+1}). Then

$$y'(t_{n+1}) \approx p_2'(t_{n+1}) = \frac{3y_{n+1} - 4y_n + y_{n-1}}{2\Delta t}, \tag{14.43}$$

which gives the BDF method

$$\frac{3y_{n+1} - 4y_n + y_{n-1}}{2\Delta t} = f_{n+1}, \quad n = 1, 2, \ldots N - 1. \tag{14.44}$$

Note that this is an implicit, multistep method.

14.3 One-Step and Multistep Methods

As we have seen, there are two broad classes of methods for the initial value problem (14.1)-(14.2): one-step methods and multistep methods.

Explicit one-step methods can be written in the general form

$$y_{n+1} = y_n + \Delta t\, \Phi(t_n, y_n, \Delta t) \tag{14.45}$$

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.

A general, m-step multistep method can be cast as

$$\begin{aligned} a_m y_{n+1} + a_{m-1} y_n + \ldots + a_0 y_{n-m+1} = \\ \Delta t \left[b_m f_{n+1} + b_{m-1} f_n + \ldots + b_0 f_{n-m+1} \right], \end{aligned} \tag{14.46}$$

for some coefficients $a_0, a_1, \ldots, a_m$, with $a_m \neq 0$, and $b_0, b_1, \ldots, b_m$. If $b_m \neq 0$ the multistep is implicit otherwise it is explicit. Without loss of generality we will assume $a_m = 1$. Shifting the index by $m - 1$, we can write an m-step ($m \geq 2$) method as

$$\sum_{j=0}^{m} a_j y_{n+j} = \Delta t \sum_{j=0}^{m} b_j f_{n+j}. \tag{14.47}$$

14.4 Local and Global Error

When computing a numerical approximation of an initial value problem, at each time step there is an error associated to evolving the solution from t_n to t_{n+1} with the numerical method instead of using the ODE (or the integral equation) and there is also an error due to the use of y_n as starting point instead of the exact value $y(t_n)$. After several time steps, these *local* errors accumulate in the *global* error of the approximation to the initial value problem. Let us make the definition of these errors more precise.

Definition 14.1. *The global error e_n at a given discrete time t_n is given by*

$$e_n = y(t_n) - y_n, \tag{14.48}$$

where $y(t_n)$ and y_n are the exact solution of the initial value problem and the numerical approximation at t_n, respectively.

Definition 14.2. *The local truncation error τ_n, also called local discretization error is given by*

$$\tau_n = y(t_n) - y_n, \tag{14.49}$$

where y_n is computed with the numerical method using as starting value $y(t_{n-1})$ for a one-step method and $y(t_{n-1}), y(t_{n-2}), \ldots, y(t_{n-m})$ for an m-step method.

For an explicit one-step method the local truncation error is simply

$$\tau_{n+1} = y(t_{n+1}) - [y(t_n) + \Delta t\, \Phi(t_n, y(t_n), \Delta t)] \tag{14.50}$$

and for an explicit multistep method ($b_m = 0$) it follows immediately that ($a_m = 1$)

$$\tau_{n+m} = \sum_{j=0}^{m} a_j y(t_{n+j}) - \Delta t \sum_{j=0}^{m} b_j f(t_{n+j}, y(t_{n+j})) \tag{14.51}$$

or equivalently

$$\tau_{n+m} = \sum_{j=0}^{m} a_j y(t_{n+j}) - \Delta t \sum_{j=0}^{m} b_j y'(t_{n+j}), \tag{14.52}$$

where we have used $y' = f(t, y)$.

For implicit methods we can use also use (14.51) because it gives the local error up to a multiplicative factor. Indeed, let

$$T_{n+m} = \sum_{j=0}^{m} a_j y(t_{n+j}) - \Delta t \sum_{j=0}^{m} b_j f(t_{n+j}, y(t_{n+j})). \tag{14.53}$$

Then

$$\sum_{j=0}^{m} a_j y(t_{n+j}) = \Delta t \sum_{j=0}^{m} b_j f(t_{n+j}, y(t_{n+j})) + T_{n+m}. \tag{14.54}$$

On the other hand y_{n+m} in the definition of the local error is computed using

$$a_m y_{n+m} + \sum_{j=0}^{m-1} a_j y(t_{n+j}) = \Delta t \left[b_m f(t_{n+m}, y_{n+m}) + \sum_{j=0}^{m-1} b_j f(t_{n+j}, y(t_{n+j})) \right]. \tag{14.55}$$

Subtracting (14.54) to (14.55) and using $a_m = 1$ we get

$$y(t_{n+m}) - y_{n+m} = \Delta t\, b_m \left[f(t_{n+m}, y(t_{n+m})) - f(t_{n+m}, y_{n+m}) \right] + T_{n+k}. \tag{14.56}$$

Assuming f is a scalar C^1 function, from the mean value theorem we have

$$f(t_{n+m}, y(t_{n+m})) - f(t_{n+m}, y_{n+m}) = \frac{\partial f}{\partial y}(t_{n+m}, \eta_{n+m}) \left[y(t_{n+m}) - y_{n+m} \right], \tag{14.57}$$

for some η_{n+m} between $y(t_{n+m})$ and y_{n+m}. Substituting this into (14.56) and solving for $\tau_{n+m} = y(t_{n+m}) - y_{n+m}$ we get

$$\tau_{n+m} = \left(1 - \Delta t \, b_m \frac{\partial f}{\partial y}(t_{n+m}, \eta_{n+m}) \right)^{-1} T_{n+m}. \tag{14.58}$$

If f is a vector valued function (a system of ODEs) then the partial derivative in (14.58) 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, *we can also view the local truncation error as a measure of how well the exact solution of the initial value problem satisfies the numerical method formula.*

Example 14.8. *The local truncation error for the forward Euler method is*

$$\tau_{n+1} = y(t_{n+1}) - \left[y(t_n) + \Delta t \, f(t_n, y(t_n)) \right]. \tag{14.59}$$

Taylor expanding the exact solution around t_n we have

$$y(t_{n+1}) = y(t_n) + \Delta t \, y'(t_n) + \frac{1}{2}(\Delta t)^2 y''(\eta_n) \tag{14.60}$$

for some η_n between t_n and t_{n+1}. Using $y' = f$ and substituting (14.59) into (14.60) we get

$$\tau_{n+1} = \frac{1}{2}(\Delta t)^2 y''(\eta_n). \tag{14.61}$$

Thus, assuming the exact solution is C^2, the local truncation error of the forward Euler method is $O(\Delta t)^2$ [1].

To simplify notation we will henceforth write $O(\Delta t)^k$ instead of $O((\Delta t)^k)$.

[1] To simplify notation we write $O(\Delta t)^2$ to mean $O((\Delta t)^2)$ and similarly for other orders of Δt.

Example 14.9. *For the explicit midpoint Runge -Kutta method we have*

$$\tau_{n+1} = y(t_{n+1}) - y(t_n) - \Delta t f\left(t_{n+1/2}, y(t_n) + \frac{\Delta t}{2} f(t_n, y(t_n))\right). \quad (14.62)$$

Taylor expanding f around $(t_n, y(t_n))$ we obtain

$$\begin{aligned} f\left(t_{n+1/2}, y(t_n) + \frac{\Delta t}{2} f(t_n, y(t_n))\right) &= f(t_n, y(t_n)) \\ &+ \frac{\Delta t}{2}\frac{\partial f}{\partial t}(t_n, y(t_n)) \\ &+ \frac{\Delta t}{2} f(t_n, y(t_n))\frac{\partial f}{\partial y}(t_n, y(t_n)) + O(\Delta t)^2. \end{aligned} \quad (14.63)$$

But $y' = f$, $y'' = f'$ and

$$f' = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y} y' = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y} f. \quad (14.64)$$

Therefore

$$f\left(t_{n+1/2}, y(t_n) + \frac{\Delta t}{2} f(t_n, y(t_n))\right) = y'(t_n) + \frac{1}{2}\Delta t\, y''(t_n) + O(\Delta t)^2. \quad (14.65)$$

On the other hand

$$y(t_{n+1}) = y(t_n) + \Delta t\, y'(t_n) + \frac{1}{2}(\Delta t)^2 y''(t_n) + O(\Delta t)^3. \quad (14.66)$$

Substituting (14.65) and (14.66) into (14.62) we get

$$\tau_{n+1} = O(\Delta t)^3. \quad (14.67)$$

In the previous two examples the methods are one-step. We know obtain the local truncation error for a multistep method.

Example 14.10. *Let us consider the 2-step Adams-Bashforth method (14.40). We have*

$$\tau_{n+2} = y(t_{n+2}) - y(t_{n+1}) - \frac{\Delta t}{2}\left[3f(t_{n+1}, y(t_{n+1})) - f(t_n, y(t_n))\right] \quad (14.68)$$

and using $y' = f$

$$\tau_{n+2} = y(t_{n+2}) - y(t_{n+1}) - \frac{\Delta t}{2}\left[3y'(t_{n+1}) - y'(t_n)\right]. \tag{14.69}$$

Taylor expanding $y(t_{n+2})$ *and* $y'(t_n)$ *around* t_{n+1} *we have*

$$y(t_{n+2}) = y(t_{n+1}) + \Delta t y'(t_{n+1}) + \frac{1}{2}(\Delta t)^2 y''(t_{n+1}) + O(\Delta t)^3, \tag{14.70}$$

$$y'(t_n) = y'(t_{n+1}) - \Delta t y''(t_{n+1}) + O(\Delta t)^2. \tag{14.71}$$

Substituting these expressions into (14.69) we get

$$\begin{aligned} \tau_{n+2} &= \Delta t y'(t_{n+1}) + \frac{1}{2}(\Delta t)^2 y''(t_{n+1}) \\ &\quad - \frac{\Delta t}{2}\left[2y'(t_{n+1}) - \Delta t y''(t_{n+1})\right] + O(\Delta t)^3 \\ &= O(\Delta t)^3. \end{aligned} \tag{14.72}$$

14.5 Order of a Method and Consistency

We saw in the previous section that assuming sufficient smoothness of the exact solution of the initial value problem, the local truncation error can be expressed as $O(\Delta t)^k$, for some positive integer k. If the local truncation error accumulates no worse than linearly as we march up N steps to get the approximate solution at $T = N\Delta t$, the global error would be order $N(\Delta t)^k = \frac{T}{\Delta t}(\Delta t)^k$, i.e. $O(\Delta t)^{k-1}$. So we need $k \geq 2$ for all methods and to prevent uncontrolled accumulation of the local errors as $n \to \infty$ we need the methods to be *stable* in a sense we will make more precise later. This motivates the following definitions.

Definition 14.3. *A numerical method for the initial value problem (14.1)-(14.2) is said to be of order* p *if its local truncation error is* $O(\Delta t)^{p+1}$.

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.

Definition 14.4. *We say that a numerical method is consistent (with the ODE of the initial value problem) if the method is at least of order 1.*

For the case of one-step methods we have

$$\begin{aligned}\tau_{n+1} &= y(t_{n+1}) - [y(t_n) + \Delta t\, \Phi(t_n, y(t_n), \Delta t)] \\ &= \Delta t\, y'(t_n) - \Delta t\, \Phi(t_n, y(t_n), \Delta t) + O(\Delta t)^2 \\ &= \Delta t\, [f(t_n, y(t_n)) - \Phi(t_n, y(t_n), \Delta t)] + O(\Delta t)^2.\end{aligned} \tag{14.73}$$

Thus, assuming the increment function Φ is continuous, a one-step method is consistent with the ODE $y' = f(t, y)$ if and only if

$$\Phi(t, y, 0) = f(t, y). \tag{14.74}$$

To find a consistency condition for a multistep method, we expand $y(t_{n+j})$ and $y'(t_{n+j})$ around t_n

$$y(t_{n+j}) = y(t_n) + (j\Delta t)y'(t_n) + \frac{1}{2!}(j\Delta t)^2 y''(t_n) + \dots \tag{14.75}$$

$$y'(t_{n+j}) = y'(t_n) + (j\Delta t)y''(t_n) + \frac{1}{2!}(j\Delta t)^2 y'''(t_n) + \dots \tag{14.76}$$

and substituting in the definition of the local error (14.51) we get that for a multistep method is consistent if and only if

$$a_0 + a_1 + \ldots + a_m = 0, \tag{14.77}$$

$$a_1 + 2a_2 + \ldots m a_m = b_0 + b_1 + \ldots + b_m. \tag{14.78}$$

All the methods that we have seen so far are consistent (with $y' = f(t, y)$).

14.6 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 as $\Delta t \to 0$.

Definition 14.5. *A numerical method for the initial value (14.1)-(14.2) is convergent if the global error at a given $t = n\Delta t$ converges to zero as $\Delta t \to 0$ and $t = n\Delta t$ i.e.*

$$\lim_{\substack{\Delta t \to 0 \\ n\Delta t = t}} [y(n\Delta t) - y_n] = 0. \tag{14.79}$$

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 (14.50) of the local truncation error τ, the exact solution satisfies

$$y(t_{n+1}) = y(t_n) + \Delta t\, \Phi(t_n, y(t_n), \Delta t) + \tau_{n+1} \tag{14.80}$$

while the approximation is given by

$$y_{n+1} = y_n + \Delta t\, \Phi(t_n, y_n, \Delta t). \tag{14.81}$$

Subtracting (14.81) from (14.80) we get a difference equation for the global error

$$e_{n+1} = e_n + \Delta t\, [\Phi(t_n, y(t_n), \Delta t) - \Phi(t_n, y_n, \Delta t)] + \tau_{n+1}. \tag{14.82}$$

This the growth of the global error as we take more and more time steps is linked not only to the local error but also to the increment function Φ. Let us suppose that Φ is Lipschitz in y, i.e. there is $L \geq 0$ such that

$$|\Phi(t, y_1, \Delta t) - \Phi(t, y_2, \Delta t)| \leq L|y_1 - y_2| \tag{14.83}$$

for all $t \in [0, T]$ and y_1 and y_2 in the relevant domain of existence of the solution. Recall that for the Euler method $\Phi(t, y, \Delta t) = f(t, y)$ and we assume $f(t, y)$ is Lipschitz in y to guarantee existence and uniqueness of the initial value problem so the Lipschitz assumption on Φ is somewhat natural. Then, taking absolute values (or norms in the vector case), using the triangle inequality and (14.83) we obtain

$$|e_{n+1}| \leq (1 + \Delta t L)|e_n| + |\tau_{n+1}|. \tag{14.84}$$

For a method of order p, $|\tau_{n+1}| \leq C(\Delta t)^{p+1}$, for sufficiently small Δt. Therefore,

$$\begin{aligned} |e_{n+1}| &\leq (1 + \Delta t L)|e_n| + C(\Delta t)^{p+1} \\ &\leq (1 + \Delta t L)\left[(1 + \Delta t L)|e_{n-1}| + C(\Delta t)^{p+1}\right] + C(\Delta t)^{p+1} \\ &\leq \dots \\ &\leq (1 + \Delta t L)^{n+1}|e_0| + C(\Delta t)^{p+1} \sum_{j=0}^{n} (1 + \Delta t L)^j \end{aligned} \tag{14.85}$$

and summing up the geometry sum we get

$$|e_{n+1}| \leq (1 + \Delta t L)^{n+1}|e_0| + \left[\frac{(1 + \Delta t L)^{n+1} - 1}{\Delta t L}\right] C(\Delta t)^{p+1}. \tag{14.86}$$

Now $1 + t \leq e^t$ for all real t and consequently $(1 + \Delta tL)^n \leq e^{n\Delta tL}$. Thus, since $e_0 = 0$,

$$|e_n| \leq \left[\frac{e^{n\Delta tL} - 1}{\Delta tL}\right] C(\Delta t)^{p+1} < \frac{C}{L} e^{n\Delta tL} (\Delta t)^p. \tag{14.87}$$

Thus, the global error goes to zero as $\Delta t \to 0$, keeping $t = n\Delta t$ fixed and we obtain the following important result.

Theorem 14.2. *A consistent ($p \geq 1$) one-step method with a Lipschitz in y increment function $\Phi(t, y, \Delta t)$ is convergent.*

The Lipschitz condition on Φ gives *stability* to the method in the sense that this condition allows us to control the growth of the global error in the limit as $\Delta t \to 0$ and $n \to \infty$.

Example 14.11. *As we have seen the forward Euler method is oder 1 and hence consistent. Since $\Phi = f$ and we are assuming that f is Lipschitz in y then by the previous theorem the forward Euler method is convergent.*

Example 14.12. *Prove that the midpoint Runge-Kutta method is convergent (assuming f is Lipschitz in y).*

The increment function in this case is

$$\Phi(t, y, \Delta t) = f\left(t + \frac{\Delta t}{2}, y + \frac{\Delta t}{2} f(t, y)\right). \tag{14.88}$$

Therefore

$$\begin{aligned} |\Phi(t, y_1, \Delta t) - \Phi(t, y_2, \Delta t)| &= \left| f\left(t + \frac{\Delta t}{2}, y_1 + \frac{\Delta t}{2} f(t, y_1)\right) \right. \\ &\quad \left. - f\left(t + \frac{\Delta t}{2}, y_2 + \frac{\Delta t}{2} f(t, y_2)\right) \right| \\ &\leq L \left| y_1 + \frac{\Delta t}{2} f(t, y_1) - y_2 - \frac{\Delta t}{2} f(t, y_2) \right| \\ &\leq L |y_1 - y_2| + \frac{\Delta t}{2} L |f(t, y_1) - f(t, y_2)| \\ &\leq \left(1 + \frac{\Delta t}{2} L\right) L |y_1 - y_2| \leq \tilde{L} |y_1 - y_2|. \end{aligned} \tag{14.89}$$

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 and hence consistent, it is convergent.

The exact solution of the initial value problem 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 look at the most important class of one-step methods: the Runge-Kutta methods.

14.7 Runge-Kutta Methods

As seen earlier Runge-Kutta (RK) methods are based on replacing the integral in

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{tn+1} f(t, y(t))dt \tag{14.90}$$

with a quadrature formula and using accurate enough intermediate approximations for integrand f (the derivative of y). For example if we use the trapezoidal rule quadrature

$$\int_{t_n}^{tn+1} f(t, y(t))dt \approx \frac{\Delta t}{2}\left[f(t_n, y(t_n)) + f(t_{n+1}, y(t_{n+1})\right] \tag{14.91}$$

and the approximations $y(t_n) \approx y_n$ and $y_{n+1} \approx y_n + \Delta t f(t_n, y_n)$, we obtain a two-stage RK method known as improved Euler method

$$K_1 = f(t_n, y_n), \tag{14.92}$$

$$K_2 = f(t_n + \Delta t, y_n + \Delta t K_1), \tag{14.93}$$

$$y_{n+1} = y_n + \Delta t\left[\frac{1}{2}K_1 + \frac{1}{2}K_2\right]. \tag{14.94}$$

Note that K_1 and K_2 are approximations to the derivative of y.

Example 14.13. *The midpoint RK method (14.36) is also a two-stage RK method and can be written as*

$$K_1 = f(t_n, y_n), \tag{14.95}$$

$$K_2 = f\left(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2}K_1\right), \tag{14.96}$$

$$y_{n+1} = y_n + \Delta t K_2. \tag{14.97}$$

We know from Example 14.9 that the midpoint RK is order 2. The improved order method is also order 2. Obtaining the order of a RK method using Taylor expansions becomes a long, tedious process because number of terms in the derivatives of f rapidly grow ($y' = f, y'' = f_t + f_y f, y''' = f_{tt} + 2f_{ty}f + f_{yy}f^2 + f_y f_t + f_y^2 f$, etc.).

The most popular RK method is the following 4-stage (and fourth order) explicit RK, known as the classical fourth order RK

$$\begin{aligned} K_1 &= f(t_n, y_n), \\ K_2 &= f(t_n + \frac{1}{2}\Delta t, y_n + \frac{1}{2}\Delta t K_1), \\ K_3 &= f(t_n + \frac{1}{2}\Delta t, y_n + \frac{1}{2}\Delta t K_2), \\ K_4 &= f(t_n + \Delta t, y_n + \Delta t K_3), \\ y_{n+1} &= y_n + \frac{\Delta t}{6}\left[K_1 + 2K_2 + 2K_3 + K_4\right]. \end{aligned} \tag{14.98}$$

A general s-stage RK method can be written as

$$\begin{aligned} K_1 &= f\left(t_n + c_1\Delta t, y_n + \sum_{j=1}^{s} a_{1j}K_j\right), \\ K_2 &= f\left(t_n + c_2\Delta t, y_n + \sum_{j=1}^{s} a_{2j}K_j\right), \\ &\vdots \\ K_s &= f\left(t_n + c_s\Delta t, y_n + \sum_{j=1}^{s} a_{sj}K_j\right), \\ y_{n+1} &= y_n + \Delta t \sum_{j=1}^{s} b_j K_j. \end{aligned} \tag{14.99}$$

RK methods are determined by the constants $c_1, \ldots, c_s$ that specify the quadrature points, 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 conditions

$$c_i = \sum_{j=1}^{s} a_{ij}, \tag{14.100}$$

$$\sum_{j=1}^{s} b_j = 1. \tag{14.101}$$

To define a RK method it is enough to specify the coefficients c_j, a_{ij} and b_j for $i, j = 1, \ldots, s$. There coefficients are often displayed in a table, called the Butcher tableau (after J.C. Butcher) of the RK method as shown in Table 14.1. For an *explicit RK method*, the matrix of coefficients $A = (a_{ij})$

Table 14.1: Butcher tableau for a general RK method.

c_1	a_{11}	$\ldots$	a_{1s}
$\vdots$	$\vdots$	$\vdots$	$\vdots$
c_s	a_{s1}	$\ldots$	a_{ss}
	b_1	$\ldots$	b_s

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 tableau.

Example 14.14. *The tables 14.2-14.4 show the Butcher tableau of some explicit RK methods.*

Table 14.2: Improved Euler.

0		
1	1	
	$\frac{1}{2}$	$\frac{1}{2}$

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 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*

Table 14.3: Midpoint RK.

0		
$\frac{1}{2}$	$\frac{1}{2}$	
	0	1

Table 14.4: Classical fourth order RK.

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

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 method or SDIRK.

Example 14.15. *Tables 14.5-14.8 show some examples of DIRK and SDIRK methods.*

Table 14.5: Backward Euler.

1	1
	1

Table 14.6: Implicit mid-point rule RK.

$\frac{1}{2}$	$\frac{1}{2}$
	1

Table 14.7: Hammer and Hollingworth DIRK.

0	0	0
$\frac{2}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
	$\frac{1}{4}$	$\frac{3}{4}$

Table 14.8: Two-stage order 3 SDIRK ($\gamma = \frac{3\pm\sqrt{3}}{6}$).

γ	γ	0
$1-\gamma$	$1-2\gamma$	γ
	$\frac{1}{2}$	$\frac{1}{2}$

14.8 Adaptive Stepping

So far we have considered a fixed Δt throughout the entire computation of an approximation to the initial value problem of an ODE or of a systems 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 use 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

$$e_{n+1} \approx w_{n+1} - y_{n+1}. \tag{14.102}$$

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 $|y_{n+1} - w_{n+1}| > \delta$, we decrease Δt (e.g. we set it to $\Delta t/2$), recompute y_{n+1}, obtain the new estimate of the error (14.102), 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 limited stability properties.

14.9 Embedded Methods

For computational efficiency, adaptive stepping as described above is implemented reusing as much as possible evaluations of f, the right hand side of $y' = f(t, y)$ because evaluating f is the most expensive part of Runge-Kutta methods. So the idea is to embed, with minimal additional f evaluations, a Runge-Kutta method inside another. The following example illustrates this idea.

Consider the explicit trapezoidal method (second order) and the Euler method (first order). We can embed them as follows

$$K_1 = f(t_n, y_n), \tag{14.103}$$
$$K_2 = f\left(t_n + \Delta t, y_n + \Delta t K_1\right), \tag{14.104}$$
$$w_{n+1} = y_n + \Delta t \left[\frac{1}{2}K_1 + \frac{1}{2}K_2\right], \tag{14.105}$$
$$y_{n+1} = y_n + \Delta t K_1. \tag{14.106}$$

Note that the approximation of the derivative K_1 is used for both methods. The computation of the higher order method (14.105) only costs an additional evaluation of f.

14.10 Multistep Methods

As we have seen, multistep methods use approximations from more than one step to update the new approximation. They can be written in the general form

$$\sum_{j=0}^{m} a_j y_{n+j} = \Delta t \sum_{j=0}^{m} b_j f_{n+j}. \tag{14.107}$$

where $m \geq 2$ is the number of previous steps the method employs.

Multistep methods only require one evaluation of f per step because the other, previously computed, values of f are stored. Thus, multistep methods have generally lower computational cost than one-step methods of the same order. The trade-off is reduced stability as we will see later.

We saw in Section 14.2 the approaches of interpolation and finite differences to construct 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, derived both from interpolation, are the most commonly used multistep methods. These are the (explicit) Adams-Bashforth methods and the (implicit) Adams-Moulton methods.

14.10.1 Adams Methods

We constructed in Section 14.2 the two-step Adams-Barshforth method

$$y_{n+1} = y_n + \frac{\Delta t}{2}\left[3f_n - f_{n-1}\right], \quad n = 1, 2, \ldots N-1.$$

An m-step explicit Adams method, also called Adams-Bashforth, can be derived by starting with the integral formulation of the initial value problem

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t))dt. \tag{14.108}$$

and replacing the integral with that of the interpolating polynomial p 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

$$p(t) = \sum_{j=n-m+1}^{n} l_j(t) f_j, \tag{14.109}$$

where

$$l_j(t) = \prod_{\substack{k=n-m+1 \\ k \neq j}}^{n} \frac{(t - t_k)}{(t_j - t_k)}, \quad \text{for } j = n-m+1, \ldots, n. \tag{14.110}$$

Thus, the m-step explicit Adams method has the form

$$y_{n+1} = y_n + \Delta t\left[b_{m-1}f_n + b_{m-2}f_{n-1} + \ldots b_0 f_{n-m+1}\right], \tag{14.111}$$

where

$$b_{j-(n-m+1)} = \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} l_j(t)dt, \quad \text{for } j = n-m+1, \ldots, n. \tag{14.112}$$

Here are the first three explicit Adams methods, 2-step, 3-step, and 4-step, respectively:

$$y_{n+1} = y_n + \frac{\Delta t}{2}\left[3f_n - f_{n-1}\right], \tag{14.113}$$

$$y_{n+1} = y_n + \frac{\Delta t}{12}\left[23f_n - 16f_{n-1} + 5f_{n-2}\right], \tag{14.114}$$

$$y_{n+1} = y_n + \frac{\Delta t}{24}\left[55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}\right]. \tag{14.115}$$

The implicit Adams methods, also called Adams-Moulton methods, are derived by including also (t_{n+1}, f_{n+1}) in the interpolation. That is, p 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:

$$y_{n+1} = y_n + \frac{\Delta t}{12}\left[5f_{n+1} + 8f_n - f_{n-1}\right], \tag{14.116}$$

$$y_{n+1} = y_n + \frac{\Delta t}{24}\left[9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}\right], \tag{14.117}$$

$$\begin{aligned} y_{n+1} = y_n + \frac{\Delta t}{720}\big[&251f_{n+1} \\ &+ 646f_n - 264f_{n-1} + 106f_{n-2} - 19f_{n-3}\big]. \end{aligned} \tag{14.118}$$

14.10.2 Zero-Stability and Dahlquist Theorem

Recall that we can write a general multistep method as

$$\sum_{j=0}^{m} a_j y_{n+j} = \Delta t \sum_{j=0}^{m} b_j f_{n+j}.$$

Stability is related to the notion of boundedness of the numerical approximation in the limit as $\Delta t \to 0$. Thus, it is natural to consider the equation:

$$a_m y_{n+m} + a_{m-1} y_{n+m-1} + \ldots a_0 y_n = 0. \tag{14.119}$$

This is a linear difference equation. Let us look for solutions of the form $y_n = \xi^n$. Plugging in (14.119) we get

$$\xi^n \left[a_m \xi^m + a_{m-1}\xi^{m-1} + \ldots a_0\right] = 0, \tag{14.120}$$

which implies that ξ is a root of the polynomial

$$\rho(z) = a_m z^m + a_{m-1} z^{m-1} + \ldots a_0. \tag{14.121}$$

If ρ has m distinct roots $\xi_1, \xi_2, \ldots, \xi_m$ then the general solution of (14.119) is

$$y_n = c_1 \xi_1^n + c_2 \xi_2^n + \ldots + c_m \xi_m^n, \tag{14.122}$$

where $c_1, c_2, \ldots, c_m$ are determined uniquely from the initialization values $y_0, y_1, \ldots, y_{m-1}$.

If the roots are not all distinct, the solution of (14.119) 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 (14.119). Let us check this is indeed the case. Substituting $y_n = n\xi_1^n$ in (14.119) we get

$$\begin{aligned} &a_m(n+m)\xi_1^{n+m} + a_{m-1}(n+m-1)\xi_1^{n+m-1} + \ldots + a_0 n\xi_1^n \\ &\quad = \xi_1^n \left[a_m(n+m)\xi_1^m + a_{m-1}(n+m-1)\xi_1^{n+m-1} + \ldots + a_0 n\right] \\ &\quad = \xi_1^n \left[n\rho(\xi_1) + \xi_1 \rho'(\xi_1)\right] = 0. \end{aligned} \tag{14.123}$$

Thus, in this case of a double root, the general solution of (14.119) is

$$y_n = c_1 \xi_1^n + c_2 n\xi_1^n + c_3 \xi_3^n + \ldots + c_m \xi_m^n. \tag{14.124}$$

If there is a triple root, say $z_1 = z_2 = z_3$, then the general solution of (14.119) is given by

$$y_n = c_1 \xi_1^n + c_2 n\xi_1^n + c_3 n(n-1)\xi_1^n + \ldots + c_m \xi_m^n. \tag{14.125}$$

Clearly, for the numerical approximation y_n to remain bounded as $n \to 0$ we need that all the roots $\xi_1, \xi_2, \ldots, \xi_m$ of ρ satisfy:

(a) $|\xi_i| \leq 1$, for all $i = 1, 2, \ldots, m$.

(b) If ξ_i is a root of multiplicity greater than one then $|\xi_i| < 1$.

Conditions (a) and (b) above are known as the root condition.

Definition 14.6. *A multistep method is zero-stable (or D-stable) if the zeros of ρ satisfy the root condition.*

Clearly, zero-stability is a necessary condition for convergence of a multistep method. It is remarkable that it is also a sufficient condition for *consistent* multistep methods. This is the content of the following fundamental theorem due to Dahlquist.

Theorem 14.3. *(Dahlquist Theorem) A consistent multistep method is convergent if and only if it is zero-stable.*

14.11 A-Stability

So far we have talked about numerical stability in the sense of boundedness of the numerical approximation in the limit as $\Delta t \to 0$. There is another type of numerical stability which give us some guidance on the actual size Δt one can take for a stable computation using a given numerical method for an ODE initial value problem. This type of stability is called linear stability, absolute stability, or A-stability. It is based on the behavior of a numerical method for the simple linear problem:

$$y' = \lambda y, \tag{14.126}$$

$$y(0) = 1, \tag{14.127}$$

where λ is a complex number. The exact solution is $y(t) = e^{\lambda t}$. Let us look at forward Euler method applied to this model problem. We have

$$y_{n+1} = y_n + \Delta t \lambda y_n = (1 + \Delta t \lambda) y_n \tag{14.128}$$

$$= (1 + \Delta t \lambda)(1 + \Delta t \lambda) y_{n-1} = (1 + \Delta t \lambda)^2 y_{n-1} \tag{14.129}$$

$$= \ldots = (1 + \Delta t \lambda)^{n+1} y_0 = (1 + \Delta t \lambda)^{n+1}. \tag{14.130}$$

Thus, the forward Euler solution is $y_n = (1+\Delta t\lambda)^n$. Clearly, in order for this numerical approximation to remain bounded as $n \to \infty$ (long time behavior) we need

$$|1 + \Delta t \lambda| \leq 1. \tag{14.131}$$

Denoting $z = \Delta t \lambda$, the set

$$\mathcal{S} = \{ z \in \mathbb{C} : |1 + z| \leq 1 \}, \tag{14.132}$$

i.e. the unit disk centered at -1 is the region of linear stability or A-stability of the forward Euler method.

Runge-Kutta methods applied to the linear problem (14.126) produce a solution of the form

$$y_{n+1} = R(\Delta t\lambda)y_n, \tag{14.133}$$

where R is a rational function, $R(z) = \frac{P(z)}{Q(z)}$, where P ands Q are polynomials. In particular, when the Runge-Kutta method is explicit R is just a polynomial. Therefore, for a Runge-Kutta method the region of A-stability is given by the set

$$\mathcal{S} = \{z \in \mathbb{C} : |R(z)| \leq 1\}. \tag{14.134}$$

R is called the *stability function* of the Runge-Kutta method.

Example 14.16. *The implicit trapezoidal rule method. We have*

$$y_{n+1} = y_n + \frac{\Delta t}{2}(\lambda y_n + \lambda y_{n+1}) \tag{14.135}$$

and solving for y_{n+1} we get

$$y_{n+1} = \left[\frac{1 + \frac{\Delta t}{2}\lambda}{1 - \frac{\Delta t}{2}\lambda}\right] y_n. \tag{14.136}$$

so the region of A-stability of the (implicit) trapezoidal rule method is the set complex numbers z such that

$$\left|\frac{1 + \frac{z}{2}}{1 - \frac{z}{2}}\right| \leq 1 \tag{14.137}$$

and this is the entire left half complex plane, $Re\{z\} \leq 0$.

Example 14.17. *The improved Euler method. We have*

$$y_{n+1} = y_n + \frac{\Delta t}{2}[\lambda y_n + \lambda(y_n + \Delta\lambda y_n),] \tag{14.138}$$

that is

$$y_{n+1} = \left[1 + \Delta t\lambda + \frac{1}{2}(\Delta t\lambda)^2\right] y_n. \tag{14.139}$$

The stability function is therefore $R(z) = 1 + z + \frac{z^2}{2}$ *and the set of linear stability consists of all the complex numbers such* $|R(z)| \leq 1$. *Note that*

$$R(z) = e^z + O(z^3). \tag{14.140}$$

That is, R *approximates* $e^{\Delta t\lambda}$ *to third order in* Δt *as it should because the method is second order (local truncation error is* $O(\Delta t)^3$.

Example 14.18. *The backward Euler method. In this case we have,*

$$y_{n+1} = y_n + \Delta t\lambda y_{n+1}, \tag{14.141}$$

and solving for y_{n+1} *we obtain*

$$y_{n+1} = \left[\frac{1}{1 - \Delta t\lambda}\right] y_n \tag{14.142}$$

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.*

Definition 14.7. *A method is called A-stable is 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 applied an m-step ($m > 1$) method to the linear ODE (14.126) we get

$$\sum_{j=0}^{m} a_j y_{n+j} - \Delta t\lambda \sum_{j=0}^{m} b_j y_{n+j} = 0. \tag{14.143}$$

This is a constant coefficients, linear difference equation. We look for solutions of this equation in the form $y_n = \xi^n$. Substituting into (14.143) we have

$$\xi^n \sum_{j=0}^{m} (a_j - \Delta t\lambda b_j)\xi^j = 0, \tag{14.144}$$

which implies that ξ is a root of the polynomial

$$\Pi(\xi; z) = (a_m - zb_m)\xi^m + (a_{m-1} - zb_{m-1})\xi^{m-1} + \ldots + (a_0 - zb_0) \tag{14.145}$$

where $z = \Delta t\lambda$. We are going to write this polynomial in terms of the polynomials defined by the coefficients of the multistep method:

$$\rho(\xi) = a_m\xi^m + a_{m-1}\xi^{m-1} + \ldots + a_0, \tag{14.146}$$
$$\sigma(\xi) = b_m\xi^m + b_{m-1}\xi^{m-1} + \ldots + b_0. \tag{14.147}$$

We have

$$\Pi(\xi; z) = \rho(\xi) - z\sigma(\xi). \tag{14.148}$$

Hence, in order for the numerical approximation y_n to remain bounded we need that all the roots of Π satisfy the root condition.

Definition 14.8. *The region of A-stability of a linear multistep method is the set*

$$\mathcal{S} = \{z \in \mathbb{C} : \textit{all the roots of } \Pi(\xi; z) \textit{ satisfy the root condition}\}. \tag{14.149}$$

Recall that consistency for a multistep method translate into the following conditions:

$$a_0 + a_1 + \ldots + a_m = 0,$$
$$a_1 + 2a_2 + \ldots ma_m = b_0 + b_1 + \ldots + b_m,$$

or in terms of the multistep method polynomials:

$$\rho(1) = 0, \tag{14.150}$$
$$\rho'(1) = \sigma(1). \tag{14.151}$$

The first condition implies that $\Pi(1; 0) = 0$. Therefore, by the implicit function theorem Π has a root for z in the neighborhood of zero. Such root is called the principal root of the multistep method. Multistep ($m > 1$) methods have one or more additional roots and these are called parasitic roots.

Example 14.19. *Consider the 2-step method*

$$y_{n+1} + 4y_n - 5y_{n-1} = \Delta t\,[\,4f_n + 2f_{n-1}]\,. \tag{14.152}$$

Then

$$\rho(\xi) = \xi^2 + 4\xi - 5 = (\xi - 1)(\xi + 5), \tag{14.153}$$
$$\sigma(\xi) = 4\xi + 2. \tag{14.154}$$

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 zero-stable. Therefore, by Dahlquist theorem, it is not convergent. For the polynomial Π we have

$$\Pi(\xi, z) = \xi^2 + 4(1 - z)\xi - (5 + 2z), \tag{14.155}$$

which has roots

$$\xi_{\pm} = -2(1 - z) \pm \sqrt{9 - 6z + 4z^2} \tag{14.156}$$

and expanding for small $|z|$ we have

$$\xi_{-} = 1 + z + O(z^2) \quad \textit{principal root.} \tag{14.157}$$

$$\xi_{+} = -5 + O(z) \quad \textit{parasitic root.} \tag{14.158}$$

14.12 Stiff ODEs