MATH3474 Numerical Methods

Professor M A Kelmanson

(revised by Dr. Yue-Kin Tsang)

Contents

0 Preliminaries

0.1 Useful formulae

0.1.1 Binomial Theorem

$$(a+b)^n = \sum_{m=0}^{n} \binom{n}{m} a^{n-m}b^m = \sum_{m=0}^{n} \frac{n!}{m!(n-m)!} a^{n-m}b^m,$$

valid for all integers $n > 0$. If the exponent, ν say, of $(a+b)$ is not a positive integer, then

$$(a+b)^\nu = a^\nu + \nu a^{\nu-1}b + \frac{\nu(\nu-1)}{2!}a^{\nu-2}b^2 + \frac{\nu(\nu-1)(\nu-2)}{3!}a^{\nu-3}b^3 + \cdots,$$

which converges, by the ratio test, only if $|b/a| < 1$. Useful examples are:

$$\sqrt{a+b} = (a+b)^{1/2} = a^{1/2} + \frac{1}{2}a^{-1/2}b - \frac{1}{8}a^{-3/2}b^2 + \frac{1}{16}a^{-5/2}b^3 + \cdots;$$

$$\frac{1}{\sqrt{a+b}} = (a+b)^{-1/2} = a^{-1/2} - \frac{1}{2}a^{-3/2}b + \frac{3}{8}a^{-5/2}b^2 - \frac{5}{16}a^{-7/2}b^3 + \cdots;$$

$$\frac{1}{a+b} = (a+b)^{-1} = a^{-1} - a^{-2}b + a^{-3}b^2 - a^{-4}b^3 + \cdots.$$

0.1.2 Taylor expansion

If a function $u(x)$ is infinitely differentiable at $x = x_0$, we can express it as the power series, with coefficients a_n,

$$u(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)^2 + a_3(x-x_0)^3 + \cdots = \sum_{n=0}^{\infty} a_n(x-x_0)^n.$$

Differentiating this series repeatedly with respect to x, we obtain

$$u'(x) = a_1 + 2a_2(x-x_0) + 3a_3(x-x_0)^2 + \cdots,$$

$$u''(x) = 2a_2 + 6a_3(x-x_0) + \cdots, \quad u'''(x) = 6a_3 + \cdots, \text{ etc.}$$

Using $u^{(n)}(x)$ to denote the nth derivative d^nf/dx^n (with $u^{(0)}(x)$ understood to mean $u(x)$), we can see that the coefficients a_n are found by setting $x = x_0$ in the above differentiated expressions to give

$$u^{(n)}(x_0) = n!\,a_n \;\Rightarrow\; a_n = \frac{u^{(n)}(x_0)}{n!}, \;\; n = 1, ..., \infty.$$

Thus we have the **Taylor expansion of** $u(x)$, with **centre** x_0,

$$u(x) = \sum_{n=0}^{\infty} \frac{u^{(n)}(x_0)}{n!}(x-x_0)^n.$$

The maximum value of $|x - x_0|$ for which the infinite expansion converges is called the **radius of convergence**, denoted by R, calculated as either $1/L_1$ from the ratio test, or $1/L_2$ from the nth-root test.

If now ξ is a point lying between x and x_0, then the **truncated Taylor expansion** with **remainder** is

$$u(x) = \sum_{n=0}^{N} \frac{u^{(n)}(x_0)}{n!}(x - x_0)^n + R_N(x), \tag{0.1}$$

where the *remainder* or *truncation error* $R_N(x)$ is given by

$$R_N(x) = \frac{u^{(N+1)}(\xi)}{(N+1)!}(x - x_0)^{N+1}.$$

The form (0.1) of the Taylor expansion allows us both to calculate only a finite number of terms and to examine the consequences of neglecting the remainder.

If $x_0 = 0$ we have a **Maclaurin series**, common examples of which are:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!}, \quad \text{all } x;$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!}, \quad \text{all } x;$$

$$\exp x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{n=0}^{\infty} \frac{x^n}{n!}, \quad \text{all } x; \tag{0.2}$$

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots = \sum_{n=1}^{\infty} \frac{(-1)^{n+1} x^n}{n}, \quad |x| < 1. \tag{0.3}$$

0.1.3 Taylor's theorem for a function of two variables

Noting that (0.1) can be written in the alternative form

$$u(x+h) = \sum_{n=0}^{N} \frac{h^n}{n!} u^{(n)}(x) + O(h^{N+1}), \tag{0.4}$$

the corresponding form (with $N = 3$) for a function $u(x, y)$ of two independent variables x and y is

$$\begin{aligned} u(x+h, y+k) &= u(x,y) + hu_x(x,y) + ku_y(x,y) \\ &+ \frac{1}{2}\left(h^2 u_{xx}(x,y) + 2hku_{xy}(x,y) + k^2 u_{yy}(x,y)\right) \\ &+ \frac{1}{6}\left(h^3 u_{xxx}(x,y) + 3h^2 k u_{xxy}(x,y) + 3hk^2 u_{xyy}(x,y) + k^3 u_{yyy}(x,y)\right) \\ &+ O(h^4, h^3k, h^2k^2, hk^3, k^4), \end{aligned} \tag{0.5}$$

wherein we note the coefficient pattern is that of Pascal's triangle. In particular, if $h = k$ we have (dropping (x,y) on the RHS)

$$\begin{aligned} u(x+h, y+k) &= u + h\left(u_x + u_y\right) \\ &+ \frac{h^2}{2}\left(u_{xx} + 2u_{xy} + u_{yy}\right) \\ &+ \frac{h^3}{6}\left(u_{xxx} + 3u_{xxy} + 3u_{xyy} + u_{yyy}\right) \\ &+ \frac{h^4}{24}\left(u_{xxxx} + 4u_{xxxy} + 6u_{xxyy} + 4u_{xyyy} + u_{yyyy}\right) \\ &+ O(h^5). \end{aligned}$$

0.2 Greek alphabet and notation

Letter	Lower case	Upper case
alpha	α	
beta	β	
gamma	γ	Γ
delta	δ	Δ
epsilon	ϵ or ε	
zeta	ζ	
eta	η	
theta	θ or ϑ	Θ
iota	ι	
kappa	κ	
lambda	λ	Λ
mu	μ	
nu	ν	
xi	ξ	Ξ
pi	π or ϖ (rare)	Π
rho	ρ or ϱ (rare)	
sigma	σ or ς (rare)	Σ
tau	τ	
upsilon	υ	Υ
phi	ϕ or φ	Φ
chi	χ	
psi	ψ	Ψ
omega	ω	Ω

- The symbol for partial differentiation, ∂, is never to be confused with that for total differentiation, d, or the above δ.
- The symbol for the Laplacian operator, ∇, is called *nabla* and is not to be confused with the above Δ, which is actually an alternative symbol for ∇^2. The name comes from the word *nebel*, an ancient Assyrian harp with a triangular frame held with its point facing downwards.
- The *calligraphic* font

  $$\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{D},\mathcal{E},\mathcal{F},\mathcal{G},\mathcal{H},\mathcal{I},\mathcal{J},\mathcal{K},\mathcal{L},\mathcal{M},\mathcal{N},\mathcal{O},\mathcal{P},\mathcal{Q},\mathcal{R},\mathcal{S},\mathcal{T},\mathcal{U},\mathcal{V},\mathcal{W},\mathcal{X},\mathcal{Y},\mathcal{Z}$$

  will sometimes be used in the notes.
- Vectors may be either underlined $\underline{x}$ or bold $\boldsymbol{x}$.
- $$\sum_{i=0}^{n}{}' f_i = \frac{1}{2}f_0 + f_1 + \cdots + f_{n-1} + f_n \qquad \text{and} \qquad \sum_{i=0}^{n}{}'' f_i = \frac{1}{2}f_0 + f_1 + \cdots + f_{n-1} + \frac{1}{2}f_n$$

1 Approximation Theory

1.1 Polynomial interpolation

The concept of interpolation is the selection of a function $p(x)$ from a class of functions in such a way that the graph of $u = p(x)$ passes through a finite set of data points having the values $\{u_0, u_1, \ldots, u_n\}$ at the *nodes* $\{x_0, x_1, \ldots, x_n\}$, that is, $p(x_i) = u_i$. These data points may be the values of some mathematical functions or come empirically from observations or experiments. Several different ways of constructing such an interpolating function are considered, our goal being to differentiate a tabulated function approximately by taking derivatives of its *continuous* interpolating polynomial at *discrete* points.

1.1.1 Lagrange interpolation

Let $\{x_0, x_1, \ldots, x_n\}$ be $n+1$ distinct real numbers ($x_0 < x_1 < \cdots < x_n$) with associated function values $\{u_0, u_1, \ldots, u_n\}$. The idea of Lagrange interpolation is to multiply each u_i by a polynomial whose value is 1 at x_i and 0 at all the other nodes. Specifically, *Lagrange's formula* for the polynomial that interpolates this data is

$$p_n(x) = \sum_{i=0}^{n} \ell_i(x) u_i \,, \tag{1.1}$$

in which the polynomials $\ell_i(x)$ are given by

$$\ell_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \left(\frac{x - x_j}{x_i - x_j} \right) \qquad i = 0(1)n \,. \tag{1.2}$$

The notation $i = m(h)n$ means $i = m, m+h, m+2h, \ldots, n$. There are $n+1$ terms in (1.1), each a polynomial of degree n. When (1.1) is applied at a node x_k, (1.2) gives

$$\ell_i(x_k) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \left(\frac{x_k - x_j}{x_i - x_j} \right) \qquad i = 0(1)n \,, \tag{1.3}$$

which is a polynomial of degree n in x_k. The numerator of this polynomial is

$$(x_k - x_0)(x_k - x_1) \ldots (x_k - x_{i-1})(x_k - x_{i+1}) \ldots (x_k - x_n) \,,$$

one factor of which must vanish when $k \neq i$. Therefore $\ell_i(x_k) = 0$ when $i \neq k$ and the n roots of $\ell_i(x_k)$ satisfy $x_k \in \{x_0, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}$. Moreover $\ell_i(x_k)$ is clearly 1 when $i = k$. Hence $\ell_i(x_k) = \delta_{ik}$, the *Kronecker delta*,

$$\delta_{ij} = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases} \,,$$

whence (1.1) gives, upon using the "filtering property" of the Kronecker delta,

$$p_n(x_k) = \sum_{i=0}^{n} \ell_i(x_k) u_i = \sum_{i=0}^{n} \delta_{ik} u_i = u_k \,, \tag{1.4}$$

as required. [For later use, it is convenient to introduce the function

$$\Psi_n(x) \equiv \prod_{i=0}^{n} (x - x_i) \,, \tag{1.5}$$

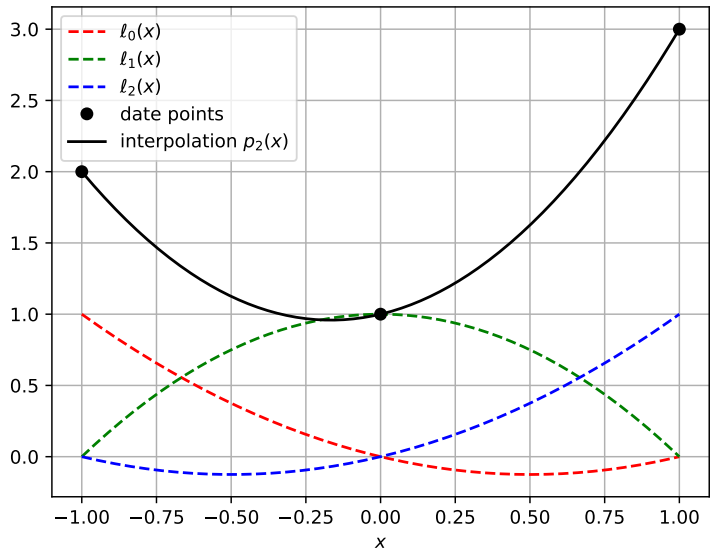
so that $\Psi_n(x)$ has roots at the $n+1$ *nodes* $\{x_0, \ldots, x_n\}$].

□ **Example 1.1** $n = 2$ *for quadratic interpolation of discrete data*
The polynomial of degree ≤ 2 that interpolates the data points $(0, 1)$, $(-1, 2)$ and $(1, 3)$ is, from (1.1) and (1.2),

$$p_2(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} u_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} u_1 + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} u_2 \,,$$

and so $p_2(x)$ for the given data is

$$p_2(x) = \frac{(x+1)(x-1)}{(0+1)(0-1)} \cdot 1 + \frac{(x-0)(x-1)}{(-1-0)(-1-1)} \cdot 2 + \frac{(x-0)(x+1)}{(1-0)(1+1)} \cdot 3 = \frac{1}{2}(2 + x + 3x^2) \,.$$



See `Python` script `3474_1.1.py` for implementation of this algorithm. □

As well as interpolating *discrete* data, (1.1) may be used to approximate a given *continuous function* $u(x)$ — e.g. for solving ODEs — at *and between* the data points $\{x_0, x_1, \ldots, x_n\}$. The sum

$$p_n(x; u) = \sum_{i=0}^{n} \ell_i(x) u(x_i) \tag{1.6}$$

has the required property since, by analogy with (1.4),

$$p_n(x_k; u) = \sum_{i=0}^{n} \ell_i(x_k) u(x_i) = \sum_{i=0}^{n} \delta_{ik} u(x_i) = u(x_k) \,. \tag{1.7}$$

The *interpolation error* is clearly

$$E_n(x;u) \equiv u(x) - p_n(x;u) = u(x) - \sum_{i=0}^{n} \ell_i(x) u(x_i) \tag{1.8}$$

which, by (1.7), vanishes at all the data nodes $x = x_k$, $k = 0(1)n$. However, the same is not true when $x \neq x_k$, i.e. between the nodes, when we have (without proof) the error formula

$$E_n(x;u) = \frac{u^{(n+1)}(\xi)}{(n+1)!} \Psi_n(x)\,, \tag{1.9}$$

for some $\xi \in [x_0, x_n]$ (assume $x_0 \leq x \leq x_n$ for simplicity) and $\Psi_n(x)$ is given by (1.5). Note that ξ *depends on* x.

Hence $u(x)$ should be $n+1$ times continuously differentiable with respect to x for the error to be bounded. Being able to quantify—and, indeed, optimise—how well a function can be approximated between given data is considered in some detail in §1.2.3.

□ **Example 1.2** *Bounding the error of linear interpolation p_1 of $u(x) = \log_{10} x$*
This example corresponds to the former practice of interpolating three- or four-figure log tables.

With $u(x) = \log_{10} x = \log_{10} e \cdot \ln x$ we have $u''(\xi) = -\log_{10} e/\xi^2 \approx -0.4343/\xi^2$. With $1 < x_0 < x_1$ (so that $u(x) > 0$ for $x \in [x_0, x_1]$) (1.9) gives the error approximation

$$E_1(x) = \frac{0.4343}{2\xi^2} \cdot (x - x_0)(x_1 - x)\,,$$

so that the upper and lower error bounds are

$$\frac{0.4343}{2x_1^2} \cdot (x - x_0)(x_1 - x) \leq E_1(x) \leq \frac{0.4343}{2x_0^2} \cdot (x - x_0)(x_1 - x)\,.$$

Moreover, since

$$\max_{x \in [x_0, x_1]} (x - x_0)(x_1 - x) = \frac{(x_1 - x_0)^2}{4}\,,$$

a *uniform bound* of the error is then

$$0 \leq \log_{10} x - p_1(x) \leq \frac{0.4343}{2x_0^2} \cdot \frac{(x_1 - x_0)^2}{4} < \frac{0.05429(x_1 - x_0)^2}{x_0^2} < 0.05429(x_1 - x_0)^2\,,$$

because $x_0 > 1$. □

□ **Example 1.3** *Approximating derivatives via Lagrange interpolation*
Substituting (1.9) into (1.8) and differentiating w.r.t. x gives

$$u'(x) = \sum_{i=0}^{n} \ell_i'(x) u(x_i) + \frac{u^{(n+1)}(\xi)}{(n+1)!} \Psi_n'(x)$$

and so, when $n = 1$,

$$\begin{aligned} u'(x) &= \left(\frac{x - x_1}{x_0 - x_1}\right)' u(x_0) + \left(\frac{x - x_0}{x_1 - x_0}\right)' u(x_1) + \frac{u''(\xi)}{2} \left[(x - x_0)(x - x_1)\right]' \\ &= \frac{u(x_1) - u(x_0)}{x_1 - x_0} + L(x) u''(\xi)\,, \end{aligned} \tag{1.10}$$

in which $L(x) \equiv x - \frac{1}{2}(x_0 + x_1)$ is linear, and so maximised at an endpoint, where $L(x_0) = \frac{1}{2}(x_0 - x_1)$ and $L(x_1) = \frac{1}{2}(x_1 - x_0)$. With $|x_1 - x_0| = h$, we obtain the error bound

$$\left| u'(x) - \frac{u(x_1) - u(x_0)}{h} \right| \leq \frac{h}{2} |u''(\xi)| \,, \tag{1.11}$$

where $\xi \in [x_0, x_0 + h]$. In order to approximate $u''(x)$ we would have to take $n \geq 2$. $\square$

The polynomial interpolation $p_n(x)$ in (1.1) for a given set of $n+1$ data points is *unique* (prove it using the fact that a polynomial of degree n with $n+1$ zeros must be identically zero). Importantly, this means the error estimate (1.9) is valid regardless of the method we use to obtain the polynomial interpolation. In the next section, we discuss a more efficient way to compute $p_n(x)$.

1.1.2 Newton divided differences

Whilst the Lagrange polynomials in §1.1.1 are useful for interpolating tabulated (particularly regularly spaced) nodes, their main disadvantage stems from the fact that if one wants to improve an approximation by increasing n (adding another node x_n) the entire calculation must be redone from scratch. We would prefer to have some sort of correction term $C(x)$ such that when we increase the number of nodes from n to $n+1$, we have

$$p_n(x) = p_{n-1}(x) + C(x) \,. \tag{1.12}$$

p_n, p_{n-1} and C are assumed to be polynomials of degrees n, $n-1$ and n respectively. By construction, at the nodes x_i we have

$$p_{n-1}(x_i) = u(x_i) \quad i = 0(1)n-1 \qquad \text{and} \qquad p_n(x_i) = u(x_i) \quad i = 0(1)n \,.$$

So for the first n nodes $x_0, \ldots, x_{n-1}$, (1.12) gives

$$C(x_i) = p_n(x_i) - p_{n-1}(x_i) = u(x_i) - u(x_i) = 0 \qquad i = 0(1)n-1 \,.$$

Since $C(x)$ is generally of degree n and it vanishes at nodes x_0 to x_{n-1}, it must have the form

$$C(x) = a_n \Psi_{n-1}(x) \,, \tag{1.13}$$

where $\Psi_n(x)$ is defined in (1.5). The constant a_n can be determined from the fact that at the new node x_n, we have $p_n(x_n) = u(x_n)$, (1.12) then gives

$$a_n = \frac{u(x_n) - p_{n-1}(x_n)}{\Psi_{n-1}(x_n)} \,, \tag{1.14}$$

which, together with (1.13), allows the correction term in (1.12) to be found. The coefficient a_n is called the *nth-order Newton divided difference of u* and is denoted by

$$a_n \equiv u[x_0, x_1, \ldots, x_n] \,. \tag{1.15}$$

From (1.12) and (1.13), you should convince yourself that $u[x_0, x_1, \ldots, x_n]$ is the coefficient of x^n in the polynomial p_n interpolating $u(x)$ at $x_0, \ldots, x_n$.

Useful formulae (without proof) for computing $u[x_0, x_1, \ldots, x_n]$ are

$$\begin{aligned} u[x_0, x_1] &= \frac{u(x_1) - u(x_0)}{x_1 - x_0}\,, \\ u[x_0, x_1, x_2] &= \frac{u[x_1, x_2] - u[x_0, x_1]}{x_2 - x_0}\,, \quad \ldots \\ u[x_0, x_1, \ldots, x_n] &= \frac{u[x_1, x_2, \ldots, x_n] - u[x_0, x_1, \ldots, x_{n-1}]}{x_n - x_0} \qquad n > 2\,, \end{aligned} \tag{1.16}$$

which explains the name. Formula (1.16) also reveals the possibility of large rounding errors if insufficient digits are used in the calculations (see Example 1.4). Combining (1.12), (1.13) and (1.15) gives

$$p_n(x) = p_{n-1}(x) + u[x_0, x_1, \ldots, x_n]\Psi_{n-1}(x)\,, \tag{1.17}$$

and so, noting that $p_0(x_0) \equiv u(x_0)$,

$$\left.\begin{aligned} p_1(x) &= u(x_0) + u[x_0, x_1]\Psi_0(x) \\ p_2(x) &= u(x_0) + u[x_0, x_1]\Psi_0(x) + u[x_0, x_1, x_2]\Psi_1(x) \\ &\vdots \\ p_n(x) &= u(x_0) + \textstyle\sum_{i=0}^{n} u[x_0, ..., x_i]\Psi_{i-1}(x)\,, \end{aligned}\right\} \tag{1.18}$$

which is *Newton's divided difference formula for the interpolating polynomial* and is much better for computation than the Lagrange formula: once all divided differences have been calculated we can progress from degree $n-1$ to degree n with the minimum of calculation.

An efficient construction of $p_n(x)$ can be obtained. Using (1.5) and the notation $D_0 \equiv u(x_0)$ and $D_i \equiv u[x_0, ..., x_i]$ for $i > 0$, the last equation in (1.18) becomes

$$p_n(x) = D_0 + D_1(x - x_0) + D_2(x - x_0)(x - x_1) + D_3(x - x_0)(x - x_1)(x - x_2) + \cdots$$

which admits the *nested* form

$$p_n(x) = D_0 + (x - x_0)\left[D_1 + (x - x_1)\left[D_2 + \cdots + (x - x_{n-2})\left[D_{n-1} + (x - x_{n-1})D_n\right]\cdots\right]\right]\,. \tag{1.19}$$

Given that the D_i are themselves evaluated iteratively using (1.16), (1.19) is an efficient recursive formula.

We now derive a formula for the error $E_n(x) = u(x) - p_n(x)$ at some $x \in [x_0, x_n]$ in terms of the divided difference. To do this, we examine the interpolation p_{n+1} through the nodes $x_0, \ldots, x_n, x$. (1.17) gives

$$p_{n+1}(x) - p_n(x) = u[x_0, x_1, \ldots, x_n, x]\Psi_n(x)\,.$$

Since x is a node of p_{n+1}, we have $p_{n+1}(x) = u(x)$. It immediately follows

$$E_n(x) = u[x_0, x_1, \ldots, x_n, x]\Psi_n(x) \tag{1.20}$$

when $x \in [x_0, x_n]$. The two expressions (1.9) and (1.20) for $E_n(x)$ must be equivalent because p_n is unique. Relabelling the nodes $\{x_0, \ldots, x_n, x\}$ so that $x_0 < x_1 < \cdots < x_m$ where $m = n + 1$, we relate the divided difference to the derivatives of $u(x)$,

$$u[x_0, x_1, \ldots, x_m] = \frac{u^{(m)}(\xi)}{m!}\,, \tag{1.21}$$

for some $\xi \in [x_0, x_m]$.

A note on workload reduction due to nested multiplication

The cubic polynomial

$$p_3(x) = a + bx + cx^2 + dx^3$$

is computed as

$$p_3(x) = a + b.x + c.x.x + d.x.x.x\,,$$

in which there are clearly $1 + 1 + 1 = 3$ additions and $1 + 2 + 3 = 6$ multiplications. In *nested* (or *Horner's*) form, the same polynomial is computed using

$$p_3(x) = a + x(b + x(c + xd)) = a + x.(b + x.(c + x.d))\,,$$

in which there are now 3 additions and 3 multiplications. It is easy to see that standard and nested computations of p_n require workloads of respectively $\frac{n}{2}(n+1) + n = \frac{n}{2}(n+3)$ and $n + n = 2n$, so that the nested form workload is $\dfrac{4}{n+3}$ times that of the standard.

Some properties of divided differences

(i) The result (1.16) is independent of the permutation of the integers $\{0, 1, \ldots, n\}$. E.g. when $n = 1$ we have

$$u[x_0, x_1] = \frac{u(x_0) - u(x_1)}{x_0 - x_1} = \frac{u(x_1) - u(x_0)}{x_1 - x_0} = u[x_1, x_0]\,.$$

When $n = 2$ we have

$$u[x_0, x_1, x_2] = \frac{u(x_0)}{(x_0 - x_1)(x_0 - x_2)} + \frac{u(x_1)}{(x_1 - x_0)(x_1 - x_2)} + \frac{u(x_2)}{(x_2 - x_0)(x_2 - x_1)}\,. \quad (1.22)$$

Swapping, e.g., 1 and 2 gives

$$u[x_0, x_2, x_1] = \frac{u(x_0)}{(x_0 - x_2)(x_0 - x_1)} + \frac{u(x_2)}{(x_2 - x_0)(x_2 - x_1)} + \frac{u(x_1)}{(x_1 - x_0)(x_1 - x_2)}\,,$$

which is identical to $u[x_0, x_1, x_2]$; the other four permutations yield the same result.

(ii) When the nodes are coincident, the divided differences can be defined as limiting cases of (1.16). When $n = 1$, we have

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

and, similarly, for $n > 1$ coincident nodes, we see from (1.21) as $\xi \to x_0$,

$$u[\underbrace{x_0, x_0, \ldots, x_0}_{n+1 \text{ times}}] = \frac{u^{(n)}(x_0)}{n!}\,.$$

(iii) When only some of the nodes are coincident we have, e.g.,

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

□ **Example 1.4** *Constructing divided differences*
The table below shows how the divided differences can be constructed systematically. The right-most entries of a particular colour are constructed from (1.16) using the four entries of the same colour in earlier columns. The D_i's in the polynomial interpolation (1.19) appear at the top of each column.

x_i	$u(x_i)$	$u[x_i, x_{i+1}]$	$u[x_i, x_{i+1}, x_{i+2}]$	$u[x_i, x_{i+1}, x_{i+2}, x_{i+3}]$
x_0	u_0			
		$u[x_0, x_1]$		
x_1	u_1		$u[x_0, x_1, x_2]$	
		$u[x_1, x_2]$		$u[x_0, x_1, x_2, x_3]$
x_2	u_2		$u[x_1, x_2, x_3]$	
		$u[x_2, x_3]$		$u[x_1, x_2, x_3, x_4]$
x_3	u_3		$u[x_2, x_3, x_4]$	
		$u[x_3, x_4]$		$u[x_2, x_3, x_4, x_5]$
x_4	u_4		$u[x_3, x_4, x_5]$	
		$u[x_4, x_5]$		$u[x_3, x_4, x_5, x_6]$
x_5	u_5		$u[x_4, x_5, x_6]$	
		$u[x_5, x_6]$		
x_6	u_6			
⋮	⋮	⋮	⋮	⋮

For the specific case $n = 4$ and $u(x) = \sqrt{x}$, with the x_i equally spaced on the interval $[2, \frac{12}{5}]$. Then, *without error*, algebraic manipulation gives

$$u[x_0, x_1, x_2, x_3, x_4] = \tfrac{250}{3}(2\sqrt{15} - 2\sqrt{230} + 6\sqrt{55} - 2\sqrt{210} + 5\sqrt{2})$$

which is $-0.0024\,8681\,0899\,39$ (correct up to the 14th decimal place) when evaluated using 18-digit arithmetic. The `Python` script `3474_1.4.py` gives a very concise algorithm for evaluating the above table (the indexing is tricky though). It illustrates the effect of *rounding error* in finite-precision arithmetic. The interval is now represented in the format $[2.0, 2.4]$. Using double-precision (16-digit) computation, this gives the value $-0.0024\,8681\,0899\,33$, whereas single-precision (7-digit) computation yields $-0.0024\,6861$. □

□ **Example 1.5** *Approximating derivatives using divided differences*
Setting $n = 1$ in (1.20) gives

$$u(x) - p_1(x) = u[x_0, x_1, x]\Psi_1(x)\,,$$

which, together with (1.5), (1.18) and (1.21), give

$$u(x) = u(x_0) + u[x_0, x_1](x - x_0) + \frac{u''(\xi)}{2!}(x - x_0)(x - x_1)\,,$$

for some $\xi \in [x_0, x_1]$. Differentiating w.r.t. x yields

$$u'(x) = u[x_0, x_1] + L(x)u''(\xi)\,, \tag{1.23}$$

where $L(x)$ is as defined in (1.10) in Example 1.3. Furthermore, since (1.10) and (1.23) are identical as they should, we obtain the same error bound as in (1.11), namely

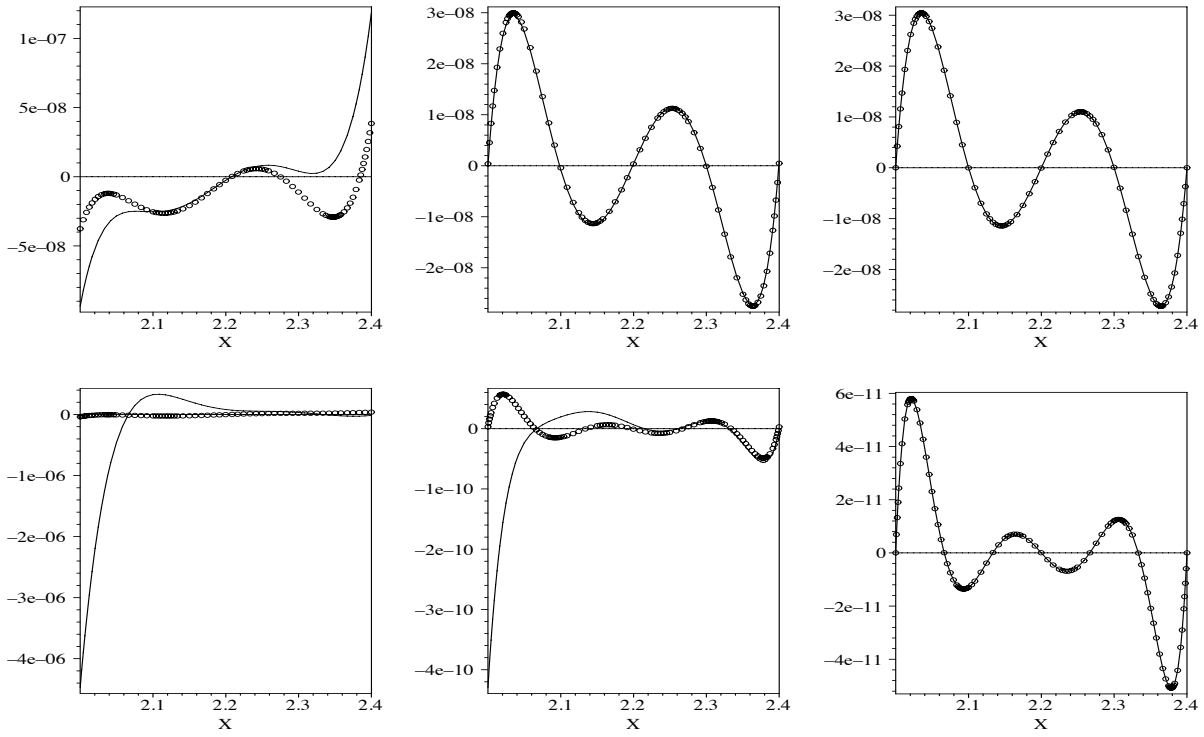
$$\left| u'(x) - \frac{u(x_1) - u(x_0)}{x_1 - x_0} \right| \leq \frac{x_1 - x_0}{2} u''(\xi)\,, \tag{1.24}$$

where now $\xi \in [x_0, x_1]$. As before, higher derivatives of u can be obtained using higher-order divided differences. □

1.1.3 Interpolation error

The *theoretical* error formulae (1.9) and (1.20) in principle apply to any polynomial interpolation p_n of a given set of nodes. However, in practice, there are rounding errors in finite-precision computation which are not taken into account by (1.9) or (1.20). Therefore it is not surprising the p_n obtained from two different algorithms, e.g. the Lagrange and the divided-difference interpolation, has different errors.

□ **Example 1.6** *Interpolating polynomial degree and floating-point error*
The following graphs show the error $u(x) - p_n(x)$ in the interpolation of the test function $u = \sqrt{x}$ on the interval $[2.0, 2.4]$ (see Example 1.4) when the nodes are *regularly spaced* throughout the interval. The errors for Lagrange (solid lines) and divided-difference (small circles) interpolation are shown.



In the first row of figures, $n = 4$; it is striking that the errors in the two interpolations coincide at this low value of n. From left to right, calculations were performed with 8-, 10- and 16-digit accuracy, the first and last of which correspond to single- and double-precision arithmetic in

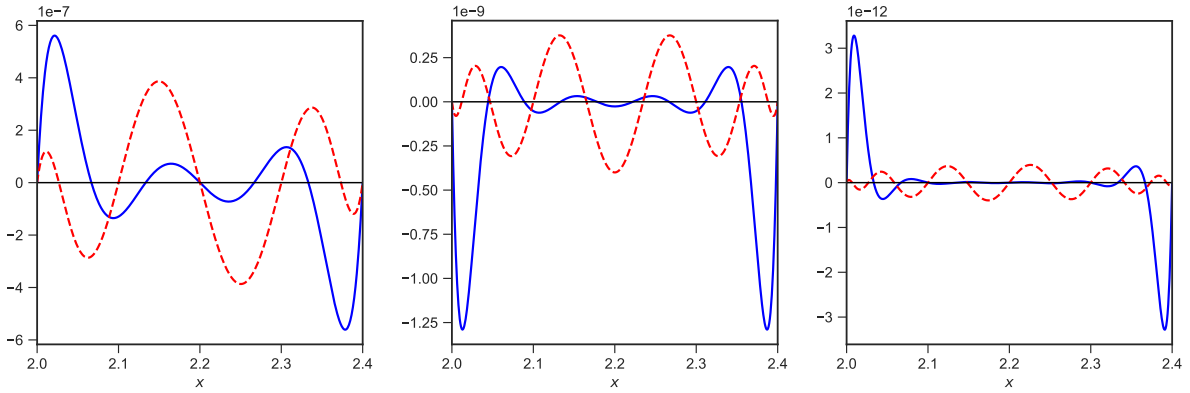
many programming languages. Note from the vertical scales that the 8-digit interpolations are far less accurate than the 16-digit ones.

In the second row of figures, $n = 6$ and, from left to right, calculations were performed with 8-, 12- and 16-digit accuracy. Thus the price to be paid for increasing n—and considerably reducing the error, as seen from the vertical scales—is that this is only achievable by using more expensive higher-accuracy computations.

Note that the first two figures in the second row show convincingly that the divided-difference approximation has a smaller error than Lagrange interpolation when lower-accuracy arithmetic is used. Note also that the largest errors are towards the interval endpoints, irrespective of n. This is explained in the next section. □

□ **Example 1.7** *The Runge phenomenon*
In Example 1.6 the largest errors $u(x) - p_n(x)$ for both Lagrange interpolation and divided-difference interpolation occurred towards the ends of the interval irrespective of n. The errors are proportional to $\Psi_n(x)$ as defined in (1.5),

$$\Psi_n(x) \equiv \prod_{i=0}^{n} (x - x_i)\,.$$



The solid lines show, from left to right, $\Psi_6(x)$, $\Psi_9(x)$ and $\Psi_{12}(x)$ when the x_i are regularly spaced in $[a, b]$. As n increases, $\Psi_n(x)$ clearly becomes increasingly peaked toward the interval ends, thereby explaining the error distribution in Example 1.6.

By contrast, the dashed lines show the same three functions when the x_i have been compressed towards the interval endpoints. Specifically, this compression is a trigonometric transformation corresponding to placing $n + 1$ nodes at *equal intervals* of π/n around a semicircle with base $[a, b]$, and then projecting parallel with the Ψ-axis down onto the x-axis. Such nodes are located at the so-called *Chebyshev points*, considered more fully in §§1.2.5–1.2.8. When they are used, the error is distributed far more uniformly over $[a, b]$, even as n increases.

These graphs were produced using `Python` script `3474_1.7.py`. □

Example 1.7 demonstrates that the desirable property

$$\max_{x\in[x_0,x_n]} |u(x) - p_n(x)| \to 0 \qquad n \to \infty \tag{1.25}$$

is not guaranteed for certain node distributions; this is a manifestation of the so-called *Runge phenomenon*. However, provided that $u(x)$ is infinitely continuously differentiable, convergence *is* guaranteed when the nodes are located at the Chebyshev points; it is also guaranteed when the nodes are regularly spaced *provided* $u(x)$ *is a periodic function of* x *on* $[a, b]$.

When the nodes are regularly spaced, the figures in Example 1.7 reveal that the interpolation error is far greater near the interval endpoints (the "end-error") than near the interval midpoint (the "mid-error"). To quantify the end-error, use $n+1$ nodes beginning with $x_0 = 0$ and constant spacing h, so that the ith node is $x_i = ih$. Then (1.5) gives

$$\Psi_n(x) \equiv \prod_{i=0}^{n}(x - ih)\,. \tag{1.26}$$

For general n, it is not possible to obtain an explicit formula for the location of the maximum in $[x_0, x_1]$. However, at the midpoint of this first interval, where $x = h/2$, straightforward evaluation of (1.26) gives a crude approximation of the end-error as

$$|\Psi_n(\tfrac{h}{2})| = \frac{(2n)!}{2^{2n+1}n!}h^{n+1}\,. \tag{1.27}$$

To quantify the mid-error, consider only odd n, where $|\Psi_n(x)|$ has a local maximum *at* the interval midpoint $x = nh/2$, see for example the graph of $\Psi_9(x)$ in Example 1.7. From (1.26),

$$\Psi_n(\tfrac{nh}{2}) = \prod_{i=0}^{n}(\tfrac{n}{2} - i)h\,.$$

With (odd) $n = 2m + 1$, $m \in \mathbb{N}$, straightforward evaluation gives the mid-error as

$$|\Psi_{2m+1}(\tfrac{2m+1}{2}h)| = \left[\frac{(2m+1)!}{2^{2m+1}m!}\right]^2 h^{2m+2}\,. \tag{1.28}$$

Thus the ratio of the end-error to the mid-error is

$$\mathcal{R}_m \equiv \frac{|\Psi_{2m+1}(\frac{h}{2})|}{|\Psi_{2m+1}(\frac{2m+1}{2}h)|} = \frac{(4m+2)!(m!)^2}{2[(2m+1)!]^3}\,.$$

By using Stirling's formula $n! \approx (2\pi n)^{1/2}n^n e^{-n}$ and the definition of the exponential function $e = \lim_{n\to\infty}(1 + \frac{1}{n})^n$, we obtain the asymptotic result

$$\mathcal{R}_m \to \frac{2^{2m-1/2}}{m+1} \qquad m \to \infty\,, \tag{1.29}$$

i.e. the error ratio is exponentially large. We conclude that the interpolation nodes should be chosen so that the interpolation point x is near to the middle of $[x_0, x_n]$.

1.2 Approximation of functions

In this section, we study using a polynomial $q(x)$ to approximate a given function $u(x)$. Different from polynomial interpolation, here we do not require the polynomial to pass through a given set of data points, we only require $q(x)$ to be "close" to $u(x)$. But what does it mean by "close"? We need some ways to quantify how good $q(x)$ is at approximating $u(x)$.

Let $\mathcal{A}$ be the set of possible approximations to $u(x)$, and let $\mathcal{A} \subseteq \mathcal{B}$, a normed linear space with norm $\|u\|$ that satisfies the usual triangle inequality and homogeneity condition

$$\|u+v\| \le \|u\| + \|v\| \qquad \text{and} \qquad \|\lambda u\| = |\lambda|\,\|u\|$$

for all $u, v \in \mathcal{B}$ and scalars λ. A theorem asserts that, for every $u \in \mathcal{B}$, there exists a $q^* \in \mathcal{A}$ such that q^* is a *best approximation* to u. That is, for all other $q \in \mathcal{A}$,

$$\|u - q^*\| \le \|u - q\| \ . \tag{1.30}$$

Methods for constructing q^*, or rather approximations to it, comprise the rest of this chapter.

1.2.1 The $\mathcal{L}_p$ norms

In most practical applications, $\mathcal{B}$ is $\mathcal{C}[a,b]$, the set of continuous functions on $[a,b]$. In discrete problems—in which $u = (u_1, \ldots, u_m)$— $\mathcal{B}$ is $\mathbb{R}^m$, the set of real m-vectors. For finite p, the *$\mathcal{L}_p$-norm* in $\mathcal{C}[a,b]$ and $\mathbb{R}^m$ is defined to have the value

$$\|u\|_p \equiv \left\{ \int_a^b |u(x)|^p \, dx \right\}^{1/p} \quad \text{and} \quad \|u\|_p \equiv \left\{ \sum_{i=1}^m |u_i|^p \right\}^{1/p} \qquad 1 \le p < \infty \tag{1.31}$$

respectively. The ∞-norms are similarly defined by

$$\|u\|_\infty \equiv \max_{x\in[a,b]} |u(x)| \quad \text{and} \quad \|u\|_\infty \equiv \max_{1\le i\le m} |u_i| \, . \tag{1.32}$$

The most commonly used norms have $p = 1,\ 2$ and ∞. $\|u-q\|_p$ can be considered as the "distance" between u and q.

The *weighted inner product* is defined for $u, v \in \mathcal{C}[a,b]$ or $u, v \in \mathbb{R}^m$ by

$$(u,v) \equiv \int_a^b w(x)\, u(x)\, v(x)\, dx \qquad \text{or} \qquad (u,v) \equiv \sum_{i=1}^m w_i\, u_i\, v_i \, , \tag{1.33}$$

in which $w(x) > 0$ for $x \in [a,b]$ or $w_i > 0$ for $i = 0(1)n$ respectively. Hence the standard inner product simply has weight function $w(x) \equiv 1$ for $x \in [a,b]$ or $w_i \equiv 1$ for $i = 0(1)n$, and in this case (1.31), (1.32) and the *Cauchy-Schwarz inequality*

$$|(u,v)| \le \|u\|_2 \, \|v\|_2$$

gives

$$\|u\|_1 \le (b-a)^{1/2} \, \|u\|_2 \le (b-a)\, \|u\|_\infty \qquad \forall u \in \mathcal{C}[a,b] \, . \tag{1.34}$$

Frequently used in the context of approximation is a *weighted* 2-norm

$$\|u\|_{w,2} \equiv \left\{ \int_a^b w(x)\,|u(x)|^2\,dx \right\}^{1/2} . \tag{1.35}$$

□ **Example 1.8** *Practical significance of the ∞-norm*
In approximating $u(x) = 1$ by $q_\lambda(x) \equiv x^\lambda$ on $[0,1]$, where $\lambda > 0$ is a parameter, we have for all $x > 0$, $q_\lambda(x) \to 1$ and so $E_\lambda(x) \equiv u(x) - q_\lambda(x) \to 0$ as $\lambda \to 0$. However, (1.31) and (1.32) give

$$\|E_\lambda\|_1 = \frac{\lambda}{\lambda+1}\,, \quad \|E_\lambda\|_2 = \left\{ \frac{2\lambda^2}{(\lambda+1)(2\lambda+1)} \right\}^{1/2} \quad \text{and} \quad \|E_\lambda\|_\infty = 1\,,$$

which respectively converge to 0, 0 and 1 as $\lambda \to 0$. Hence convergence in the 1- or 2-norm does not guarantee *uniform* convergence, so if we develop approximations that converge in the ∞-norm, (1.34) (with u replaced by E_λ) guarantees convergence in both the 1- and 2-norms. In this sense, the ∞-norm is said to be the "least forgiving" norm. □

1.2.2 Weierstrass' theorem

Let $u(x)$ be continuous for $x \in [a,b]$ and let $\epsilon > 0$. Then there is a polynomial $p(x)$ for which

$$\|u - p\|_\infty < \epsilon\,.$$

Effectively, there exists a polynomial p that is arbitrarily close to u, hence providing justification to approximating continuous functions by polynomials. The theorem is proved by construction on $[0,1]$, onto which any interval can be mapped by a linear transformation. For $n > 0$ define

$$p_n(x) = \sum_{k=0}^{n} \frac{n!}{k!(n-k)!} u\left(\tfrac{k}{n}\right) x^k (1-x)^{n-k} \qquad x \in [0,1] \tag{1.36}$$

and let $u(x)$ be bounded on $[0,1]$. Then such *Bernstein polynomials* satisfy

$$\lim_{n\to\infty} p_n(x) = u(x)\,,$$

and they also mimic well the qualitative behaviour of $u(x)$ in that

$$\lim_{n\to\infty} \left\| u^{(r)} - p_n^{(r)} \right\|_\infty = 0 \qquad u \in \mathcal{C}^r[0,1]\,.$$

However, the price to pay is that the convergence in the above limits can be *very* slow, even for simple functions. For example, when $u(x) = x^2$ and $r = 0$, it can be shown that

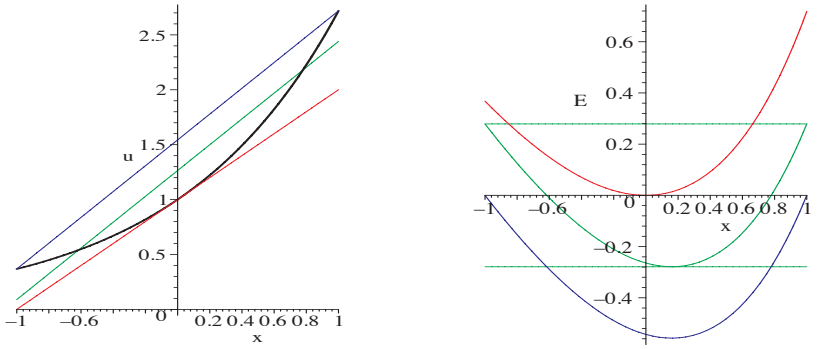
$$u(x) - p_n(x) = \frac{x(x-1)}{n} \qquad n > 0\,,$$

from which

$$\|u - p_n\|_\infty = (4n)^{-1}\,, \tag{1.37}$$

i.e. the norm converges only as $1/n$. Our goal is to find alternative approximating functions $p_n(x)$ for which this convergence is considerably more rapid.

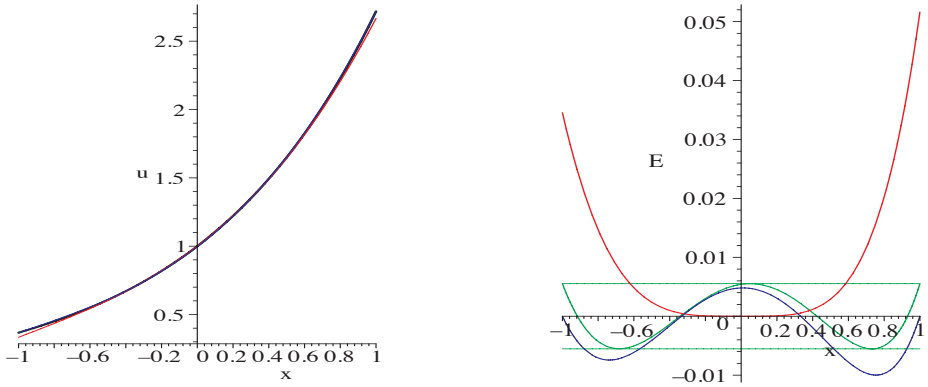
□ **Example 1.9** *"Close" approximations to* $u(x) = \mathrm{e}^x$ *on [-1,1]*



The left-hand figure shows the curve $u = \mathrm{e}^x$ and the straight lines are, from top to bottom, the *linear*

- interpolation polynomial through $(-1, \mathrm{e}^{-1})$ and $(1, \mathrm{e})$: $p_1(x) \approx 1.5431 + 1.1752\,x$;
- optimal polynomial (discussed in §1.2.3): $q_1^*(x) \approx 1.2643 + 1.1752\,x$, and;
- Taylor expansion about $x = 0$: $t_1(x) = 1 + x$.

The right-hand figure shows, from top to bottom, the errors curves $E_t = u(x) - t_1(x)$, $E_q = u(x) - q_1^*(x)$ and $E_p = u(x) - p_1(x)$, showing $\|E_t\|_\infty > \|E_p\|_\infty > \|E_q\|_\infty$. Because p_1 and q_1^* are parallel, E_p and E_q are translations of each other. However, the optimality of q_1^* is quantified by the confinement of E_q between the two parallel lines, *symmetrically disposed about* $E = 0$, at $E \approx \pm 0.2788$. Effectively, q_1^* is a linear Lagrange interpolation but with the nodes moved from $\{-1, 1\}$ to approximately $\{-0.6164, 0.7795\}$ which, notably, are not symmetric about $x = 0$.



When n is increased to 3, *cubic* polynomials p_3, q_3^* and t_3 can be similarly constructed. The right-hand curves reveal that now $\|E_t\|_\infty \gg \|E_p\|_\infty \approx \|E_q\|_\infty$, because the Taylor expansion is ineffective away from $x = 0$. E_p and E_q are now distributed much more evenly across $[-1, 1]$, E_q within the parallel lines $E \approx \pm 0.00553$. The error curves E_p and E_q are reminiscent of the Chebyshev-node plots of $\Psi(x)$ in Example 1.7; these nodes arise in the context of near-optimal interpolation.

Systematic determination of q_n^* is addressed in §1.2.3. □

1.2.3 Minimax approximation

We seek the polynomial approximation $q_n(x)$ (of degree $\leq n$) to $u(x)$ that **min**imises the **max**imum error. Define the *minimax error* by

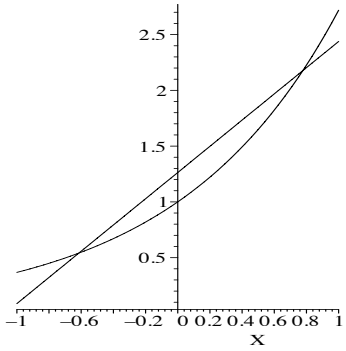
$$\rho_n(u) \equiv \min_{\deg(q_n)\leq n} \|u - q_n\|_\infty \,. \tag{1.38}$$

The minimisation is over all polynomials with degree $\leq n$. In other words, there does not exist a polynomial q_n of degree $\leq n$ that can approximate u with a smaller maximum error than $\rho_n(u)$. The *minimax approximation*, $q_n^*(x)$, to $u(x)$ on $[a,b]$ should therefore satisfy

$$\|u - q_n^*\|_\infty = \rho_n(u)\,. \tag{1.39}$$

Whether or not q_n^* is unique is considered in §1.2.4.

□ **Example 1.10** *Constructing q_1^* in Example 1.9*
Recall $u(x) = \mathrm{e}^x$ and postulate $q_1^*(x) = a_0 + a_1 x$ so that the error is $E_1(x) = \mathrm{e}^x - (a_0 + a_1 x)$.



Geometric arguments show that $|E_1(x)|$ must vanish at exactly 2 points and attain its maximum value, ρ_1, at exactly 3 points, say ξ_0, ξ_1 and ξ_2. From the figure, $\xi_0 = -1$, $\xi_1 \in [0,1]$ and $\xi_2 = 1$. Note also that $E_1(x)$ has a local minimum at $x = \xi_1$. Thus we have 4 equations

$$\left.\begin{array}{ll} E_1(\xi_0) = +\rho_1 & \\ E_1(\xi_1) = -\rho_1 & E_1'(\xi_1) = 0 \\ E_1(\xi_2) = +\rho_1 & \end{array}\right\} \tag{1.40}$$

in the 4 unknowns ξ_1, a_0, a_1 and ρ_1. In explicit form these are

$$\left.\begin{array}{rl} e^{-1} - a_0 + a_1 = \rho_1 & \\ e^{\xi_1} - a_0 - a_1\,\xi_1 = -\rho_1 & e^{\xi_1} - a_1 = 0 \\ e - a_0 - a_1 = \rho_1 & \end{array}\right\}, \tag{1.41}$$

yielding $a_1 = \sinh 1 \approx 1.1752$, $\xi_1 = \ln a_1 \approx 0.1614$, $\rho_1 \approx 0.2788$, $a_0 \approx 1.2643$, and so

$$E_1(x) = \mathrm{e}^x - 1.2643 - 1.1752\,x\,.$$

Then, by (1.39), $\|E_1\|_\infty = \|u - q_1\|_\infty = \rho_1 \approx 0.2788$. □

Note that the above approach works only if $u''(x)$ is of one sign throughout $[a,b]$.

As the following example shows, matters become considerably more complicated when constructing q_n^* for $n \geq 2$, when the interior values of ξ_i may not be explicitly calculated as they could be in Example 1.10.

□ **Example 1.11** *Constructing q_2^* for $u(x) = \mathrm{e}^x$ on $[-1, 1]$*
We now postulate $q_2^*(x) = b_0 + b_1 x + b_2 x^2$ so that the error is $E_2(x) = u(x) - q_2^*(x)$. By the arguments preceding (1.40), we seek b_0, b_1, b_2, ξ_1, ξ_2 and ρ_2 from the six nonlinear equations

$$\left.\begin{array}{ll} E_2(\xi_0) = +\rho_2 & \\ E_2(\xi_1) = -\rho_2 & E_2'(\xi_1) = 0 \\ E_2(\xi_2) = +\rho_2 & E_2'(\xi_2) = 0 \\ E_2(\xi_3) = -\rho_2 & \end{array}\right\}, \tag{1.42}$$

in which $\xi_0 = -1$ and $\xi_3 = 1$ and the oscillatory $E_2(\xi_i) = (-1)^i \rho_2$ should be noted (see §1.2.4). By excessive manipulation we arrive at

$$\begin{aligned} b_0 &= \cosh 1 - \frac{e^{\xi_1} - e^{\xi_2}}{2(\xi_1 - \xi_2)} \\ b_1 &= -\frac{4\sinh 1 + (\xi_1 + \xi_2 - 2)(e^{\xi_1} - e^{\xi_2})}{2(\xi_1 - \xi_2 - 2)} \\ b_2 &= \frac{e^{\xi_1} - e^{\xi_2}}{2(\xi_1 - \xi_2)} \\ \rho_2 &= \frac{2(\xi_1 - \xi_2)\sinh 1 + (\xi_1 + \xi_2 - 2)(e^{\xi_1} - e^{\xi_2})}{\xi_1 - \xi_2 - 2} \end{aligned}$$

in which ξ_1 and ξ_2 are determined from the coupled equations

$$f_1(\xi_1, \xi_2) \equiv e^{\xi_2} + \frac{2\sinh 1}{\xi_1 - \xi_2 - 2} + \frac{[(\xi_1 - \xi_2)^2 - 2(\xi_1 - 3\xi_2)](e^{\xi_1} - e^{\xi_2})}{(\xi_1 - \xi_2)(\xi_1 - \xi_2 - 2)} = 0 \tag{1.43}$$

$$f_2(\xi_1, \xi_2) \equiv e^{\xi_1} - \cosh 1 + \frac{(\xi_1 + \xi_2)\sinh 1}{\xi_1 - \xi_2 - 2} + \frac{(\xi_1 - 1)(2 - \xi_1 + 3\xi_2 + \xi_1\xi_2 - \xi_2^2)(e^{\xi_1} - e^{\xi_2})}{2(\xi_1 - \xi_2)(\xi_1 - \xi_2 - 2)} = 0\,, \tag{1.44}$$

which can be solved using 2-D Newton-Raphson iteration with initial guess $\xi_1 = -\xi_2 = 0.5$.

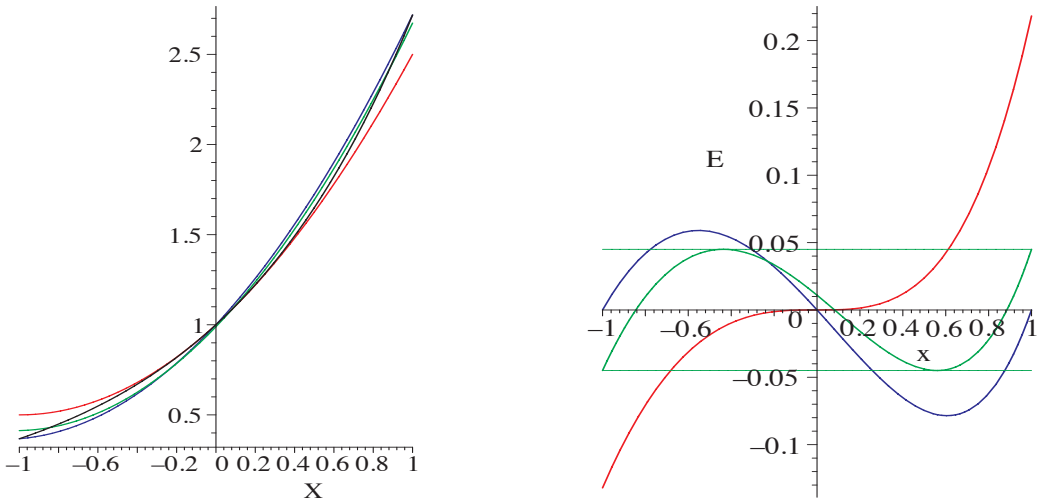
Digression: n-dimensional Newton-Raphson. Recall that the 1-D Newton-Raphson iteration for finding a root of $f(x) = 0$ is $x^{(k+1)} = x^{(k)} - f(x^{(k)})/f'(x^{(k)})$. Write this as $f'(x^{(k)})[x^{(k+1)} - x^{(k)}] = -f'(x^{(k)})$ or $f'(x^{(k)})\delta x^{(k)} = -f(x^{(k)})$ which, with $x^{(k)}$ known, yields the increment $\delta x^{(k)}$, so giving the new iteration $x^{(k+1)}$.

The n-dimensional analogy of this is the solution of $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ via the iteration $\mathcal{J}(\boldsymbol{f}(\boldsymbol{x}^{(k)}))\boldsymbol{\delta x}^{(k)} = -\boldsymbol{f}(\boldsymbol{x}^{(k)})$, in which $\mathcal{J}$ is the *Jacobian* matrix of $\boldsymbol{f} = (f_1, \ldots, f_n)$, each of whose elements are functions of $\boldsymbol{x} = (x_1, \ldots, x_n)$, evaluated at $\boldsymbol{x} = \boldsymbol{x}^{(k)}$. Hence we solve a *linear* problem at each step k. To solve (1.43) and (1.44) we guess $\xi_1^{(0)} = -0.5$ and $\xi_2^{(0)} = 0.5$ and iterate using

$$\left(\begin{array}{cc} \dfrac{\partial f_1}{\partial \xi_1} & \dfrac{\partial f_1}{\partial \xi_2} \\[2ex] \dfrac{\partial f_2}{\partial \xi_1} & \dfrac{\partial f_2}{\partial \xi_2} \end{array}\right)_{\substack{\xi_1 = \xi_1^{(k)} \\ \xi_2 = \xi_2^{(k)}}} \left(\begin{array}{c} \delta\xi_1^{(k)} \\[2ex] \delta\xi_2^{(k)} \end{array}\right) = -\left(\begin{array}{c} f_1(\xi_1^{(k)}, \xi_2^{(k)}) \\[2ex] f_2(\xi_1^{(k)}, \xi_2^{(k)}) \end{array}\right) \quad \text{or} \quad \mathcal{J}(\boldsymbol{\xi}^{(k)})\,\boldsymbol{\delta\xi}^{(k)} = -\boldsymbol{f}(\boldsymbol{\xi}^{(k)})\,.$$

The iteration terminates when $\left\|\boldsymbol{\delta\xi}^{(k)}\right\|_2$ is less than a pre-specified small tolerance.

We find that $\xi_1 \approx -0.4370$ and $\xi_2 \approx 0.5601$, giving $b_0 \approx 0.9890$, $b_1 \approx 1.1302$, $b_2 \approx 0.5540$ and $\rho_2 \approx 0.04502$. With this data, we construct the quadratic minimax approximation to e^x, which is compared with the quadratic interpolation through the nodes $\{x_0, x_1, x_2\} = \{-1, 0, 1\}$ and the Taylor expansion about $x = 0$ in the figures below.



Note that the right-hand error plot clearly shows that q_2^* is bounded by $E_q = \pm\rho_2$. It also shows, as expected, the Taylor series becomes less accurate as we move away from the centre of the expansion, in this case $x = 0$. The minimax approximation q_2^* vanishes at $\{\eta_0, \eta_1, \eta_2\} \approx \{-0.8447, 0.08211, 0.8857\}$. Hence it can be viewed as an interpolation polynomial with nodes $\{\eta_0, \eta_1, \eta_2\}$ and by (1.9), the error of the approximation is given by

$$E_2(x) = \frac{e^\xi}{3!}(x - \eta_0)(x - \eta_1)(x - \eta_2)$$

for some $\xi \in [-1, 1]$. $\square$

It is clear that when $E_n(x)$ has more than one local maximum or minimum in (a, b), construction of q_n^* for even the lowest value of $n = 1$ requires solution of equations such as (1.42), or extended versions thereof. This illustrates that *direct* solutions of systems such as (1.42) in the case of a general $u(x)$ can be unyieldingly complicated. In Example 1.11, it was possible to express 4 of the 6 unknowns in terms of the other 2, which could then be obtained numerically. In other cases we might have to solve all 6 equations numerically. It is also clear that a more systematic approach than the *ad hoc* methods of Examples 1.9, 1.10 and 1.11 is required.

1.2.4 Error-oscillation theorems

Two theorems arise in the consideration of an error which is uniformly distributed and which oscillates. The first one is useful for estimating $\rho_n(u)$ in (1.38) without having to find q_n^*, and the second one forms the basis of an algorithm for determining $q_n^*(x)$ and $\rho_n(u)$.

Theorem (de la Vallée-Poussin) (VP) Let $u \in \mathcal{C}[a, b]$ and $n \geq 0$, and let $q_n(x)$ be a polynomial of degree $\leq n$ that satisfies

$$u(\xi_i) - q_n(\xi_i) = (-1)^i e_i \qquad i = 0(1)n + 1\,, \tag{1.45}$$

with the e_i nonzero and of the same sign and with $a \le \xi_0 < \xi_1 < \cdots < \xi_{n+1} \le b$. In other words, the curve of q_n alternates between being above and being below the curve of u and the error $u - q_n$ oscillates between positive and negative at the $n+2$ points ξ_i. Then

$$\underbrace{e_{\min} \equiv \min_{0\le i\le n+1} |e_i|}_{\text{best}} \le \underbrace{\rho_n(u) \equiv \|u - q_n^*\|_\infty}_{\text{optimum}} \le \underbrace{\|u - q_n\|_\infty}_{\text{worst}} . \tag{1.46}$$

That is, the minimax error can be estimated using $\rho_n(u) \in [e_{\min}, \|u - q_n\|_\infty]$. Or if we have obtained a q_n satisfying (1.45), then a narrow interval of $[e_{\min}, \|u - q_n\|_\infty]$ indicates that q_n is close to the best approximation q_n^*. $\diamond$

Chebyshev Alternation (Equioscillation) Theorem (CAT) q_n^* is the unique polynomial (of degree $\le n$) of best approximation to $u \in \mathcal{C}[a,b]$ if and only if there exist $n+2$ points $a \le \xi_0 < \xi_1 < \cdots < \xi_{n+1} \le b$ such that

$$u(\xi_i) - q_n^*(\xi_i) = (-1)^i \epsilon_n \qquad i = 0(1)n+1\,, \tag{1.47}$$

where $\epsilon_n = \pm\rho_n(u) = \pm\|u - q_n^*\|_\infty$, i.e. if and only if the difference $E_n(x) \equiv u(x) - q_n^*(x)$ takes consecutively its maximal value with alternating signs at least $n+2$ times. $\diamond$

Note carefully that, in (1.45), the distinct errors e_i are dependent upon the nodal index i (and, implicitly, the polynomial degree n) whereas in (1.47), the constant nodal error ϵ_n is dependent upon only n.

$\square$ **Example 1.12** *CAT for* $u(x) = \mathrm{e}^x$ *on* $[-1,1]$
(i) With $n = 1$ (Example 1.10), $u(x) - q_1^*(x)$ has $n+2 = 3$ oscillating extrema of approximately $+0.2788$, -0.2788 and $+0.2788$ at the nodes $\xi_0 = -1$, $\xi_1 \approx 0.1614$ and $\xi_2 = 1$ respectively.

(ii) With $n = 2$ (Example 1.11), $u(x) - q_2^*(x)$ has $n+2 = 4$ oscillating extrema of approximately $+0.045$, -0.045, $+0.045$ and -0.045 at the nodes $\xi_0 = -1$, $\xi_1 \approx -0.4370$, $\xi_2 \approx 0.5601$ and $\xi_3 = 1$ respectively. $\square$

An important application of the CAT arises in determining the best approximation to the function $u(x) = x^{n+1}$ on $[-1,1]$ by a polynomial of degree $\le n$.

The Remez Algorithm

The set of nodes $\mathcal{N}_n = \{\xi_0, \xi_1, \ldots \xi_{n+1}\}$ in the CAT are referred to as an *alternating set* of *length* $n+2$; for $n > 1$, we have already seen how complicated (e.g. Example 1.11) they can be to obtain *directly*. When this set is known, and when the minimax polynomial is postulated to be of the form (see, e.g., Examples 1.10 and 1.11)

$$q_n^*(x) = \sum_{j=0}^{n} a_j x^j\,,$$

setting $x = \xi_i$ in (1.47) of the CAT yields

$$u(\xi_i) - \sum_{j=0}^{n} a_j \xi_i^j = (-1)^i \epsilon_n \qquad i = 0(1)n+1\,,$$

which are $n+2$ *linear* equations, readily solved, for the set of unknowns

$$\mathcal{U}_n = \{\underbrace{a_0, a_1, \ldots, a_n}_{\text{coeffs. in } q_n^*}, \quad \underbrace{\epsilon_n}_{\text{minimax error}} \quad \}\,.$$

In other words, when the alternating set $\mathcal{N}_n$ is known, the problem of determining the minimax polynomial is effectively solved. However, determining the *a priori* unknown set $\mathcal{N}_n$ is a non-trivial component of the solution process when $n > 1$.

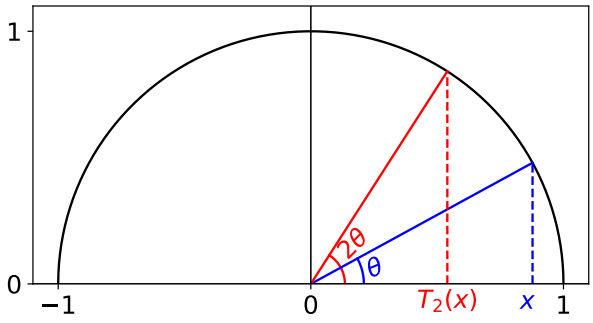
The *Remez/Remes* or *exchange* algorithm does this *iteratively* (and not directly as in Example 1.10). An initial distribution $\mathcal{N}_n^{(0)}$ of nodes is specified from which an initial estimate $\mathcal{U}_n^{(0)}$ of unknowns is computed using the second equation above; these are used in the first equation to compute an initial approximation $q_n^{(0)}(x)$ of the minimax polynomial. The elements of $\mathcal{N}_n^{(0)}$ are then exchanged systematically for those that lie at the turning points of $E_n^{(0)}(x) = u(x) - q_n^{(0)}(x)$, thereby giving a new set $\mathcal{N}_n^{(1)}$, from which $\mathcal{U}_n^{(1)}$ can be determined to yield $q_n^{(1)}(x)$, and hence $E_n^{(1)}(x)$.

This process is repeated iteratively and the algorithm terminates at step k when the nodes in $\mathcal{N}_n^{(k)}$ and coefficients in $\mathcal{U}_n^{(k)}$ cause the second equation above to be satisfied to within a prescribed tolerance; the minimax polynomial is then approximated by $q_n^*(x) \approx q_n^{(k)}(x)$.

However, since there are far simpler *direct* methods for obtaining *near-minimax* approximations, no further details of the Remes algorithm (for which there is an extensive theory) will be presented.

1.2.5 Chebyshev polynomials

Consider the semicircle of unit radius below.



For any x in the interval $[-1, 1]$, there is an angle $\theta \in [0, \pi]$ such that

$$x = \cos\theta.$$

The *Chebyshev polynomial of degree* n is given on $[-1, 1]$ by the rule

$$T_n(x) = \cos(n\theta) = \cos[n(\cos^{-1} x)]\,. \tag{1.48}$$

From the definition (1.48), we have

$$T_0(x) = 1\,, \qquad T_1(x) = x. \tag{1.49}$$

For $n > 1$, $T_n(x)$ is given by the 2-term recurrence relation

$$T_{n+1}(x) = 2\,x\,T_n(x) - T_{n-1}(x) \tag{1.50}$$

which is initiated using (1.49); that is, $T_{n+1}(x)$ is a polynomial in x for all n. The relation (1.50) can be proved using the trigonometric formula $\cos(a+b) + \cos(a-b) = 2\cos a \cos b$ with $a = n\theta$ and $b = \theta$. It can also be shown that

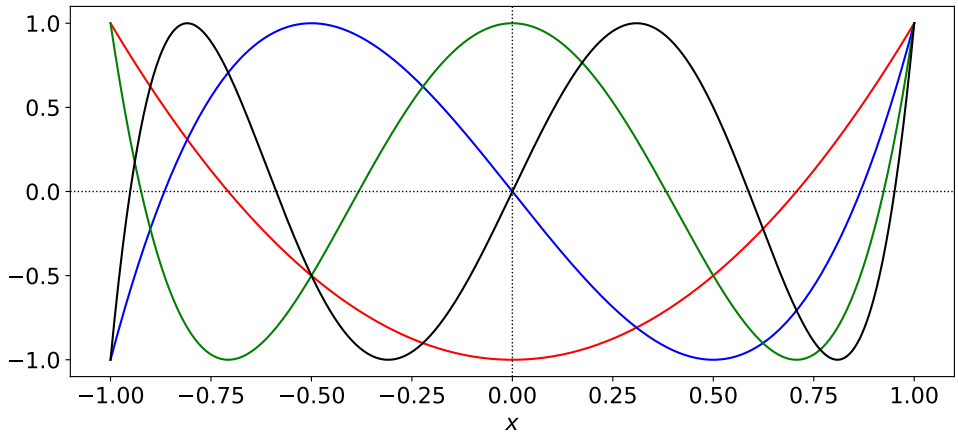
$$T_n(-x) = (-1)^n T_n(x)\,,$$

i.e. when n is even/odd, $T_n(x)$ is an even/odd polynomial respectively. Also, via (1.49) and (1.50), $T_n(x)$ is a polynomial of degree n in which the coefficient of x^n is 2^{n-1}, i.e.

$$T_n(x) = 2^{n-1}x^n + t_{n-1}x^{n-1} + \cdots + t_1 x + t_0\,. \tag{1.51}$$

We shall come back to this point in §1.2.8.

□ **Example 1.13** *Examples of Chebyshev polynomials*



The Chebyshev polynomials

$$\begin{aligned} T_2(x) &= 2x^2 - 1 \\ T_3(x) &= 4x^3 - 3x \\ T_4(x) &= 8x^4 - 8x^2 + 1 \\ T_5(x) &= 16x^5 - 20x^3 + 5x \end{aligned}$$

are plotted in the above figure. Also, note the coefficient of x^n. □

It is clear from (1.48) that

$$|T_n(x)| \le 1 \qquad x \in [-1, 1]\,, \tag{1.52}$$

the bound ± 1 being attained alternately at the $n+1$ *Chebyshev extrema*, $x_k^{(e)}$, where

$$T_n(x_k^{(e)}) = \cos n\theta_k^{(e)} = \pm 1\,, \qquad k = 0(1)n\,.$$

This gives $n\theta_k^{(e)} = k\pi$, therefore, the $n+1$ Chebyshev extrema of $T_n(x)$ are at

$$x_k^{(e)} = \cos\frac{k}{n}\pi\,, \qquad k = 0(1)n\,. \tag{1.53}$$

(1.53) suggests that these Chebyshev extrema, which are distributed more densely towards the endpoints of $[-1,1]$, may be constructed geometrically as projections onto the x-axis of equally-spaced points along the semicircle of unit radius, see the figures below (1.54).

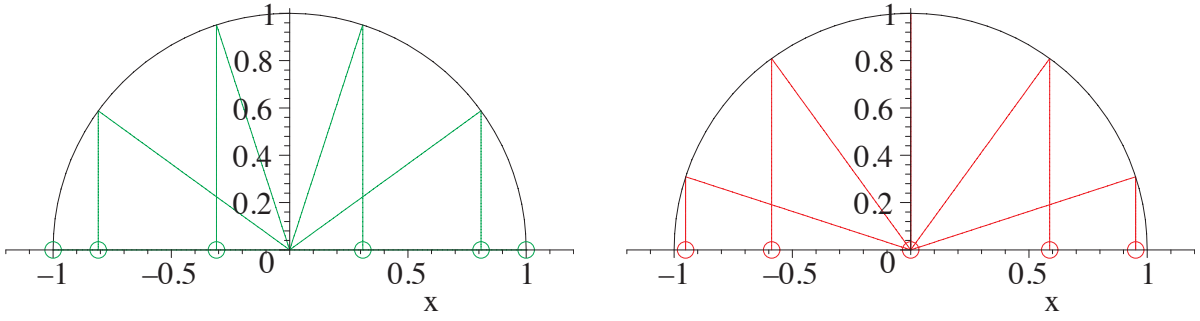
Moreover, by (1.48), $T_n(x)$ switches sign between two consecutive extrema resulting in n zeros, or *Chebyshev roots*, x_k, where

$$T_n(x_k) = \cos n\theta_k = 0 \qquad k = 0(1)n\,.$$

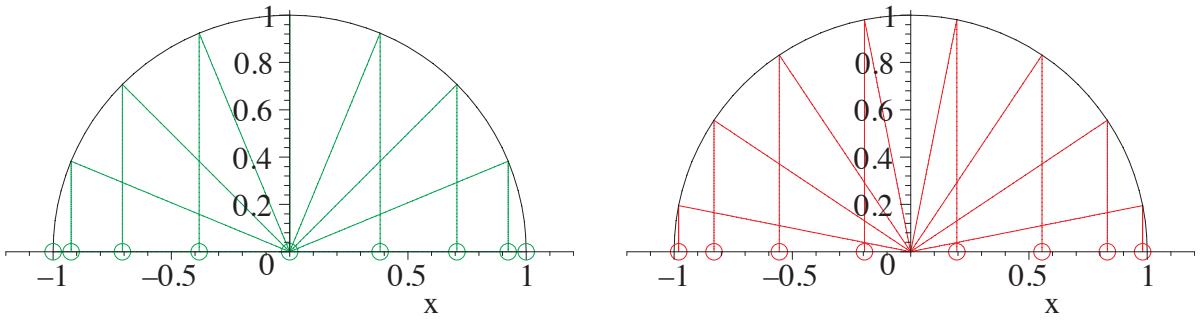
Hence the n roots of $T_n(x)$ are at

$$x_k = \cos\frac{2k-1}{2n}\pi \qquad k = 1(1)n\,. \tag{1.54}$$

So $T_n(x)$ has n zeros and $n+1$ extrema. The following figures show the 6 extreme points (left) and the 5 roots (right) of $T_5(x)$. Note the rotation of $\pi/10$ about $(0,0)$ of the radial lines between the two figures.



The following figures show the 9 extreme points (left) and the 8 roots (right) of $T_8(x)$. Note the rotation of $\pi/16$ about $(0,0)$ of the radial lines between the two figures.



Equations (1.53) and (1.54) explain the half-sector rotations.

As we shall see in §§1.2.7 and 1.2.8, the extrema and roots of the Chebyshev polynomials play an important role in finding so-called *near-minimax* approximations.

1.2.6 Chebyshev least-squares approximation

We can always write a polynomial given in the usual powers of x in terms of the Chebyshev polynomials $T_k(x)$,

$$p_n(x) \equiv \sum_{k=0}^{n} a_k\, x^k = \sum_{k=0}^{n} b_k\, T_k(x)\,.$$

To see this, note that, for example, the first five Chebyshev polynomials (see list at start of Example 1.13) may be inverted to give

$$\begin{aligned} 1 &= T_0\,,\\ x &= T_1\,,\\ x^2 &= \tfrac{1}{2}(T_0 + T_2)\,,\\ x^3 &= \tfrac{1}{4}(3T_1 + T_3)\,,\\ x^4 &= \tfrac{1}{8}(3T_0 + 4T_2 + T_4)\,. \end{aligned}$$

Similar expressions for other x^k allow the relationships between a_k and b_k be readily found. If $u(t)$ is given for $t \in [a,b]$, we use the change of variable $t = \frac{1}{2}[(b+a)+(b-a)x]$ to map onto $x \in [-1,1]$. $u(x)$ can then be approximated using Chebyshev polynomials. Specifically, we use the approximation

$$C_n(x) = {\sum_{k=0}^{n}}' c_k\, T_k(x) = \frac{c_0}{2}T_0(x) + c_1T_1(x) + \cdots + c_nT_n(x)\,, \tag{1.55}$$

in which the coefficients $\{c_0, c_1, \ldots, c_n\}$ are to be found that minimise $\|u - C_n\|$. The prime on the summation means that the first ($k = 0$) term should be halved (see (1.56)). But which norm should we use?

From our discussion of the minimax approximation in §1.2.3, we know that minimising the ∞-norm $\|u - C_n\|_\infty$ gives a *non-linear* system of equations for the c_k in (1.55). By instead minimising $\|u - C_n\|_{w,2}$ (using the norm defined in (1.35)), we obtain a *linear* system of equations for the c_k; the resulting $C_n(x)$ then has a *near-minimax* error (§§1.2.7), and it is called the *least-squares approximation* to $u(x)$.

Using (1.33) with the weight function $w(x) = 1/\sqrt{1-x^2}$ means that the Chebyshev polynomials $T_n(x)$ $(n \geq 0)$ form a system of *orthogonal polynomials* on $[-1,1]$, i.e.

$$(T_j, T_k) = \int_{-1}^{1} \frac{T_j(x)\,T_k(x)}{\sqrt{1-x^2}}dx = \int_0^{\pi} \cos j\theta\, \cos k\theta\, d\theta = \begin{cases} 0 & j \neq k \\ \dfrac{\pi}{2} & j = k > 0 \\ \pi & j = k = 0 \end{cases}\,, \tag{1.56}$$

which is proved using (1.48) and the substitution $x = \cos\theta$. Via (1.35), (1.55) and (1.56) we seek to determine the c_k that minimise $\Phi \equiv \|u - C_n\|^2_{w,2}$, from which

$$\Phi = \frac{\pi}{4}c_0^2 + \frac{\pi}{2}\sum_{k=1}^{n} c_k^2 - c_0\int_{-1}^{1}\frac{u(x)\,dx}{\sqrt{1-x^2}} - 2\sum_{k=1}^{n} c_k \int_{-1}^{1}\frac{u(x)\,T_k(x)\,dx}{\sqrt{1-x^2}} + \int_{-1}^{1}\frac{u(x)^2\,dx}{\sqrt{1-x^2}}\,. \tag{1.57}$$

The necessary conditions for a stationary value of Φ are that

$$\frac{\partial \Phi}{\partial c_k} = 0 \qquad k = 0(1)n$$

which, because Φ is quadratic in the c_k, gives $n+1$ linear equations for $c_0, c_1, \ldots, c_n$. Applying these conditions to (1.57) yields

$$\begin{aligned}
\frac{\partial \Phi}{\partial c_0} &= \frac{\pi}{2} c_0 - \int_{-1}^{1} \frac{u(x)\,dx}{\sqrt{1-x^2}} = 0 \\
\frac{\partial \Phi}{\partial c_k} &= \pi c_k - 2 \int_{-1}^{1} \frac{u(x)\,T_k(x)\,dx}{\sqrt{1-x^2}} = 0 \qquad k = 1(1)n\,,
\end{aligned}$$

and so

$$c_k = \frac{2}{\pi} \int_{-1}^{1} \frac{u(x)\,T_k(x)\,dx}{\sqrt{1-x^2}} \qquad k = 0(1)n\,. \tag{1.58}$$

The substitution $x = \cos\theta$, along with (1.48) and (1.49), gives

$$c_k = \frac{2}{\pi} \int_0^{\pi} u(\cos\theta) \cos k\theta \, d\theta \qquad k = 0(1)n\,. \tag{1.59}$$

The approximation problem is thus reduced to the calculation of Fourier coefficients, efficient computation of which is a (vast) subject in its own right. Although discussion of a cheap method for calculating c_k is deferred to §1.2.10, two points presently arise. First note that (1.59) may be manipulated into the form

$$c_k = \frac{2}{\pi} \int_0^{\pi/2} \left\{ u(\cos\theta) + (-1)^k u(-\cos\theta) \right\} \cos k\theta \, d\theta \qquad k = 0(1)n\,, \tag{1.60}$$

from which it follows that, when $u(x)$ is an even or odd function,

$$c_k = \begin{cases} 0 & u(x) \text{ and } k \text{ of different parity} \\ \dfrac{4}{\pi} \displaystyle\int_0^{\pi/2} u(\cos\theta) \cos k\theta \, d\theta & u(x) \text{ and } k \text{ of the same parity} \end{cases} \qquad k = 0(1)n\,. \tag{1.61}$$

Second, the integrand in (1.59) is an even, 2π-periodic function of θ, and hence

$$c_k = \frac{1}{\pi} \int_{-\pi}^{\pi} u(\cos\theta) \cos k\theta \, d\theta \qquad k = 0(1)n\,, \tag{1.62}$$

i.e. the c_k are the Fourier coefficients of the even, 2π-periodic function $u(\cos\theta)$. In §1.2.10 we show that the simple trapezoidal rule is extremely accurate for computing integrals of the form (1.62).

□ **Example 1.14** *Chebyshev least-squares approximations to* e^x *on* $[-1,1]$
The minimax approximations to e^x on $[-1,1]$ in Example 1.9 for $n=1$ and $n=3$ were

$$q_1^*(x) \approx 1.264279 + 1.175201x \quad \text{and} \quad q_3^*(x) \approx 0.994579 + 0.995668x + 0.542973x^2 + 0.179533x^3$$

respectively. Numerical integration of (1.62) here gives

$$c_0 \approx 2.53213176 \qquad c_1 \approx 1.13031821 \qquad c_2 \approx 0.27149534 \qquad c_3 \approx 0.04433685\,,$$

from which (1.55) gives the Chebyshev least-squares approximations

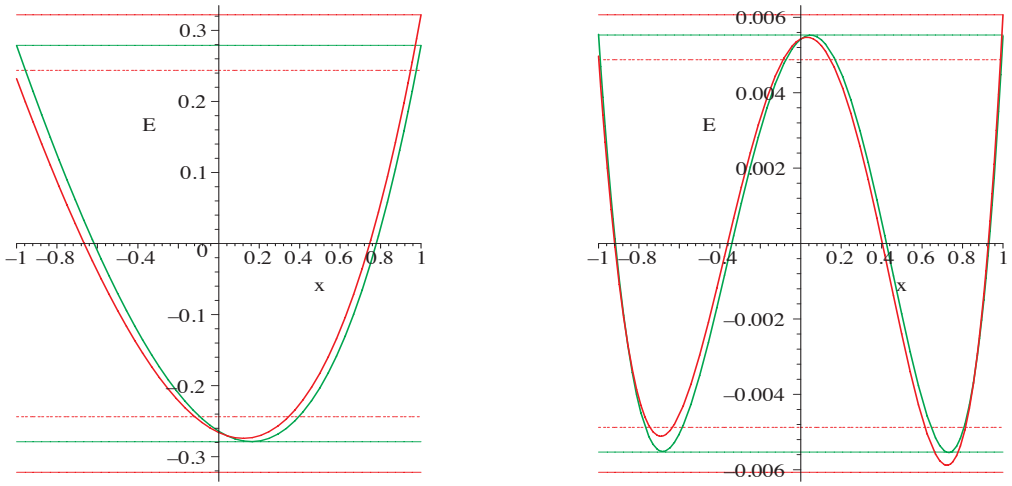
$$C_1(x) \approx 1.266066 + 1.130318x \quad \text{and} \quad C_3(x) \approx 0.994571 + 0.997308x + 0.542991x^2 + 0.177347x^3\,.$$

The average error in approximating $u(x)$ by $C_n(x)$ on $[a,b]$ is given by the *(weighted) root-mean-square error*, defined by (1.35) to be

$$\langle E_n \rangle \equiv \frac{\|u - C_n\|_{w,2}}{\sqrt{b-a}}\,. \tag{1.63}$$

Hence, by (1.34) and (1.39), we presently have

$$\langle E_n \rangle = 2^{-1/2}\,\|u - C_n\|_{w,2} \le \|u - C_n\|_\infty \quad \text{and} \quad \rho_n(u) \equiv \|u - q_n^*\|_\infty \le \|u - C_n\|_\infty\,. \tag{1.64}$$



The graphs show the errors $u(x) - q_n^*(x)$ and $u(x) - C_n(x)$ for $n = 1$ (left) and $n = 3$ (right). The solid lines are $\pm\rho_n(u)$ and $\pm\,\|u - C_n\|_\infty$, and the dotted lines show the root-mean-square error $\pm\,\langle E_n \rangle$ in the Chebyshev least-squares approximations.

For both values of n we have $\langle E_n \rangle < \rho_n(u) < \|u - C_n\|_\infty$, in accordance with (1.64). □

It is clear from Example 1.14 that the Chebyshev least-squares approximation $C_n(x)$ is a very good—in the sense that $\rho_n(u) \approx \|u - C_n\|_\infty$—approximation to the minimax polynomial $q_n^*(x)$ for even the lowest possible value of $n = 1$. We now proceed to explain this "near-minimax" behaviour.

1.2.7 Near-minimax approximation

In proposing the Chebyshev least-squares approximation $C_n(x) = \sum_{k=0}^{n}{}' c_k\,T_k(x)$ in (1.55) we implicitly assume that $u(x)$ on $[-1,1]$ can be represented in the form

$$u(x) = \sum_{k=0}^{\infty}{}' c_k\,T_k(x)\,, \tag{1.65}$$

with convergence from $C_n(x)$ to $u(x)$ holding in the sense that $\lim_{n\to\infty} \|u - C_n\|_{w,2} = 0$ (*cf.* (1.57)). For *uniform* convergence, it can be shown that, if $u(x)$ is sufficiently differentiable on $[-1,1]$,

$$\rho_n(u) \le \|u - C_n\|_\infty \le \left(4 + \frac{4}{\pi^2}\ln n\right)\rho_n(u)\,, \tag{1.66}$$

which says that $C_n(x)$ is a good approximation to $q_n^*(x)$ (because $\ln n$ grows slowly with n). This is a direct result of the rapid decrease (explained in §1.2.10) of $|c_k|$ with increasing k. For example, with $u(x) = \mathrm{e}^x$ in Example 1.14, we have $c_0/c_1 \approx 2.2402$, $c_0/c_2 \approx 9.3266$, $c_0/c_3 \approx 57.111$, $c_0/c_4 \approx 462.55$ and $c_0/c_5 \approx 4663.9$.

Because of this rapid decrease, and provided $c_{n+1} \neq 0$, (1.55) and (1.65) give

$$u(x) - C_n(x) = \sum_{k=n+1}^{\infty} c_k\,T_k(x) \approx c_{n+1}T_{n+1}(x)\,. \tag{1.67}$$

By (1.52), $|T_{n+1}(x)| \le 1$, this bound being attained at the $n+2$ extreme points given by (1.53),

$$x_k^{(e)} = \cos\frac{k\pi}{n+1} \quad k = 0(1)n+1 \qquad \text{at which} \qquad T_{n+1}(x_k^{(e)}) = (-1)^k\,. \tag{1.68}$$

It then follows that the *approximate* error $c_{n+1}T_{n+1}(x)$ in (1.67) has exactly $n+2$ extrema $(-1)^k c_{n+1}$ of alternating sign and equal magnitude $|c_{n+1}|$ at the points $x_k^{(e)}$. By the Chebyshev Alternation Theorem, $C_n(x)$ should be close to the minimax approximation $q_n^*(x)$, in which the $n+2$ extrema have magnitude $\rho_n(u)$. That is, provided that $u(x)$ is sufficiently differentiable, $\rho_n \approx |c_{n+1}|$.

□ **Example 1.15** *Chebyshev-expansion error for* $u(x) = \mathrm{e}^x$ *on* $[-1,1]$
In Examples 1.10 and 1.11 we laboriously found that $\{\rho_1,\rho_2,\rho_3\} \approx \{0.279, 0.0450, 0.00553\}$. Using (1.62), we compute $\{|c_2|,|c_3|,|c_4|\} \approx \{0.272, 0.0443, 0.00543\}$, so that $\rho_n \approx |c_{n+1}|$ as predicted; the good agreement is here due to the infinite differentiability of u. □

□ **Example 1.16** *Chebyshev least-squares approximation of* $u(x) = x^5$ *on* $[-1,1]$
Here n must be ≤ 4. Taking $n = 4$ in (1.55), we need to find the c_k in

$$C_4(x) = {\sum_{k=0}^{4}}' c_k\,T_k(x)\,.$$

Using (1.62), we have

$$c_k = \frac{1}{\pi}\int_{-\pi}^{\pi} u(\cos\theta)\,\cos k\theta\,d\theta = \frac{1}{\pi}\int_{-\pi}^{\pi} \cos^5\theta\,\cos k\theta\,d\theta\,, \qquad k = 0(1)4\,.$$

To perform the integrations, the power form of $\cos^5\theta$ must first be converted to multiple-angle form as follows. Let $z = e^{i\theta}$ so that $\cos\theta = \frac{1}{2}\left(z + \frac{1}{z}\right)$. Then

$$\begin{aligned}\cos^5\theta = \frac{1}{2^5}\left(z+\frac{1}{z}\right)^5 &= \frac{1}{32}\left(z^5 + 5z^3 + 10z + \frac{10}{z} + \frac{5}{z^3} + \frac{1}{z^5}\right)\\ &= \tfrac{5}{16}\left(z + z^{-1}\right) + \tfrac{5}{32}\left(z^3 + z^{-3}\right) + \tfrac{1}{32}\left(z^5 + z^{-5}\right)\,.\end{aligned}$$

Now use

$$z^m + z^{-m} = e^{im\theta} + e^{-im\theta} = 2\cos m\theta\,,$$

so that

$$c_k = \frac{1}{\pi}\int_{-\pi}^{\pi} \left(\tfrac{5}{8}\cos\theta + \tfrac{5}{16}\cos 3\theta + \tfrac{1}{16}\cos 5\theta\right)\cos k\theta\, d\theta \qquad k = 0(1)4\,.$$

We then use (1.56) in the form

$$\int_{-\pi}^{\pi} \cos j\theta\, \cos k\theta\, d\theta = \begin{cases} 0 & j \neq k \\ \pi & j = k > 0 \\ 2\pi & j = k = 0 \end{cases},$$

to yield $c_0 = c_2 = c_4 = 0$, $c_1 = \frac{5}{8}$ and $c_3 = \frac{5}{16}$; we also find $|c_{n+1}| = |c_5| = \frac{1}{16}$. Thus, using the formulae from Example 1.13, we have found that

$$C_4(x) = \tfrac{5}{8}\,T_1(x) + \tfrac{5}{16}\,T_3(x) = \tfrac{5}{16}\,x\,(2x-1)\,(2x+1)\,.$$

By direct calculation,

$$\|u - C_4\|_\infty = \max_{x\in[-1,1]} |u(x) - C_4(x)| = |1 - C_4(1)| = |1 - \tfrac{15}{16}| = \tfrac{1}{16} = |c_5|\,.$$

So here $\|u - C_n\|_\infty = |c_{n+1}|$ whereas, via (1.67), we discovered $\|u - C_n\|_\infty \approx |c_{n+1}|$ in Example 1.15. This is a result of the function $u(x)$ being respectively a finite polynomial and a transcendental function in this and the previous examples. Specifically, here $u(x)$ differs from $C_n(x)$ by only *one term* (of order x^{n+1}), and hence the Fourier coefficients c_k *vanish*, rather than decay, for $k > n+1$; then, $c_{n+1}T_{n+1}$ is the *only* omitted term rather than the largest one; additionally, the roots of $T_{n+1}(x)$ coincide with the minimax nodes.

Note that $c_4 = 0$ because $u(x)$ is an odd function, i.e. C_n is in fact $C_3(x)$ and not $C_4(x)$. Hence here $\|u - C_n\|_\infty = |c_{n+2}|$. The generalisation of this is that, for odd or even $u(x)$, (1.67) must be modified to $u(x) - C_n(x) \approx c_{n+2}T_{n+2}(x)$. $\square$

1.2.8 Chebyshev interpolation

To approximate $u(x)$ using Chebyshev polynomials, an interpolating polynomial is often as suitable as the truncated expansion (1.55). Interpolation has the advantage that it does not require the evaluation of Fourier coefficients. We now show that a polynomial interpolation with nodes at the Chebyshev zeros has near-minimax behaviour.

By (1.67), the error in the near-minimax approximation $C_n(x)$ is approximately $c_{n+1}T_{n+1}(x)$, which vanishes at the $n+1$ roots given by (1.54) as

$$x_k = \cos\frac{2k-1}{2n+2}\pi \quad k = 1(1)n+1\,, \quad \text{equivalently} \quad x_k = \cos\frac{2k+1}{2n+2}\pi \quad k = 0(1)n\,.$$

Now take these roots as the nodes of the Lagrange polynomial $I_n(x)$ (of degree $\leq n$) that interpolates $u(x)$, i.e. suppose we sample $u(x)$ at x_k to obtain $u(x_k)$, then (1.1) gives

$$I_n(x) = \sum_{k=0}^{n} \ell_k(x)\,u(x_k) = \sum_{k=0}^{n} \ell_k(x)\,C_n(x_k) + \sum_{k=0}^{n} \ell_k(x)\,\left(u(x_k) - C_n(x_k)\right)\,.$$

But, by (1.67)

$$u(x_k) - C_n(x_k) \approx c_{n+1}T_{n+1}(x_k) = 0 \quad \text{i.e.} \quad I_n(x) \approx \sum_{k=0}^{n} \ell_k(x)\, C_n(x_k) \underbrace{=}_{\text{think!}} C_n(x)\,, \tag{1.69}$$

i.e. $I_n(x)$, *the Lagrange polynomial whose nodes are the roots of* $T_{n+1}(x)$ *is approximately equal to* $C_n(x)$ so, by the results of §1.2.7, $I_n(x)$ is a near-minimax approximation to $u(x)$.

[Warning: if $c_{n+1} = 0$ we may have to interpolate using the roots of $T_{n+2}(x)$ or higher. This usually happens when $u(x)$ is either even or odd on $[-1, 1]$.]

Another way to see that $I_n(x)$ is a near-minimax approximation requires the following theorem on *monic polynomials*; those in which the coefficient of the highest power of x is 1. First note that (1.51),

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

effectively means that $2^{1-n}\,T_n(x)$ is a monic polynomial which, by (1.52), takes extreme values $\pm 2^{1-n}$.

Theorem Among all monic polynomials $M_n(x)$ of degree $n \geq 1$, the polynomial $2^{1-n}\,T_n(x)$ has the smallest ∞-norm on $[-1, 1]$, i.e.

$$\max_{x\in[-1,1]} |M_n(x)| \geq \max_{x\in[-1,1]} |2^{1-n}\,T_n(x)| = 2^{1-n}\,, \tag{1.70}$$

with equality occurring if and only if $M_n(x) \equiv 2^{1-n}\,T_n(x)$. ◇

Consider now the problem of determining $n + 1$ nodes in $[-1, 1]$ that minimise the error

$$u(x) - p_n(x) = \frac{u^{(n+1)}(\xi)}{(n+1)!}\Psi_n(x) \qquad \xi \in [-1, 1]$$

in the Lagrange interpolation $p_n(x)$ (*cf.* (1.9)). Here ξ depends upon the x_k (and x), but not in a way that we can determine explicitly. In order to minimise $\|u - p_n\|_\infty$, we therefore focus on Ψ_n and determine the nodes x_k that minimise

$$\|\Psi_n\|_\infty \equiv \max_{x\in[-1,1]} |(x - x_0)(x - x_1)\ldots(x - x_n)|\,.$$

Clearly $\Psi_n(x)$ is a monic polynomial of degree $n + 1$ so, by the above theorem, $\|\Psi_n\|_\infty$ equals the minimum value of 2^{-n} when the x_k are the roots of $T_{n+1}(x)$, with which choice of nodes $\Psi_n(x) = 2^{-n}\,T_{n+1}(x)$, $p_n(x) = I_n(x)$ and

$$\|u - I_n\|_\infty = \frac{\left\|u^{(n+1)}(\xi)\Psi_n\right\|_\infty}{(n+1)!} \leq \frac{\left\|u^{(n+1)}(\xi)\right\|_\infty \|\Psi_n\|_\infty}{(n+1)!} = \frac{\left\|u^{(n+1)}\right\|_\infty}{2^n(n+1)!}\,, \tag{1.71}$$

which bound is usually over-pessimistic in practice. Here $I_n(x)$ may also be constructed using the divided-difference form (1.18) of the interpolating polynomial. Note that this minimisation of $\|\Psi_n\|_\infty$ fully explains the suppression, observed in Example 1.7, of the Runge phenomenon.

The previous theorem (1.71) also allows us to prove the following bound on $\rho_n(u)$ over the interval $[a, b]$:

$$\rho_n(u) \leq \left(\frac{b-a}{2}\right)^{n+1} \frac{\left\|u^{(n+1)}\right\|_\infty}{2^n(n+1)!}\,. \tag{1.72}$$

Finally, a further result illustrating the near-minimax behaviour of $I_n(x)$ is (without proof)

$$\|u - I_n\|_\infty \leq \left\{ \frac{2}{\pi} \ln(n+1) + 2 \right\} \rho_n(u) \,. \tag{1.73}$$

□ **Example 1.17** *Chebyshev interpolation of* $u(x) = \mathrm{e}^x$ *on* $[-1, 1]$
In Example 1.9 we introduce the minimax expansions

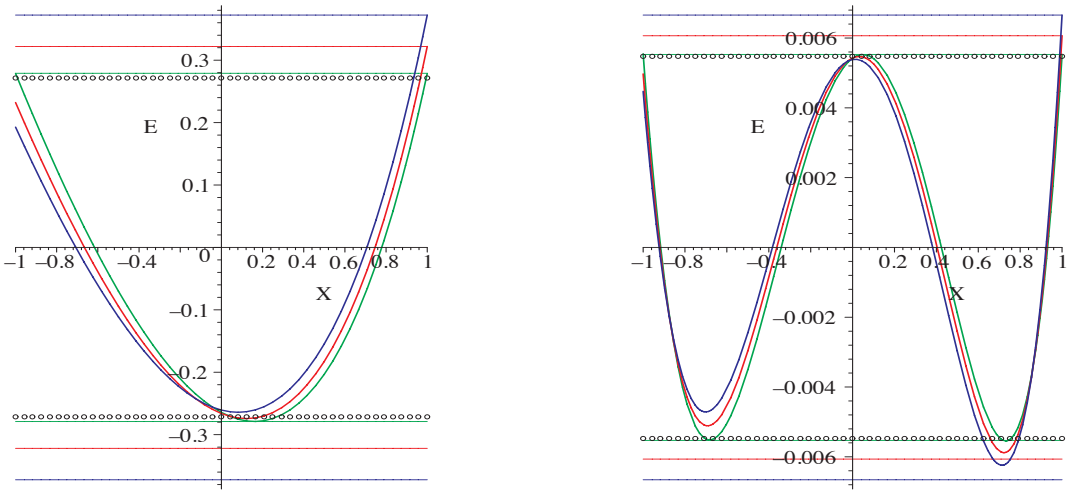
$q_1^*(x) \approx 1.264279 + 1.175201x$ and $q_3^*(x) \approx 0.994579 + 0.995668x + 0.542973x^2 + 0.179533x^3$

and in Example 1.14 we compute the Chebyshev least-squares approximations

$C_1(x) \approx 1.266066 + 1.130318x$ and $C_3(x) \approx 0.994571 + 0.997308x + 0.542991x^2 + 0.177347x^3$.

Here, we find that the Chebyshev interpolating polynomials are

$I_1(x) \approx 1.260592 + 1.085442x$ and $I_3(x) \approx 0.994615 + 0.998933x + 0.542901x^2 + 0.175176x^3$.



The graphs show the errors $u(x) - q_n^*(x)$, $u(x) - C_n(x)$ and $u(x) - I_n(x)$ for $n = 1$ (left) and $n = 3$ (right). The solid lines show $\pm\rho_n(u)$, $\pm\|u - C_n\|_\infty$ and $\pm\|u - I_n\|_\infty$. Also shown as a dotted line is $\pm|c_{n+1}|$ which, by (1.67), should be approximately $\pm\|u - C_n\|_\infty$. For this example, the Fourier coefficient provides the even more impressive result that $|c_{n+1}| \approx \rho_n(u)$.

It is noteworthy that both $C_n(x)$ and $I_n(x)$ have smaller errors than $q_n^*(x)$ for $x \in [-1, 0]$, and that the situation is reversed for $x \in [0, 1]$. In this sense, the ∞-norm is uncompromising. However, the weighted 2-norm (1.35) will incorporate information from the entire interval $[-1, 1]$. To this end, if we use (1.35) and (1.63) to define the weighted root-mean-square error

$$\langle E(p_n) \rangle \equiv \left\{ \frac{1}{2} \int_{-1}^{1} \frac{\{u(x) - p_n(x)\}^2}{\sqrt{1 - x^2}} dx \right\}^{1/2} = 2^{-1/2} \|u - p_n\|_{w,2} \,,$$

we obtain the following, in which there is far more parity between columns:

n	$\langle E(q_n^*) \rangle$	$\langle E(C_n) \rangle$	$\langle E(I_n) \rangle$
1	2.4708×10^{-1}	2.4384×10^{-1}	2.4716×10^{-1}
2	3.9896×10^{-2}	3.9594×10^{-2}	3.9893×10^{-2}
3	4.8994×10^{-3}	4.8754×10^{-3}	4.8992×10^{-3}

By construction, $C_n(x)$ is the optimum approximation under this norm. □

1.2.9 Forced oscillation of the Chebyshev error

One final near-minimax approximation is now considered. In §1.2.4, the Chebyshev Alternation Theorem (1.47) states that the error of the best (minimax) approximation $q_n^*(x)$ to $u(x)$ has (at least) $n+2$ local maxima and minima of equal magnitude $\rho_n(u)$ and opposite sign alternating over the alternating set $\{\xi_0, \xi_1, \ldots, \xi_{n+1}\}$:

$$\underset{?}{u(\xi_k)} - \underset{?}{q_n^*(\xi_k)} = (-1)^k \left[\pm \underset{?}{\rho_n(u)}\right] \qquad k = 0(1)n+1\,. \tag{1.74}$$

Determining the *initially unknown* ξ_k's and $\rho_n(u)$ solves the minimax problem. In §1.2.7, we argue that the Chebyshev least-squares approximation $C_n(x)$ is a near-minimax approximation because its error has $n+2$ local maxima and minima of approximately equal magnitudes ($\approx |c_{n+1}|$) and opposite sign alternating over locations approximately given by $x_k^{(e)}$, $k = 0(1)n+1$. Recall that $x_k^{(e)}$ are the extrema of $T_{n+1}(x)$ in (1.68).

Here, we take the above argument in the opposite direction and propose an approximation $F_n(x)$ to $u(x)$ on $[-1,1]$ that is a polynomial of degree $\le n$ satisfying

$$\underset{\checkmark}{u(x_k^{(e)})} - \underset{\checkmark}{F_n(x_k^{(e)})} = (-1)^k \underset{?}{\phi_n} \qquad k = 0(1)n+1\,, \tag{1.75}$$

in which we have introduced the unknown ϕ_n which we *hope* will be non-zero. In other words, we force the error of $F_n(x)$ to alternate *exactly* at $x_k^{(e)}$ between *equal* and opposite values that may not necessarily be extrema (but are hopefully nearly so). By (1.46) in the theorem of de la Vallée-Poussin, we shall have

$$|\phi_n| \le \rho_n(u) \le \|u - F_n\|_\infty\,. \tag{1.76}$$

Because the alternating set in (1.75) is known, the problem of determining F_n becomes linear and we can proceed without needing the Remez algorithm.

With the connection between $F_n(x)$ and $C_n(x)$ in mind, we recall (1.55) and propose $F_n(x)$ as

$$F_n(x) = {\sum_{k=0}^{n}}' c_{n,k}\, T_k(x)\,, \tag{1.77}$$

in which the coefficients $\{c_{n,0}, c_{n,1}, \ldots, c_{n,n}\}$ are to be found that satisfy (1.75). From the above discussion, we expect the $c_{n,k}$ to be approximations to the Fourier coefficients c_k in $C_n(x)$. From (1.75) and (1.77), we have

$${\sum_{k=0}^{n}}' c_{n,k}\, T_k(x_i^{(e)}) + (-1)^i \phi_n = u(x_i^{(e)}) \qquad i = 0(1)n+1\,, \tag{1.78}$$

which are $n+2$ equations for $\{c_{n,0}, c_{n,1}, \ldots, c_{n,n}\}$ and ϕ_n. Note that, by (1.68) and (1.48),

$$T_k(x_i^{(e)}) = T_k\left(\cos\frac{i\pi}{n+1}\right) = \cos\frac{ik\pi}{n+1}\,,$$

so that $T_{n+1}(x_i^{(e)}) = \cos i\pi = (-1)^i$. Now introduce $\phi_n = c_{n,n+1}/2$, so that (1.78) can be written in the concise form

$$\sum_{k=0}^{n+1}{}'' c_{n,k} \cos\frac{ik\pi}{n+1} = u(x_i^{(e)}) \qquad i = 0(1)n+1\,, \tag{1.79}$$

wherein the double prime means the first and last terms in the sum are halved. Eqn. (1.79) are now $n+2$ equations for $c_{n,k}$, $k = 0(1)n+1$, of which the last one is $2\phi_n$. These equations can be inverted using discrete versions of the orthogonality relations (1.56),

$$\sum_{i=0}^{n+1}{}'' \cos\left(j\frac{i\pi}{n+1}\right)\cos\left(k\frac{i\pi}{n+1}\right) = \begin{cases} n+1 & j=k=0 \text{ or } n+1 \\ \dfrac{n+1}{2} & 0<j=k<n+1 \\ 0 & j\neq k,\ 0\le j,k\le n+1\,. \end{cases} \tag{1.80}$$

To pick out the jth coefficient in (1.79), multiplying it by $\cos(j\dfrac{i\pi}{n+1})$ and sum over i. Applying (1.80) then gives,

$$c_{n,j} = \frac{2}{n+1}\sum_{i=0}^{n+1}{}'' u(x_i^{(e)}) \cos\frac{ij\pi}{n+1} \qquad j = 0(1)n+1\,, \tag{1.81}$$

so that, when $j = n+1$,

$$\phi_n = \frac{1}{n+1}\sum_{i=0}^{n+1}{}''(-1)^i\, u(x_i^{(e)})\,. \tag{1.82}$$

We can now show that the coefficients $c_{n,j}$ in $F_n(x)$ are indeed approximations to the Fourier coefficients c_j in $C_n(x)$. In (1.59), evaluate the integral using the trapezoidal rule with $n+1$ subdivisions; as will be proved in §1.2.10, this rule is *spectrally accurate* for periodic integrands. We find

$$\begin{aligned} c_j &= \frac{2}{\pi}\int_0^\pi u(\cos\theta)\cos j\theta\, d\theta \qquad j = 0(1)n \\ &\approx \frac{2}{\pi}\sum_{k=0}^{n+1}{}'' u\left(\cos\frac{k\pi}{n+1}\right)\cos\frac{jk\pi}{n+1}\,\frac{\pi}{n+1} \\ &= \frac{2}{n+1}\sum_{k=0}^{n+1}{}'' u(x_k^{(e)})\cos\frac{jk\pi}{n+1} \\ &= c_{n,j}\,. \end{aligned} \tag{1.83}$$

The orthogonality relations used in deriving (1.81) can also be used to remove the "$\approx$" sign in (1.83) to give

$$c_{n,j} = c_j + c_{2(n+1)-j} + c_{2(n+1)+j} + c_{4(n+1)-j} + c_{4(n+1)+j} + \cdots \tag{1.84}$$

so that, when the Fourier coefficients decrease rapidly (see §1.2.10), $F_n(x)$ is both a good approximation to $C_n(x)$ and is easier to calculate than $C_n(x)$ because it employs summation rather than integration.

$F_n(x)$ is also comparable with $I_n(x)$ in the sense that it can be proved that

$$\|u - F_n\|_\infty \leq \omega(n)\,\rho_n(u)\,,$$

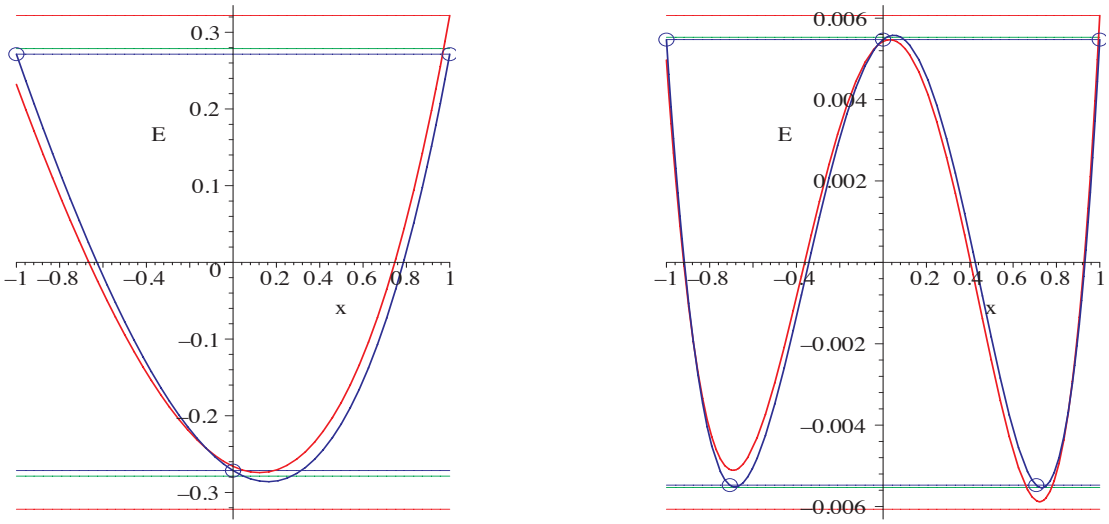
with $\omega(n)$ empirically nearly equal to the bounding coefficient in (1.73).

□ **Example 1.18** *Forced-oscillation approximation to* $u(x) = \mathrm{e}^x$ *on* $[-1, 1]$
Recall the Chebyshev least-squares approximations

$C_1(x) \approx 1.266066{+}1.130318x$ and $C_3(x) \approx 0.994571{+}0.997308x{+}0.542991x^2{+}0.177347x^3$.

from Example 1.14. Using (1.77), (1.81) and (1.82), we find

$F_1(x) \approx 1.271540{+}1.175201x$ and $F_3(x) \approx 0.994526{+}0.995682x{+}0.543081x^2{+}0.179519x^3$.



The graphs show the errors $u(x) - C_n(x)$ and $u(x) - F_n(x)$ for $n = 1$ (left) and $n = 3$ (right). The solid lines show $\pm\rho_n(u)$, $\pm\,\|u - C_n\|_\infty$ and $\pm\phi_n$. The blue circles have abscissae at the Chebyshev extrema and ordinates at $\pm\phi_n$; note that these ordinates are exceeded by $\pm\,\|u - F_n\|_\infty$.

The bounds (1.76) predicted by the theorem of de la Vallée-Poussin are corroborated in the following table (and, albeit hardly visible on the scale presented, on the above graphs).

n	ϕ_n	$\rho_n(u)$	$\|u - F_n\|_\infty$
1	2.715×10^{-1}	2.788×10^{-1}	2.861×10^{-1}
2	4.443×10^{-2}	4.502×10^{-2}	4.547×10^{-2}
3	5.474×10^{-3}	5.528×10^{-3}	5.581×10^{-3}

The following table shows the rapid convergence with n of the coefficients $c_{n,j}$ to c_j.

j	$c_{1,j}$	$c_{2,j}$	$c_{3,j}$	$c_{4,j}$	$\cdots$	c_j
0	2.543080634	2.532221710	2.532132152	2.532131756	$\cdots$	2.532131755
1	1.175201194	1.130864333	1.130321417	1.130318219	$\cdots$	1.130318208
2	(0.271540317)	0.276969779	0.271540317	0.271495538	$\cdots$	0.271495340
3		(0.044336861)	0.044879777	0.044340049	$\cdots$	0.044336850
4			(0.005474240)	0.005519217	$\cdots$	0.005474240
5				(0.000542926)	$\cdots$	0.000542926

The values of ϕ_n are shown in parentheses at the bottom of each n-dependent column; these reveal that $\phi_n \approx c_{n+1}$ $(n \geq 1)$, in keeping with the theory of Chebyshev interpolation. Recalling also from Example 1.17 that $c_{n+1} \approx \rho_n(u)$, the forced-oscillation, interpolation and minimax approximations are consistent. □

The four methods for approximating functions considered so far are compared below.

Minimax $q_n^*(x) = \sum_{k=0}^{n} a_k x^k$	• minimise $\|u - q_n^*\|_\infty$ • solve nonlinear equations using Remez algorithm for ξ_i and $\rho_n(u)$	• extrema of error at ξ_i • $\|E_n(\xi_i)\| = \rho_n(u)$ for all i, i.e. all extrema have equal magnitudes
Chebyshev least-squares $C_n(x) = \sum_{k=0}^{n}{}' c_k T_k(x)$	• minimise $\|u - C_n\|_{w,2}$ • c_k are Fourier coefficients using integral formula (1.59)	• extrema of error at $x_i^* \approx x_k^{(e)}$ of $T_{n+1}(x)$ • unequal $\|E_n(x_i^*)\| \approx \|c_{n+1}\| \approx \rho_n(u)$ when u is sufficiently differentiable
Chebyshev interpolation $I_n(x) = \sum_{k=0}^{n} \ell_k(x) u(x_k)$	• $I_n(x)$ passes through the $n+1$ zeros of $T_{n+1}(x)$ • Lagrange formula or Newton divided difference	similar to the case of $C_n(x)$
Forced error oscillation $F_n(x) = \sum_{k=0}^{n}{}' c_{n,k} T_k(x)$	• requires $E_n(x)$ to alternate between $\pm\phi_n$ at $x_k^{(e)}$ • $c_{n,k}$ as a series summation, equals to trapezoidal approximation of c_k	• $\|E_n(x_k^{(e)})\| = \|\phi_n\|$ for all k • generally $\|\phi_n\| \neq$ maximum error

1.2.10 Spectrally accurate computation of rapidly decaying Fourier coefficients

Integrals such as those in (1.62), *with 2π-periodic integrands*, can be calculated with great accuracy using the simple trapezoidal rule. Consider computing the integral

$$\mathcal{I} \equiv \int_{-\pi}^{\pi} f(\theta)\, d\theta = \int_{0}^{2\pi} f(\theta)\, d\theta\,, \tag{1.85}$$

in which $f(\theta)$ and its first r derivatives are 2π-periodic. We begin by writing $f(\theta)$ as the complex Fourier expansion

$$f(\theta) = \sum_{k=-\infty}^{\infty} c_k\, e^{ik\theta} \qquad \text{where} \qquad c_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(\theta)\, e^{-ik\theta}\, d\theta\,. \tag{1.86}$$

The N-subdivision trapezoidal-rule approximation to $\mathcal{I}$ has stepsize $h = 2\pi/N$ and so $\theta_n \equiv nh = 2\pi n/N$. Then

$$\mathcal{T}_N \equiv h \sum_{n=0}^{N}{}'' f(\theta_n) \underbrace{=}_{\text{think!}} \frac{2\pi}{N} \sum_{n=0}^{N-1} f(\theta_n)\,. \tag{1.87}$$

From (1.86) and (1.87),

$$\mathcal{T}_N = \frac{2\pi}{N}\sum_{n=0}^{N-1}\underbrace{\sum_{k=-\infty}^{\infty} c_k\, e^{ik2\pi n/N}}_{=f(\theta_n)\text{ by (1.86)}} = \frac{2\pi}{N}\sum_{k=-\infty}^{\infty} c_k \underbrace{\sum_{n=0}^{N-1}(e^{ik2\pi/N})^n}_{\equiv G,\text{ say}} .$$

Here, the order of summation can be interchanged due to the convergence of the infinite series in (1.86). The geometric sum denoted by G is equal to $\dfrac{1-e^{i2\pi k}}{1-e^{i2\pi k/N}}$, which is zero for all k for which k/N is not an integer. When k/N is an integer, $j \in \mathbb{N}$ say, G is equal to N. Thus, with $k = jN$,

$$\mathcal{T}_N = \frac{2\pi}{N}\sum_{j=-\infty}^{\infty} c_{jN}\times N = 2\pi\sum_{j=-\infty}^{\infty} c_{jN} = 2\pi\left[c_0 + \sum_{j=1}^{\infty}(c_{jN}+c_{-jN})\right].$$

Since $2\pi c_0 = \mathcal{I}$ by (1.85) and (1.86), the trapezoidal-rule error is

$$\mathcal{T}_N - \mathcal{I} = 2\pi(c_N + c_{-N} + c_{2N} + c_{-2N} + \cdots)\,. \tag{1.88}$$

For $k \geq 1$, (1.86) gives

$$c_k + c_{-k} = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(\theta)\,(e^{-ik\theta}+e^{ik\theta})\,d\theta = \frac{1}{\pi}\int_{-\pi}^{\pi} f(\theta)\,\cos k\theta\,d\theta \equiv a_k\,, \tag{1.89}$$

which is the kth Fourier-cosine coefficient in the real Fourier expansion of $f(\theta)$,

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

Then by (1.88) and (1.89),

$$\mathcal{T}_N - \mathcal{I} = 2\pi(a_N + a_{2N} + a_{3N} + \cdots)\,. \tag{1.90}$$

We now show that $|a_k|$ decreases rapidly with k; this rapid convergence has already been seen in practice in Examples 1.14 and 1.18. Integration by parts on the expression for a_k gives

$$\pi a_k = \int_{-\pi}^{\pi} f(\theta)\,\cos k\theta\,d\theta = \underbrace{\Big[f'(\theta)\,\cos k\theta\Big]_{-\pi}^{\pi}}_{=0\text{ by periodicity}} - \frac{1}{k}\int_{-\pi}^{\pi} f'(\theta)\,\sin k\theta\,d\theta\,,$$

and so, because $|\sin k\theta| \leq 1$ for all k and θ, we have

$$|a_k| \leq \frac{1}{\pi k}\int_{-\pi}^{\pi}|f'(\theta)|\,d\theta\,,$$

and a further integration by parts gives, on using the 2π-periodicity of $f''(\theta)$,

$$|a_k| \leq \frac{1}{\pi k^2}\int_{-\pi}^{\pi}|f''(\theta)|\,d\theta\,.$$

Continuing this process and remembering that f has r 2π-periodic derivatives, we arrive at

$$|a_k| \leq \frac{1}{\pi k^r} \int_{-\pi}^{\pi} \left| f^{(r)}(\theta) \right| \, d\theta = \mathcal{O}\left(k^{-r}\right) \, . \tag{1.91}$$

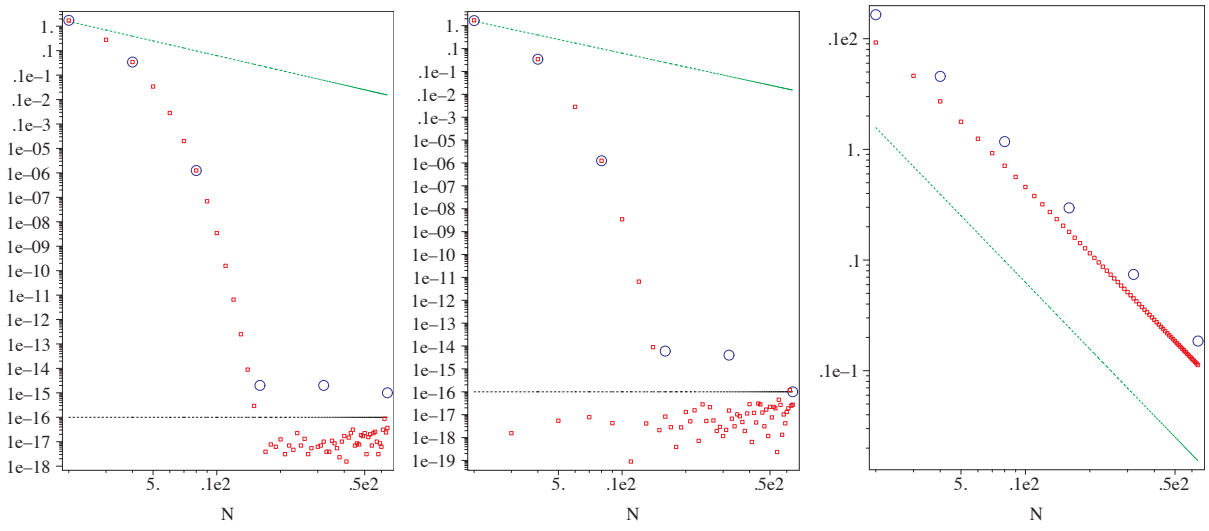
Combining (1.90) and (1.91), the trapezoidal-rule error (based upon $N > 1$ subdivisions) is bounded according to

$$|\mathcal{T}_N - \mathcal{I}| = \mathcal{O}\left(N^{-r}\right) \, . \tag{1.92}$$

Thus when $f(\theta)$ is infinitely differentiable on $[-\pi, \pi]$, i.e. $r \to \infty$, the N-panel trapezoidal-rule error converges to zero *faster than any power of* $1/N$, and is referred to as *spectrally accurate*. Recall that, in stark contrast, the trapezoidal-rule error converges merely as $1/N^2$ for non-periodic integrands, *irrespective of their differentiability*.

Thus the N-division trapezoidal-rule computation of c_k on $[-\pi, \pi]$ in (1.62) can be performed with spectral accuracy when $u(x)$ is sufficiently differentiable on $[-1, 1]$. The same accuracy is obtained by using only $N/2$ intervals on $[0, \pi]$ in (1.59). If moreover $u(x)$ is an even or odd function, the same accuracy requires using only $N/4$ intervals on $[0, \frac{\pi}{2}]$ in (1.61).

□ **Example 1.19** *Spectrally accurate trapezoidal-rule integration*
The following graphs show log-log plots for the 2π-periodic functions $f(\theta) = e^{\cos\theta}$ (left panel) and $f(\theta) = e^{\sin\theta}$ (middle panel) on $[-\pi, \pi]$. In both figures the large circles and small squares respectively indicate the values of the computed trapezoidal-rule errors $|\mathcal{T}_N - \mathcal{I}|$ and $2\pi|a_N|$, the latter of which is the modulus of the first term in the theoretical error expansion (1.90).



The figures above raise several noteworthy points.

[1] The accuracy of (1.90) and the rapid decay of $|a_N|$ predicted by (1.91) are both corroborated.

[2] The non-decrease and clustering of data for the higher values of N is a manifestation of the calculations having reached the *roundoff plateau* of the machine, shown as a horizontal line; here, 16-digit accuracy was used.

[3] For $f(\theta) = e^{\cos\theta}$, all a_N are non-zero, but for $f(\theta) = e^{\sin\theta}$, $a_N = 0$ when N is odd. This explains the squares at the roundoff plateau for the low values of N in the right-hand figure.

[4] For non-periodic functions, the trapezoidal-rule error decreases as $1/N^2$. This rate of convergence is indicated on both plots above by sloping straight lines.

[5] For comparison, the panel on the right shows the corresponding results for the *non-periodic* function $f(\theta) = e^{\theta}$. When $f(\theta)$ is not 2π-periodic, both $|\mathcal{T}_N - \mathcal{I}|$ and $2\pi|a_N|$ clearly converge as N^{-2}. $\square$

2 Numerical Differentiation

Examples 1.3 and 1.5 reveal that an approximate derivative of a function given at discrete data points can be obtained by differentiating the data's interpolating polynomial. Finding derivatives on such discrete *meshes* of nodes is invaluable in solving both ODEs and PDEs.

2.1 Finite differences in 1-D

We assume that the mesh comprises regularly spaced nodes at $\{x_0, ..., x_n\}$, the constant node separation being $h = (x_n - x_0)/n$. We use the notation $u(x_i) = u_i$, $u'(x_i) = u'_i$, $u''(x_i) = u''_i$, etc., $i = 0(1)n$. By Taylor's theorem (0.4) expanded about the *base point* x_i,

$$u_{i+1} \equiv u(x_{i+1}) \equiv u(x_i + h) = u_i + hu'_i + \frac{h^2}{2}u''_i + \frac{h^3}{6}u'''_i + O(h^4), \tag{2.1}$$

where $O(h^4)$ (read "big-oh of h to the 4") denotes terms containing fourth and higher powers of h; this is referred to as a *fourth-order error*. From (2.1) we have the *forward-difference* (FD) approximation to u'_i,

$$\frac{u_{i+1} - u_i}{h} = u'_i + \frac{h}{2}u''_i + O(h^2)\,, \tag{2.2}$$

which is *first-order* because the leading error term is $O(h)$. We denote by FD'_1 a first-order FD formula for u'_i. Analogous to (2.1) there is

$$u_{i-1} \equiv u(x_{i-1}) \equiv u(x_i - h) = u_i - hu'_i + \frac{h^2}{2}u''_i - \frac{h^3}{6}u'''_i + O(h^4), \tag{2.3}$$

which yields the *first-order backward-difference* (BD'_1) approximation to u'_i,

$$\frac{u_i - u_{i-1}}{h} = u'_i - \frac{h}{2}u''_i + O(h^2)\,. \tag{2.4}$$

Note that (2.2) and (2.4) are identical to (1.11) and (1.24), obtained by Lagrangian and divided-difference interpolation respectively.

Subtracting (2.3) from (2.1) and rearranging yields the more accurate *second-order central-difference* (CD'_2) approximation to u'_i,

$$\frac{u_{i+1} - u_{i-1}}{2h} = u'_i + \frac{h^2}{6}u'''_i + O(h^4)\,, \tag{2.5}$$

which we did not compute by interpolation.

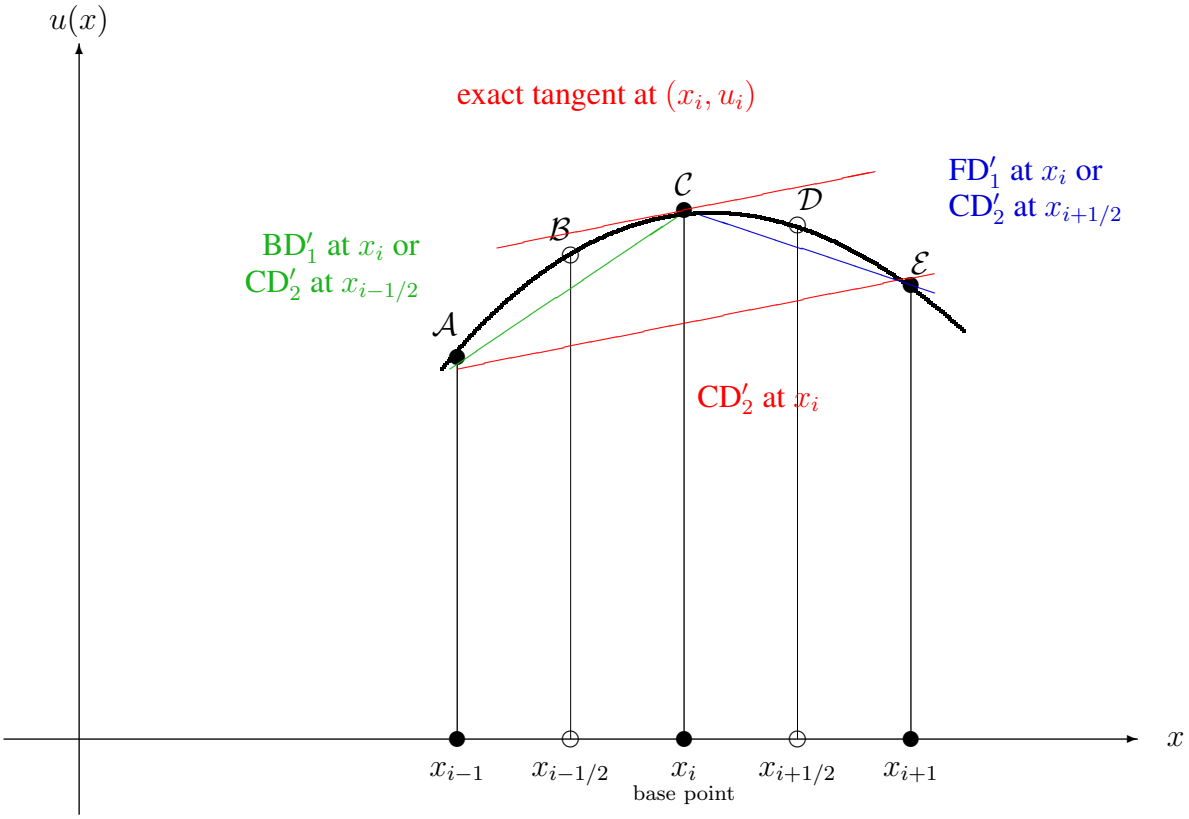
To derive an approximation for the second derivatives u''_i, we take the sum of (2.1) and (2.3). Rearrangement of terms then yields the CD''_2 approximation to u''_i,

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = u''_i + \frac{h^2}{12}u^{\text{iv}}_i + O(h^4)\,, \tag{2.6}$$

where u^{iv}_i is the fourth derivative of u at x_i (try to derive the coefficient $h^2/12$ yourself). In order to obtain both higher-order derivatives and error terms, we simply use more terms in the Taylor

series, thereby increasing the number of nodes required. Note that first-order formulae for first derivatives are *exact* for linear functions, second-order formulae exact for quadratic functions, etc., so that nth-order schemes give the exact derivative for a polynomial of degree n.

□ **Example 2.1** *Geometric interpretation of finite-difference formulae*
The geometric interpretation of the superior accuracy of the CD'_2 formula (2.5) over the FD'_1 and BD'_1 formulae (2.2) and (2.4) is given in the following figure.



- The CD'_2, FD'_1 and BD'_1 formulae (2.5), (2.2) and (2.4) respectively approximate the slopes of the chords $\mathcal{AE}$, $\mathcal{CE}$ and $\mathcal{AC}$, the first of these most closely approximating u'_i, the slope of the tangent at $\mathcal{C}$. This is consistent with (2.5) being $O(h^2)$ and (2.2) and (2.4) being merely $O(h)$.

- The slope of the chord $\mathcal{CE}$ most closely approximates $u'_{i+1/2}$, the slope of the tangent at $\mathcal{D}$. This is because the FD'_1 formula (2.2) for u'_i is in fact a CD'_2 formula for $u'_{i+1/2}$, but on a mesh with spacing $h/2$. Hence replacing h by $h/2$ and moving the required derivative point from x_i to $x_{i+1/2}$ in (2.5), we have

$$\frac{u_{i+1}-u_i}{h} = u'_{i+1/2} + \frac{h^2}{24}u'''_{i+1/2} + O(h^4)\,. \tag{2.7}$$

- Similarly, by (2.5) we have

$$\frac{u_i-u_{i-1}}{h} = u'_{i-1/2} + \frac{h^2}{24}u'''_{i-1/2} + O(h^4)\,. \tag{2.8}$$

  Hence, the slope of the chord $\mathcal{AC}$ most closely approximates $u'_{i-1/2}$, the slope of the tangent at $\mathcal{B}$. □

2.1.1 Higher-order accuracy and/or derivatives

Order the nodes $x_0 < x_1 < \ldots < x_n$. At the LH-end node x_0 the FD$'_1$ (2.2) computes u'_0 to $O(h)$ accuracy and, at the RH-end node x_n, the BD$'_1$ (2.4) computes u'_n to $O(h)$ accuracy. At either end node, the $O(h^2)$ CD$'_2$ (2.5) cannot be used.

We shall construct a BD$'_2$ formula for u'_n. The BD$'_1$ (2.4) applied at the base point x_n,

$$\frac{u_n - u_{n-1}}{h} = u'_n - \frac{h}{2}u''_n + O(h^2)\,,$$

uses two data (i.e. function values) u_n and u_{n-1}: reducing the error from $O(h)$ to $O(h^2)$ clearly requires the additional datum u_{n-2}. So the first step is to postulate the BD$'_2$ formula for u'_n as

$$\frac{w_{n-2}u_{n-2} + w_{n-1}u_{n-1} + w_n u_n}{h} = u'_n + Ch^2 + O(h^3) \tag{2.9}$$

in which the *weights* w_{n-2}, w_{n-1} and w_n and the constant C are to be determined. Note that (i) three data points are involved, (ii) it is h^1 in the denominator on the left since we seek approximation to a first derivative and (iii) we require the error to be proportional to $O(h^2)$, hence the term Ch^2 on the right. Here we can state that the neglected terms are $O(h^3)$ because the three data are *asymmetric* with respect to the base point, and so the series commencing with Ch^2 increases in *single powers of* h. If, however, we change the base point from x_n to x_{n-1}, the three data are *symmetric* with respect to the base point, and so the series commencing with Ch^2 will increase in *double powers of* h, whence the CD$'_2$ formula for u'_{n-1} would be

$$\frac{w_{n-2}u_{n-2} + w_{n-1}u_{n-1} + w_n u_n}{h} = u'_{n-1} + Ch^2 + O(h^4)\,.$$

Back to (2.9), the next step is to express u_{n-1} and u_{n-2} in terms of quantities at x_n: u_n, u'_n, etc and then match the terms on both sides of (2.9). Taylor expansions about the base-point x_n give

$$\begin{aligned} u_{n-1} &= u_n - hu'_n + \frac{h^2}{2}u''_n - \frac{h^3}{6}u'''_n + O(h^4)\,,\\ u_{n-2} &= u_n - 2hu'_n + 2h^2u''_n - \frac{4h^3}{3}u'''_n + O(h^4)\,. \end{aligned}$$

Since we want the error of our approximation to be $O(h^2)$, see (2.9), we keep terms up to $(Ch^2)h \sim h^3$ in the Taylor expansions above. Thus, multiplying (2.9) by h,

$$\begin{aligned} hu'_n + Ch^3 + O(h^4) &= w_{n-2}u_{n-2} + w_{n-1}u_{n-1} + w_n u_n\\ &= (w_{n-2} + w_{n-1} + w_n)u_n - h(2w_{n-2} + w_{n-1})u'_n\\ &\quad + \frac{h^2}{2}(4w_{n-2} + w_{n-1})u''_n - \frac{h^3}{6}(8w_{n-2} + w_{n-1})u'''_n + O(h^4)\,, \end{aligned} \tag{2.10}$$

in which we note that the unspecified error is, by construction, consistently $O(h^4)$ on both sides. Comparing coefficients of u_n, hu'_n and $\frac{h^2}{2}u''_n$ in (2.10) provides three equations for the weights,

$$w_{n-2} + w_{n-1} + w_n = 0, \quad 2w_{n-2} + w_{n-1} = -1 \quad \text{and} \quad 4w_{n-2} + w_{n-1} = 0,$$

with solution $w_{n-2} = \frac{1}{2}$, $w_{n-1} = -2$ and $w_n = \frac{3}{2}$. Comparing the coefficient of h^3 in (2.10) now yields $C = -\frac{1}{6}(8w_{n-2} + w_{n-1})u'''_n = -\frac{1}{3}u'''_n$, so that the BD'_2 formula for u'_n is

$$\frac{u_{n-2} - 4u_{n-1} + 3u_n}{2h} = u'_n - \frac{h^2}{3}u'''_n + O(h^3). \tag{2.11}$$

By a similar argument, we obtain the FD'_2 formula for u'_0 as

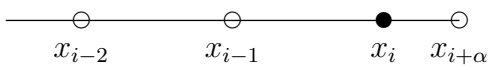
$$\frac{-3u_0 + 4u_1 - u_2}{2h} = u'_0 - \frac{h^2}{3}u'''_0 + O(h^3). \tag{2.12}$$

This approach can be used to obtain higher-order derivatives by postulating, e.g., the BD''_1

$$\frac{w_{n-2}u_{n-2} + w_{n-1}u_{n-1} + w_n u_n}{h^2} = u''_n + Ch + O(h^2)\,, \tag{2.13}$$

whose RH side should be compared with that in (2.9) and note the h^2 on the LH side.

□ **Example 2.2** *Irregularly spaced mesh points*
It occasionally occurs in practice that the mesh points are not regularly spaced, e.g. near a boundary. Suppose we seek an approximation to the derivative $u''_i = u''(x_i)$ in the following figure. $x_{i+\alpha}$ is the RH boundary value of x where, because $0 < \alpha < 1$, the mesh is irregular.



Since u_{i+1} does not exist, the symmetric CD''_2 formula (2.6) cannot be applied. Additionally, a mixture of FD and BD terms either side of x_i will be required. We therefore introduce the term *asymmetric difference* (AD) formula.

It is easy to show, using Taylor's theorem (0.4), that

$$\frac{u_{i+\alpha} - (1+\alpha)u_i + \alpha u_{i-1}}{\frac{1}{2}\alpha(\alpha+1)h^2} = u''_i - \frac{1-\alpha}{3}hu'''_i + \frac{1-\alpha+\alpha^2}{12}h^2u^{\mathrm{iv}}_i + O(h^3)\,. \tag{2.14}$$

Since $\alpha < 1$, the three values $u_{i+\alpha}$, u_i and u_{i-1} provide an AD''_1 for u''_i. An AD''_2 approximation therefore requires the introduction of the value u_{i-2}, whence the above method of weights gives

$$\frac{1}{h^2}\left[-\frac{1-\alpha}{2+\alpha}u_{i-2} + \frac{2(2-\alpha)}{1+\alpha}u_{i-1} - \frac{3-\alpha}{\alpha}u_i + \frac{6}{\alpha(1+\alpha)(2+\alpha)}u_{i+\alpha}\right] = u''_i + \frac{3\alpha-2}{12}h^2u^{\mathrm{iv}}_i + O(h^3),$$

from which we make the unexpected discovery that the unique value $\alpha \equiv \frac{2}{3}$ provides an AD''_3 approximation. □

2.1.2 Operator methods for 1-D finite-difference formulae

The use of Taylor expansions and weights is cumbersome, needing recalculation each time a different accuracy or order of differentiation is required. A more *systematic* and efficient

approach requires abstraction of ideas using the concept of *difference operators*. In abstract terms, an operator acting on an object transforms it into another object. In our case, the objects are the u_i, u_i', etc. The following table defines the operators that we shall encounter.

Name	Symbol	Effect
Forward-shift operator	E	$Eu_i = u_{i+1}$
Forward-difference operator	δ_+	$\delta_+ u_i = u_{i+1} - u_i$
Backward-difference operator	δ_-	$\delta_- u_i = u_i - u_{i-1}$
Central-difference operator I	δ_0	$\delta_0 u_i = u_{i+1/2} - u_{i-1/2}$
Central-difference operator II	δ	$\delta u_i = \frac{1}{2}(u_{i+1} - u_{i-1})$
Averaging operator	μ_0	$\mu_0 u_i = \frac{1}{2}(u_{i+1/2} + u_{i-1/2})$
Differential operator	D	$Du_i = u_i' \equiv \frac{du}{dx}$ at x_i
Identity operator	I	$Iu_i = u_i$

First note that the above operators are all *linear*; that is for any u_i, they satisfy

$$L(\alpha u_i + \beta \widetilde{u}_i) = \alpha L u_i + \beta L \widetilde{u}_i$$

for any constants α and β. We also have for any two of them, say L and $\widetilde{L}$,

$$(\alpha L + \beta \widetilde{L}) u_i = \alpha L u_i + \beta \widetilde{L} u_i \,.$$

We say that two operators *commute* if

$$L(\widetilde{L} u_i) = \widetilde{L}(L u_i) \,.$$

Two operators are *equivalent*, $L = \widetilde{L}$, if

$$L u_i = \widetilde{L} u_i \,.$$

For example,

$$(E - I)u_i = Eu_i - Iu_i = u_{i+1} - u_i = \delta_+ u_i,$$

via the definition of δ_+, we have the *operator equivalence* (OE)

$$\delta_+ = E - I. \tag{2.15}$$

It is obvious that a forward shift is cancelled by a backward shift, so that the backward-shift operator is the *inverse* of E; we therefore denote it by E^{-1}. Then

$$E^{-1} u_i = u_{i-1},$$

which leads to the following two OEs

$$\begin{aligned} \delta_- &= I - E^{-1} \,, \\ \delta_- &= E^{-1} \delta_+ \,. \end{aligned} \tag{2.16}$$

Repeated application of an operator L is denoted by L^2. It can be deduced that

$$\delta_+\delta_- = \delta_-\delta_+ = \delta_+ - \delta_- = \delta_0^2.$$

If m is either a positive or negative integer, we have

$$E^m u_i = u_{i+m}.$$

If m is not restricted to being an integer, we further have the following OEs:

$$\mu_0 = \frac{1}{2}(E^{1/2} + E^{-1/2})\,, \tag{2.17}$$

$$\delta_0 = E^{1/2} - E^{-1/2}\,, \tag{2.18}$$

$$\delta = \tfrac{1}{2}(E - E^{-1})\,, \tag{2.19}$$

in which the *half-shift* operator, $E^{1/2}$, can be thought of as the square root of E in the sense that

$$(E^{1/2})^2 u_i = E^{1/2}(E^{1/2}u_i) = E^{1/2}u_{i+1/2} = u_{i+1/2+1/2} = u_{i+1} = Eu_i.$$

Further OEs arising through repeated actions of operators are

$$\delta_+^2 = E^2 - 2E + I,$$

$$\delta_+^3 = E^3 - 3E^2 + 3E - I,$$

both of which are a direct consequence of (2.15).

Note that $E^{m+n} = E^m E^n = E^n E^m$ since the order in which shifts are performed is immaterial. So, E^m and E^n commute. Therefore, the operators δ_+, δ_-, δ_0, δ and μ_0, which can all be expressed in terms of E^m, commute with each other. On this note, observe that if $n = -m$ we have $E^0 = E^m E^{-m} = I$. So *any operator raised to the power of zero* is I, the identity.

2.1.3 Finite-difference formulae for first derivatives

Formal proof of the following is given in *A First Course in the Numerical Analysis of Differential Equations, A. Iserles, Chapter 8.* We remark that operator convergence is justified throughout because $h \to 0$ as the mesh is refined. Equation (2.1) (Taylor's theorem),

$$u_{i+1} = u_i + hu_i' + \frac{h^2}{2}u_i'' + \frac{h^3}{6}u_i''' + O(h^4)\,,$$

can be re-expressed in terms of operators as

$$Eu_i = Iu_i + hDu_i + \frac{h^2}{2}D^2u_i + \frac{h^3}{6}D^3u_i + O(h^4)\,.$$

Hence the OE statement is

$$E = I + hD + \frac{(hD)^2}{2} + \frac{(hD)^3}{6} + O(h^4)$$

which, by comparison with (0.2), gives

$$E = \exp(hD). \tag{2.20}$$

Now we can express differentiation in terms of shifting,

$$D = \frac{1}{h}\ln E, \tag{2.21}$$

which is the key point here. (2.21) will be used to derive an approximation to D (and, later, D^m). Taylor's theorem has now served its purpose and has been absorbed into a more general approach.

□ **Example 2.3** *Forward differences for D*
Rewriting the shifting operator E in (2.21) in terms of the forward-difference operator δ_+ using (2.15) and expanding using (0.3) yields

$$D = \frac{1}{h}\ln E = \frac{1}{h}\ln(I+\delta_+) = \frac{1}{h}\left\{\delta_+ - \frac{\delta_+^2}{2} + \frac{\delta_+^3}{3} - \frac{\delta_+^4}{4} + O(\delta_+^5)\right\}. \tag{2.22}$$

First note that, using operators, FD_1' (2.2) may be written in the form

$$\frac{\delta_+ u_i}{h} = Du_i + \frac{h}{2}D^2u_i + O(h^2)\,,$$

which implies the OE $\delta_+ = hD + O(h^2) = O(h)$; by raising both sides of this to the power of m, we have $\delta_+^m = O(h^m)$, so that the unspecified error $O(\delta_+^5)$ in the approximation (2.22) yields the *truncation error* $\frac{1}{h}O(h^5) = O(h^4)$. Hence by retaining the first m terms in the braces in (2.22), i.e. omitting $O(\delta_+^{m+1})$ and higher terms, (2.22) becomes a FD_m' formula. For example, the first term gives $D = \frac{1}{h}\delta_+ + O(h)$, i.e. $u_i' = \frac{1}{h}(u_{i+1}-u_i)+O(h)$, which is the FD_1' (2.2). The first two terms give $D = \frac{1}{h}\left\{\delta_+ - \frac{1}{2}\delta_+^2\right\} + O(h^2)$, i.e. $u_i' = \frac{1}{h}\left\{(u_{i+1}-u_i) - \frac{1}{2}(u_{i+2}-2u_{i+1}+u_i)\right\} + O(h^2)$, i.e. the FD_2'

$$u_i' = \frac{-3u_i + 4u_{i+1} - u_{i+2}}{2h}\,,$$

which was derived (for $i=0$) as (2.12) via considerably more effort. Note that the FD_∞' (2.22) is a series in single powers of δ_+, equivalently h, formally proving the point made immediately after (2.9). □

□ **Example 2.4** *Backward differences for D*
Similar to the previous example, we introduce (2.16) into (2.21) and expand using (0.3),

$$D = \frac{1}{h}\ln E = -\frac{1}{h}\ln(I-\delta_-) = \frac{1}{h}\left\{\delta_- + \frac{\delta_-^2}{2} + \frac{\delta_-^3}{3} + \frac{\delta_-^4}{4} + \cdots\right\}. \tag{2.23}$$

The first term gives the BD_1' (2.4) and the first two terms give the BD_2' (see (2.11))

$$u_i' = \frac{u_{i-2} - 4u_{i-1} + 3u_i}{2h}. \quad \square$$

□ **Example 2.5** *Type-I central differences for D*
Substituting (2.20) into (2.18) yields

$$\delta_0 = (\exp(hD)^{1/2} - \exp(hD)^{-1/2}) = (\exp(hD/2) - \exp(-hD/2)) = 2\sinh(hD/2)$$

$$\Rightarrow D = \frac{2}{h}\sinh^{-1}\left(\frac{\delta_0}{2}\right) = \frac{1}{h}\left\{\delta_0 - \frac{\delta_0^3}{24} + \frac{3\delta_0^5}{640} - \frac{5\delta_0^7}{7168} + \cdots\right\}. \tag{2.24}$$

(Maple or Mathematica can be used to obtained the Taylor expansion above.) First note that replacing h by $h/2$ in (2.5) gives

$$\frac{u_{i+1/2} - u_{i-1/2}}{h} = u'_i + \frac{h^2}{24}u'''_i + O(h^4)\,, \tag{2.25}$$

whose operator form implies $\delta_0 = hD + O(h^3) = O(h)$. Hence if m terms are retained in the braces in (2.24), last term therein will be $O(\delta_0^{2m+1})$, yielding a CD truncation error of $O(h^{2m})$. In contrast to the one-sided formulae in the previous two examples, the addition of each extra term here reduces the error by a factor of h^2.

As we have seen, the first term gives the CD'_2 (2.25) whereas the first two terms give the CD'_4

$$\frac{-u_{i+3/2} + 27u_{i+1/2} - 27u_{i-1/2} + u_{i-3/2}}{24h} = u'_i - \frac{3h^4}{640}u^{\mathrm{v}}_i + O(h^6). \tag{2.26}$$

A CD'_6 formula can be derived in similar way. □

The previous example shows that increasingly accurate approximations to u'_i can be obtained by increasing the number of terms evaluated at *half-integer* mesh points. But recall that it is the function values at the *integer* mesh points which are known to us, so that in order to apply the half-integer formulae (2.25) and (2.26) we would first have to interpolate the original data onto a *staggered* inter-node mesh. It is better to use the function values themselves, suggesting the use of δ rather than δ_0.

□ **Example 2.6** *Type-II central differences for D*
Substituting (2.20) into (2.19) gives

$$\delta = \frac{1}{2}\left(\exp(hD) - \exp(-hD)\right) = \sinh hD,$$

which inverts to give $hD = \sinh^{-1}\delta$, yielding

$$D = \frac{1}{h}\left\{\delta - \frac{\delta^3}{6} + \frac{3\delta^5}{40} - \frac{5\delta^7}{112} + \cdots\right\}, \tag{2.27}$$

which should be compared with (2.24). The first term gives the CD'_2 (2.5) but the first two terms give the CD'_4

$$\frac{-u_{i+3} + 27u_{i+1} - 27u_{i-1} + u_{i-3}}{48h} = u'_i - \frac{3h^4}{40}u^{\mathrm{v}}_i + O(h^6)\,, \tag{2.28}$$

which can be obtained simply by doubling h in (2.26). But this is still undesirable since it uses information at $x_{i\pm3}$ rather than at the nearer $x_{i\pm2}$: the $2h$-jump (from x_{i+1} to x_{i-1}) in the δ operator propagates into the CD formulae at all orders of accuracy. □

We infer that somehow the h-jump (from $x_{i+1/2}$ to $x_{i-1/2}$) of the δ_0 operator must be employed, but we have already rejected this on the basis of the half-integer location of the information points. We conclude that the use of δ_0 alone is insufficient, and the introduction of the as-yet-unused μ_0 is heralded.

Restriction to integer mesh points

It is straightforward to show that

$$\mu_0^2 = I + \frac{\delta_0^2}{4},$$

from which

$$I = \mu_0 \left\{ I + \frac{\delta_0^2}{4} \right\}^{-1/2} = \mu_0 \left\{ I - \frac{\delta_0^2}{8} + \frac{3\delta_0^4}{128} - \frac{5\delta_0^6}{1024} + \cdots \right\}. \tag{2.29}$$

The nifty trick is now to act on the LHS of (2.24) with the LHS of (2.29), i.e. the identity operator, and the RHS of (2.24) with the RHS of (2.29). There results

$$D = \frac{\mu_0}{h} \left\{ \delta_0 - \frac{\delta_0^3}{6} + \frac{\delta_0^5}{30} - \frac{\delta_0^7}{140} + \cdots \right\} = \frac{\delta}{h} \left\{ I - \frac{\delta_0^2}{6} + \frac{\delta_0^4}{30} - \frac{\delta_0^6}{140} + \cdots \right\}, \tag{2.30}$$

since $\mu_0\delta_0 = \delta$. We observe that (2.30) contains a single power of δ (one $2h$-jump) and even powers of δ_0 (pairs of $\frac{h}{2}$-jumps $= h$-jumps). Hence all the points arising through (2.30) lie at integer grid points as required. The first term recovers the CD_2' (2.5) whereas the first two terms yield the new CD_4'

$$\frac{-u_{i+2} + 8u_{i+1} - 8u_{i-1} + u_{i-2}}{12h} = u_i' - \frac{h^4}{30} u_i^{\mathrm{v}} + O(h^6)\,. \tag{2.31}$$

Notice how the error coefficient in (2.31) is 2.25 times smaller than in (2.28), which used $u_{i\pm3}$. One would rarely in practice use more than three terms in (2.27) and (2.30).

2.1.4 Finite-difference formulae for higher derivatives

Recalling that $\delta_\pm^m = O(h^m)$ in (2.22) and (2.23), we have

$$D = \frac{1}{h} \left\{ \delta_\pm \mp \frac{\delta_\pm^2}{2} + \frac{\delta_\pm^3}{3} \right\} \pm O(h^3)$$

which, when both sides are raised to the power of m, gives

$$D^m = \frac{1}{h^m} \left\{ \delta_\pm^m \mp \frac{m}{2}\delta_\pm^{m+1} + \frac{m(3m+5)}{24}\delta_\pm^{m+2} \right\} \pm O(h^3)\,, \tag{2.32}$$

wherein $\delta_\pm^m = O(h^m)$ has again been used. In computational terms, this means that the $m+3$ grid values $u_i,\ u_{i\pm1}, \ldots,\ u_{i\pm m\pm2}$ implied by (2.32) make it a $\mathrm{FD}_3^{(m)}$ or $\mathrm{BD}_3^{(m)}$ formula when the positive or negative signs are respectively taken.

To construct a $\mathrm{CD}^{(m)}$ we raise (2.24) to the mth power,

$$D^m = \frac{1}{h^m} \left\{ \delta_0 - \frac{\delta_0^3}{24} + \frac{3\delta_0^5}{640} - \frac{5\delta_0^7}{7168} + \cdots \right\}^m = \frac{\delta_0^m}{h^m} \left\{ I - \frac{m\delta_0^2}{24} + \frac{m(22+5m)\delta_0^4}{5760} + \cdots \right\} \tag{2.33}$$

which, for the reasons given in §2.1.3, uses function values at integer mesh points when m is even and half-integer mesh points when m is odd. On the other hand, acting I in (2.29) onto D^m in (2.33), we have

$$ID^m = \frac{\mu_0}{h^m}\left\{I + \frac{\delta_0^2}{4}\right\}^{-1/2}\left\{\delta_0 - \frac{\delta_0^3}{24} + \frac{3\delta_0^5}{640} - \frac{5\delta_0^7}{7168} + \cdots\right\}^m ,$$

i.e.

$$D^m = \frac{\mu_0\delta_0^m}{h^m}\left\{I - \frac{(m+3)\delta_0^2}{24} + \frac{(5m^2+52m+135)\delta_0^4}{5760} + \cdots\right\}, \tag{2.34}$$

which uses function values at integer mesh points when m is odd and half-integer mesh points when m is even.

□ **Example 2.7** *Central-difference formulae for second derivatives*
When $m = 2$, the first term of (2.34) gives the CD_2''

$$\frac{u_{i+3/2} - u_{i+1/2} - u_{i-1/2} + u_{i-3/2}}{2h^2} = u_i'' + \frac{5h^2}{24}u_i^{\mathrm{iv}} + O(h^4)\,,$$

and the first two terms give the (6-term) CD_4''

$$\frac{-5u_{i+5/2} + 39u_{i+3/2} - 34u_{i+1/2} - 34u_{i-1/2} + 39u_{i-3/2} - 5u_{i-5/2}}{48h^2} = u_i'' - \frac{259h^4}{5760}u_i^{\mathrm{vi}} + O(h^6)\,.$$

When $m = 2$, the first term of (2.33) recovers the CD_2'' (2.6), and the first two terms give a more efficient 5-term CD_4''

$$\frac{-u_{i+2} + 16u_{i+1} - 30u_i + 16u_{i-1} - u_{i-2}}{12h^2} = u_i'' - \frac{h^4}{90}u_i^{\mathrm{vi}} + O(h^6) \tag{2.35}$$

which uses function values at integer mesh points. □

2.1.5 A summary

Our derivation of finite-difference formulas relies on (i) the relation between D and E: $D = \frac{1}{h}\ln E$, (ii) relations between E and the finite-difference operators $\delta_+, \delta_-, \delta, \ldots$ etc. and the Taylor's theorem. The following summarise various formulas from previous sections.

For first derivative D,

1. Forward-difference
$$D = \frac{1}{h}\left\{\delta_+ - \frac{\delta_+^2}{2} + \frac{\delta_+^3}{3} - \frac{\delta_+^4}{4} + \cdots\right\}$$
2. Backward-difference
$$D = \frac{1}{h}\left\{\delta_- + \frac{\delta_-^2}{2} + \frac{\delta_-^3}{3} + \frac{\delta_-^4}{4} + \cdots\right\}$$
3. Central-difference
$$D = \frac{\delta}{h}\left\{I - \frac{\delta_0^2}{6} + \frac{\delta_0^4}{30} - \frac{\delta_0^6}{140} + \cdots\right\}$$

Forward- and backward-difference formulas for higher derivatives D^m can be obtained easily by raising the corresponding series above to the mth power. Central-difference for D^m is more tricky:

1. Central-difference for odd m

$$D^m = \frac{\mu_0\delta_0^m}{h^m}\left\{I - \frac{(m+3)\delta_0^2}{24} + \frac{(5m^2+52m+135)\delta_0^4}{5760} + \cdots\right\}$$

2. Central-difference for even m

$$D^m = \frac{\delta_0^m}{h^m}\left\{I - \frac{m\delta_0^2}{24} + \frac{m(22+5m)\delta_0^4}{5760} + \cdots\right\}$$

2.1.6 Implicit finite-difference formulae

The formulae derived in §§2.1.3 and 2.1.4 are all *explicit*: in every case, the derivative $u_i^{(\mathrm{m})} = d^m u(x_i)/dx^m$ at the single mesh point x_i is given directly in terms of neighbouring mesh points. *Implicit* formulae are defined as those where derivatives at *more than one* mesh point appear simultaneously. Their advantage comes from the high order of accuracy generated when derivatives at different mesh points are related to each other. The price to paid is that the interwoven values of the derivatives must be solved *en bloc* via the solution of a set of simultaneous algebraic equations.

□ **Example 2.8** *Reduction of truncation error via implicit formulae*
Setting $m = 2$ in (2.33) we have

$$D^2 = \frac{1}{h^2}\left\{\delta_0^2 - \frac{\delta_0^4}{12} + \frac{\delta_0^6}{90} + \cdots\right\}$$

which, recalling that $\delta_0^m = O(h^m)$, gives both

$$\text{(a)}\;\; D^2 = \frac{\delta_0^2}{h^2}\left\{I - \frac{\delta_0^2}{12}\right\} + O(h^4) \quad \underline{\text{and}} \quad \text{(b)}\;\; D^2 = \frac{\delta_0^2}{h^2} + O(h^2). \tag{2.36}$$

Recalling that all operators commute with each other, (2.36)(a) gives

$$\left\{I - \frac{\delta_0^2}{12}\right\}^{-1} D^2 = \frac{\delta_0^2}{h^2} + O(h^4), \tag{2.37}$$

wherein the inverse can be expanded then truncated as

$$\left\{I - \frac{\delta_0^2}{12}\right\}^{-1} = I + \frac{\delta_0^2}{12} + \frac{\delta_0^4}{144} + \frac{\delta_0^6}{1728} + \cdots = I + \frac{\delta_0^2}{12} + O(h^4),$$

so that (2.37) gives

$$\left\{I + \frac{\delta_0^2}{12}\right\} D^2 = \frac{\delta_0^2}{h^2} + O(h^4). \tag{2.38}$$

Thus, from (2.36)(b), δ_0^2/h^2 is a second-order approximation to D^2 whereas, from (2.38), it is a fourth-order approximation to $(I + \delta_0^2/12)D^2$, which must now be interpreted. Recall that $\delta_0 = E^{1/2} - E^{-1/2}$, so that $\delta_0^2 = E - 2I + E^{-1}$ and the LHS of (2.38) is

$$\left(\frac{1}{12}E + \frac{5}{6}I + \frac{1}{12}E^{-1}\right)D^2 = \frac{1}{12}\left(ED^2 + 10D^2 + E^{-1}D^2\right).$$

Since $ED^2u_i = Eu_i'' = u_{i+1}''$ and $E^{-1}D^2u_i = E^{-1}u_i'' = u_{i-1}''$, (2.38) yields

$$\frac{u_{i+1}'' + 10u_i'' + u_{i-1}''}{12} = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + O(h^4), \tag{2.39}$$

which is an *implicit, three-point* fourth-order formula, to be compared with the explicit, five-point fourth-order formula (2.35). Moreover, the error of the implicit three-point formula (2.39), can be calculated more precisely as

$$\frac{h^4}{240}u_i^{\mathrm{vi}} + O(h^6),$$

which is approximately 2.7 times smaller than the error in the explicit five-point formula (2.35). Finally, note that if the weights on the LHS of (2.39) are perturbed from $\{\frac{1}{12}, \frac{5}{6}, \frac{1}{12}\}$ to $\{0, 1, 0\}$, we recover the explicit three-point formula (2.6), which is accurate only to second order; $\{\frac{1}{12}, \frac{5}{6}, \frac{1}{12}\}$ is the *unique* set of weights giving fourth-order accuracy. □

Assembly of the implicit equations (2.39)

The above example illustrates the twofold advantages of the implicit approach—the reduction in *both* the truncation error and the number of mesh points required. The disadvantage is that our fourth-order formula for the second-derivative has not actually provided us with the value of u_i'' for any value of i, so that we have to solve (2.39) via the following *tridiagonal* system of simultaneous algebraic equations, in which the RHS vector comprises known function values.

$$\begin{pmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \cdots & 10 & 1 & 0 & 0 & 0 & \cdots \\ \cdots & 1 & 10 & 1 & 0 & 0 & \cdots \\ \cdots & 0 & 1 & 10 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & 10 & 1 & \cdots \\ \cdots & 0 & 0 & 0 & 1 & 10 & \cdots \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ \vdots \\ u_{i-1}'' \\ u_i'' \\ u_{i+1}'' \\ \vdots \\ \vdots \end{pmatrix} = \frac{12}{h^2} \begin{pmatrix} \vdots \\ \vdots \\ u_i - 2u_{i-1} + u_{i-2} \\ u_{i+1} - 2u_i + u_{i-1} \\ u_{i+2} - 2u_{i+1} + u_i \\ \vdots \\ \vdots \end{pmatrix} \tag{2.40}$$

Note that the tridiagonal pattern may "break" at the boundary points x_0 and x_n, where one-sided formulae are used. The effect of this in the above matrix is that the first row will not necessarily commence with $(10,\ 1,\ \cdots)$ and the last row will not necessarily terminate with $(\cdots,\ 1,\ 10)$. Note also that the *bandwidth* of the system is 3: in general, the smaller the bandwidth, the easier the system is to solve. In particular, tridiagonal systems can be solved very efficiently indeed, as we shall see later. We infer that implicit formulae should be constructed so as to minimise

the bandwidth, and we see that this is achieved by relating derivatives at mesh points which are as close as possible.

All results so far obtained are for a function u of one variable x. We now turn to their extension into higher dimensions, i.e. for use in the solution of PDEs, rather than ODEs.

2.2 Finite-difference formulae in 2-D

Now $u = u(x, y)$ and the mesh points lie on a rectangular mesh with increments h and k in the x- and y-directions respectively, so that $x_{i\pm m} = x_i \pm mh$ and $y_{j\pm n} = y_j \pm nk$. The value of $u(x_i, y_j)$ is denoted by u_{ij} or $u_{i,j}$. All of the operators in §2.1.2 can be applied to either direction, acting separately on the i and j subscripts.

□ **Example 2.9** *Finite-difference formulae for partial derivatives u_x and u_{yy}*
Single $\pm$ shifts in the x-direction are represented by

$$E_x u_{ij} = u_{i+1,j} \quad \text{and} \quad E_x^{-1} u_{ij} = u_{i-1,j},$$

and double $\pm$ shifts in the y-direction by

$$E_y^2 u_{ij} = u_{i,j+2} \quad \text{and} \quad E_y^{-2} u_{ij} = u_{i,j-2},$$

so that FD_1^x, the first-order forward difference for u_x (in which j is unaltered), is

$$(D_x u)_{ij} \simeq \frac{1}{h}\delta_{+,x} u_{ij} \Rightarrow \left(\frac{\partial u}{\partial x}\right)_{ij} = \frac{u_{i+1,j} - u_{i,j}}{h} + O(h)\,,$$

and CD_2^{yy}, the second-order central difference for u_{yy} (in which i is unaltered),

$$(D_y^2 u)_{ij} \simeq \frac{1}{k^2}\delta_{0,y}^2 u_{ij} \Rightarrow \left(\frac{\partial^2 u}{\partial y^2}\right)_{ij} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{k^2} + O(k^2).$$

The above results were obtained using (0.5) with $u(x, y) = u_{ij}$. Any required multi-dimensional derivative can be constructed by applying the newly defined operators in this way. □

The most common derivative which is ubiquitous in the study of elliptic, parabolic and hyperbolic PDEs is the Laplace/Laplacian operator, which in 2-D cartesian coordinates is

$$\Delta = \nabla^2 \equiv \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\,. \tag{2.41}$$

□ **Example 2.10** *The Laplace operator*
The CD_2 form of the *Laplacian* operator Δ is, from (2.41),

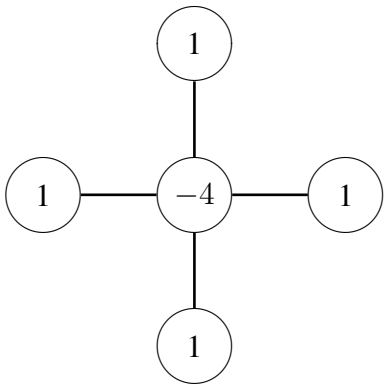
$$\Delta_0^{(5,+)} = \frac{\delta_{0,x}^2}{h^2} + \frac{\delta_{0,y}^2}{k^2}, \tag{2.42}$$

which, on a regular 2-D mesh with $h = k$, (hereafter, a *square* mesh), gives the so-called *five-point* CD_2 approximation to the Laplacian,

$$\frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2} = (\Delta u)_{ij} + \frac{h^2}{12}\left(u_{xxxx} + u_{yyyy}\right)_{ij} + O(h^4), \tag{2.43}$$

which is possibly *the* most-used formula in the numerical analysis of 2-D physically-motivated problems. Since both i and j jump by ± 1, (2.43) leads to a system whose *bandwith* is 3. Equation (2.43) can be pictured in terms of the *molecule*, or *stencil*

$$\Delta_0^{(5,+)}u_{ij} \equiv \frac{1}{h^2}$$



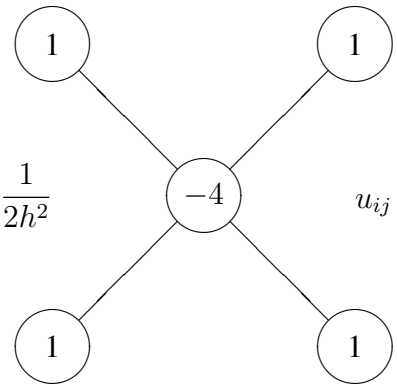
$$u_{ij} = \Delta u_{ij} + \frac{h^2}{12}\left(u_{xxxx} + u_{yyyy}\right)_{ij} + O(h^4).$$

As finite-difference formulae become more complicated, the molecular representation is the only practical one. Other CD_2 molecules can be obtained using a mixture of operators, as per (2.30). The most common mixed approximation of the Laplacian operator is

$$\Delta_0^{(5,\times)} \equiv \frac{(\mu_{0,y}\delta_{0,x})^2}{h^2} + \frac{(\mu_{0,x}\delta_{0,y})^2}{k^2}, \tag{2.44}$$

which, on a square mesh, leads to a different five-point CD_2 molecule,

$$\Delta_0^{(5,\times)}u_{ij} \equiv \frac{1}{2h^2}$$



$$u_{ij} = \Delta u_{ij} + \frac{h^2}{12}\left(u_{xxxx} + 6u_{xxyy} + u_{yyyy}\right)_{ij} + O(h^4).$$

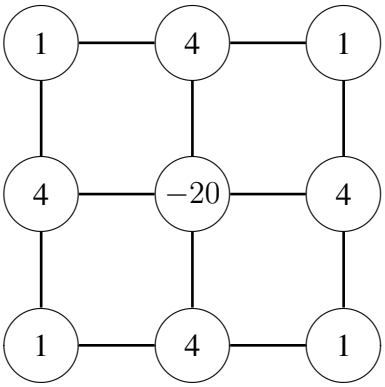
However, $\Delta_0^{(5,\times)}$ is not to be recommended for practical purposes, since the odd-numbered points are detached from the even-numbered ones, so that two sets of *decoupled* equations result. Although $\Delta_0^{(5,\times)}$ is not recommended *per se*, it can be linked with $\Delta_0^{(5,+)}$ to reduce the truncation error provided that u satisfies certain conditions. □

□ **Example 2.11** *Combining molecules to reduce the truncation error*
We require the preliminary observation that $\Delta u = u_{xx} + u_{yy} \Rightarrow \Delta(\Delta u) \equiv \Delta^2 u = u_{xxxx} + 2u_{xxyy} + u_{yyyy}$. Δ^2 is known as the *biharmonic* operator. When $h = k$, we can combine $\Delta_0^{(5,+)}u_{ij}$ and $\Delta_0^{(5,\times)}u_{ij}$ to make the h^2 term proportional to $\Delta^2 u_{ij}$,

$$\frac{2}{3}\Delta_0^{(5,+)}u_{ij} + \frac{1}{3}\Delta_0^{(5,\times)}u_{ij} \equiv \Delta_0^{(9)} = \Delta u_{ij} + \frac{h^2}{12}\Delta^2 u_{ij} + O(h^4). \tag{2.45}$$

Algebraic operations such as addition and multiplication can be formally applied to the molecules, thus leading to the *nine-point* square-mesh molecule

$$\Delta_0^{(9)} u_{ij} \equiv \frac{1}{6h^2}$$



$$u_{ij} = \Delta u_{ij} + \frac{h^2}{12}\left(\Delta^2 u\right)_{ij} + O(h^4).$$

We have seen the reason for not using $\Delta_0^{(5,\times)}$ on practical grounds, but why should $\Delta_0^{(9)}$ be preferred to the *simpler* $\Delta_0^{(5,+)}$; both are second-order, having $O(h^2)$ errors! The answer lies in the actual coefficient of the h^2 term. In the many physical problems for which u is either harmonic ($\Delta u = 0$) or biharmonic ($\Delta^2 u = 0$), the coefficient of the $O(h^2)$ truncation error of the nine-point operator vanishes so that, for that class of functions, the nine-point formula automatically improves—with no extra effort—to a *fourth*-order scheme. An extremely impressive by-product of this example, known as *Mehrstellenverfahren*, is given in the next section. □

2.2.1 Higher-order approximations to the Laplacian

Equation (2.42), being a second-order central-difference approximation to a second derivative, can be compared to (2.33) with $m = 2$,

$$D^2 = \frac{1}{h^2}\left\{\delta_0^2 - \frac{\delta_0^4}{12} + \frac{\delta_0^6}{90} + \cdots\right\},$$

which is valid at 1-D integer mesh points. On the square mesh, we may similarly derive

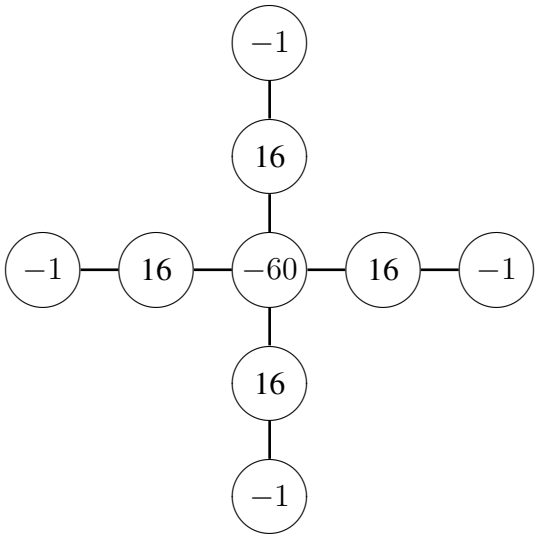
$$\Delta = \frac{1}{h^2}\left\{\delta_{0,x}^2 + \delta_{0,y}^2 - \frac{1}{12}\left(\delta_{0,x}^4 + \delta_{0,y}^4\right)\right\} + O(h^4), \tag{2.46}$$

the first two terms of which is identifiable with (2.42).

□ **Example 2.12** *A general fourth-order molecule for the Laplacian*
The first and third terms in (2.46) are essentially the first two terms in (2.33) for D^2 applied to the x-direction. Similarly for the second and fourth terms in (2.46) for the y-direction. We see that the first two terms in (2.33) give us CD_4'' in (2.35). Here, (2.46) gives us the nine-point

fourth-order formula whose square-mesh molecule is

$$\Delta_0^{(9,+)}u_{ij} \equiv \frac{1}{12h^2}$$



$$u_{ij} = \Delta u_{ij} + O(h^4)$$

and should be compared to to its 1-D counterpart (2.35). Whilst this molecule is genuinely fourth-order for *all* functions $u(x, y)$, it too is unsatisfactory from a practical point of view since it leads to a system of equations with bandwidth 5; a *pentadiagonal* system, which is much more expensive to solve than a tridiagonal one. □

2.2.2 "Mehrstellenverfahren" for the Poisson equation

Quite literally, "more-places process", i.e a combination of molecules. Although attributed to Collatz (1966), it is curious that this extremely effective procedure is rarely presented in the modern literature. The idea is to obtain a more accurate solution for the Poisson equation

$$\Delta u(x, y) = f(x, y) \qquad \text{or} \qquad \nabla^2 u(x, y) = f(x, y) \tag{2.47}$$

than is at first sight possible from the nine-point formula of Example 2.11, which we observed was second-order in general and fourth-order if u was biharmonic. By adjusting that formula in a very subtle way, fourth-order accuracy can be obtained for *any* u!

Now

$$\Delta u = f \Rightarrow \frac{h^2}{12}\Delta^2 u = \frac{h^2}{12}\Delta f \Rightarrow \Delta u + \frac{h^2}{12}\Delta^2 u = f + \frac{h^2}{12}\Delta f = \left(I + \frac{h^2}{12}\Delta\right) f \equiv \bar{f}, \text{ say.}$$

By the nine-point molecule in Example 2.11,

$$\Delta u + \frac{h^2}{12}\Delta^2 u = \Delta_0^{(9)} u + O(h^4),$$

and therefore

$$\Delta_0^{(9)} u_{ij} = \bar{f}_{ij} + O(h^4). \tag{2.48}$$

But the nine-point formula in Example 2.11 applied to the Poisson equation (2.47) can be stated as

$$\Delta_0^{(9)} u_{ij} = f_{ij} + O(h^2). \tag{2.49}$$

In other words, the near-trivial act of multiplying the *known* RHS of (2.47), f, by the operator $I + \frac{h^2}{12}\Delta$ improves the scheme from second order to fourth order *for any function* u: this is a most impressive result. Finally, we note from (2.42) and (2.43) that

$$h^2\Delta = h^2\Delta_0^{(5,+)} + O(h^4) = \delta_{0,x}^2 + \delta_{0,y}^2 + O(h^4),$$

and so (2.48) gives

$$\Delta_0^{(9)} u_{ij} = \left(I + \frac{1}{12}\left(\delta_{0,x}^2 + \delta_{0,y}^2\right) \right) f_{ij} + O(h^4)$$

or, in molecular form,

$$\frac{1}{6h^2}\begin{array}{ccc} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{array} u_{ij} = \frac{1}{12}\begin{array}{ccc} & 1 & \\ 1 & 8 & 1 \\ & 1 & \end{array} f_{ij} + O(h^4)$$

We reiterate that the new RHS is extremely cheap to compute since f is a known function of x and y. Hence the gain in accuracy is remarkable for the effort involved. This technique should be used in preference to any other if the finite-difference method is selected. □

□ **Example 2.13** *An application of Mehrstellenverfahren*
In this example, the Poisson equation $\Delta u = \sin x \exp(y^2)(1 + 4y^2)$ is solved via various finite-difference approximations.

The exact solution, $u(x, y) = \sin x \exp(y^2)$, is used to derive boundary conditions on a $2h \times 2h$ square centred at the point $(x, y) = (1/2, 1/2)$. The initial value of h is $1/2$, and h is repeatedly halved (M times) so that the (initially-unit) square shrinks around $(1/2, 1/2)$, at which lies the only unknown value of u, say u_h. The exact value of $u(1/2, 1/2)$ is denoted by u_e.

Care must be taken here to accommodate the atypical situation that the *domain* halves as the mesh is halved: normally, it would remain unchanged while the *mesh within it* was halved. This, however, would yield an increasingly large set of unknowns, which we wish to defer until §§3 and 4. The domain scalings $x \to x/2$ and $y \to y/2$ between successive meshes gives (in an obvious notation) $\Delta u_h \to 4\Delta u_{h/2}$, i.e. the *difference* equations on the finer mesh—including their error—carry a scaling factor of 1/4 over those on the coarser mesh. Hence the error on the smaller mesh must be amplified by a factor of 4 if a comparison is to be effected with the error on the coarser mesh. We recall that the 5-point (+), 5-point (×) and 9-point molecules all give an $O(h^2)$ error in approximating the PDE $\Delta u = f$, and the 9-point-modified formula gives an $O(h^4)$ error. So, defining the *(numerical) error attenuation ratio* by

$$\alpha(h) \equiv \frac{\text{error on mesh } h}{4 \times \text{error on mesh } h/2} = \frac{u_h - u_e}{4(u_{h/2} - u_e)},$$

we expect $\alpha(h) \to 2^2$ for the first three schemes and $\alpha(h) \to 2^4$ for the modified scheme. In the following tables, we denote the *theoretical* limiting value of the attenuation ration by

$$\alpha_0 = \lim_{h\to 0} \alpha(h).$$

Working to 10-digit accuracy, we obtain the following attenuation ratios.

Scheme	$\alpha(1/2)$	$\alpha(1/4)$	$\alpha(1/8)$	$\alpha(1/16)$	$\alpha(1/32)$	α_0
5-pt. $+$	4.342142730	4.080821295	4.019930735	4.004085480	3.957711443	4
5-pt. $\times$	3.967184823	3.998139058	4.000351360	3.997318693	3.979674798	4
9-pt.	4.293311293	4.069838315	4.017056085	4.007839128	3.922459893	4
9-pt. mod.	18.42840734	16.57285924	16.39156627	8.300000000	-0.4166666668	16

The non-convergence (as $h \to 0$) to α_0 is a result of the decreasing differences in both the numerator and denominator of $\alpha(h)$ being swamped by rounding errors; evidently, 10-digit accuracy is insufficient. Performing all calculations to 20-digit accuracy, we obtain the following ratios, which are *presented* to 10 digits for reasons of space.

Scheme	$\alpha(1/2)$	$\alpha(1/4)$	$\alpha(1/8)$	$\alpha(1/16)$	$\alpha(1/32)$	α_0
5-pt. $+$	4.342142548	4.080814465	4.019923734	4.004963667	4.001239841	4
5-pt. $\times$	3.967186298	3.998130405	3.999885786	3.999992902	3.999999557	4
9-pt.	4.293311893	4.069853329	4.017255838	4.004301136	4.001074484	4
9-pt. mod.	18.42838975	16.56658767	16.13926389	16.03466924	16.00865839	16

Now, as $h \to 0$, we observe convergence to α_0. Thus it can be seen that h cannot be made arbitrarily small, the round-off error of the machine imposing a restriction. Note that the factor of 4 should *not* be included in the definition of $\alpha(h)$ when the domain size is maintained and the mesh only is halved, as in Question 6 on Examples Sheet 5. □

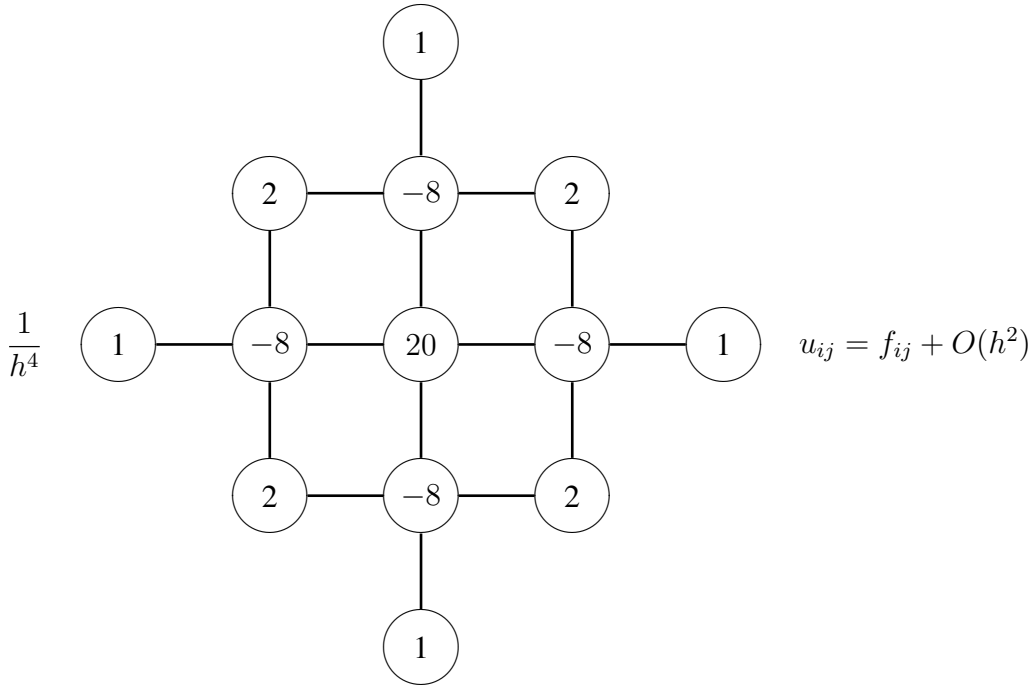
2.2.3 Higher-order multidimensional derivatives

These can be approximated using either operator or Mehrstellenverfahren techniques. In practical terms, the most common such PDEs usually involve the *biharmonic* operator, Δ^2, in the form

$$\Delta^2 u = f.$$

The case $f = 0$ arises physically when u is the 2-D *streamfunction* of viscous fluid mechanics; when u is a constant u is the deflection shown by a clamped rectangular plate.

□ **Example 2.14** *Molecule for the biharmonic operator*
The most common molecular representation of $\Delta^2 u = f$ is the second-order



Note that the bandwidth is 5 even though the scheme is only second-order. The higher order of differentiation requires the increased bandwidth if an explicit formula is used. □

Concluding remarks

The power of the theory of finite-difference operators lies in its generality. With it, one can construct one-sided, central and asymmetric approximations to D^m, for any m and to any order: the application will determine which formulae are required and/or possible. A notable omission from the theory is the consideration of finite-difference formulae on *irregular* meshes, e.g. near curved boundaries, where special formulae are required. In such cases, it is arguable that other approaches, e.g. the *finite-element* method or *boundary-integral* method, are best employed.

3 Numerical Linear Algebra

3.1 Fundamentals

3.1.1 Matrix and vector norms; spectral radius

We often need some measure of the "size" of a vector or a matrix. This is provided by the *norm* of the vector or matrix. In the following, we introduce different kinds of norm.

The p-norm of the n-vector $\boldsymbol{x} = (x_1,\ x_2, \ldots,\ x_n)^T$ is defined to be

$$\|\boldsymbol{x}\|_p \equiv \left\{ \sum_{i=1}^{n} x_i^p \right\}^{\frac{1}{p}} .$$

From this definition, we have the following norms of a vector:

$$\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i| \ , \qquad \text{sum of moduli;}$$

$$\|\boldsymbol{x}\|_2 = \left\{ \sum_{i=1}^{n} x_i^2 \right\}^{\frac{1}{2}} , \quad \text{Euclidean;}$$

$$\|\boldsymbol{x}\|_\infty = \max_{1\le i\le n} |x_i| \ , \qquad \text{maximum.}$$

The Euclidean norm (2-norm) has the geometric interpretation of an n-dimensional Pythagoras' theorem, and the maximum norm is simply the largest modulus of any element.

The real $(n\times n)$ matrix A (with elements a_{ij}) admits several *norm* definitions, the most common of which are:

$$\|A\|_F = \left\{ \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2 \right\}^{\frac{1}{2}} , \quad \text{Frobenius;}$$

$$\|A\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}| \ , \qquad \text{maximum column sum;}$$

$$\|A\|_2 = \max |\text{eigenvalue of } A^TA|^{\frac{1}{2}}, \quad \text{spectral;}$$

$$\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}| \ , \qquad \text{maximum row sum.}$$

It transpires that $\|A\|_2$ is the minimum of all the norms; it is the tightest measure of the "size" of A. It is appropriate to introduce the *spectral radius* of A,

$$\rho(A) = \max_i |\lambda_i|,$$

where λ_i, $i = 1(1)n$, are the eigenvalues of A. Thus, the spectral radius of A is simply the largest of the moduli of the eigenvalues of A. Hence the spectral-norm definition can be formally phrased as

$$\|A\|_2 = \left\{\rho\left(A^TA\right)\right\}^{\frac{1}{2}} .$$

Three important properties (of any norm) we shall exploit are

$$\|A\boldsymbol{x}\| \leq \|A\| \, \|\boldsymbol{x}\| \,, \quad \|AB\| \leq \|A\| \, \|B\| \,, \quad \text{and} \quad \|A^m\| \leq \|A\|^m \,, \tag{3.1}$$

where m is a positive integer.

3.1.2 Diagonal dominance and eigenvalue theorems

The matrix A is said to be *strictly (row) diagonally dominant* if

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \quad i = 1(1)n, \tag{3.2}$$

i.e. if the modulus of the diagonal term is greater than or equal to the sum of the moduli of the off-diagonal terms in any row. In some texts, the weaker condition "$\geq$" is used to replace "$>$", and the word "*strictly*" is removed from the description. Also "*row*" can be changed to "*column*" in the description by changing a_{ij} to a_{ji} in (3.2). We shall return to diagonal dominance in §3.2.3.

Closely related to the above is *Gerschgorin's (first) theorem*, which asserts that the spectral radius of A cannot exceed the largest sum of the moduli of the elements in any column or any row. This is equivalent to $\rho(A) \leq \|A\|_1$ and $\rho(A) \leq \|A\|_\infty$. In fact, an important result of linear algebra is that, for *any* norm,

$$\rho(A) \leq \|A\| \,. \tag{3.3}$$

Gerschgorin's (first) theorem leads directly to *Gerschgorin's circle theorem*, or *Brauer's theorem*. Define

$$P_i \equiv \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \quad i = 1(1)n, \tag{3.4}$$

(cf. (3.2)) to be the sum of the moduli of the off-diagonal elements in the i^{th} row of A. Then there is always an (i, j) pair, with $1 \leq i, j \leq n$, such that $|\lambda_j - a_{ii}| \leq P_i$, i.e. every eigenvalue of A lies inside or on the boundary of at least one of the circles of radius P_i centred on a_{ii} on the complex plane. The formula for P_i in (3.4) yields *row discs*: by changing a_{ij} to a_{ji}, we can similarly construct *column discs*.

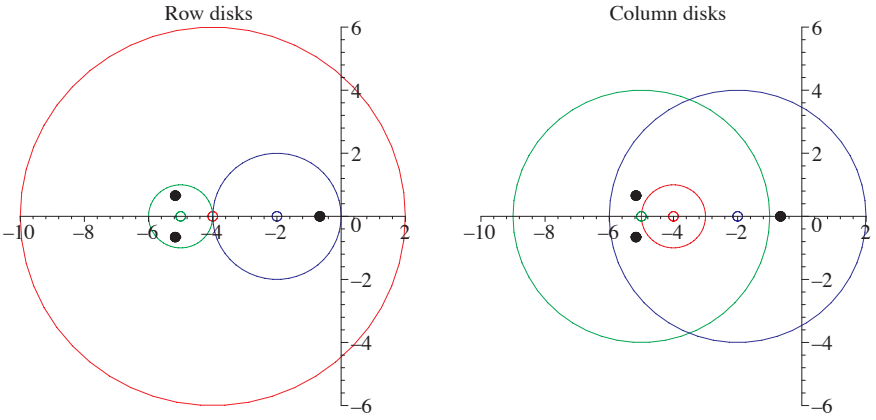
□ **Example 3.1** *Illustration of Brauer's theorem*
If

$$A = \begin{pmatrix} -4 & 3 & 3 \\ 1 & -2 & 1 \\ 0 & 1 & -5 \end{pmatrix},$$

we find $\|A\|_\infty = 10 > \|A\|_1 = 9 > \|A\|_F \simeq 8.124 > \|A\|_2 \simeq 6.536$, i.e. $\|A\|_2$ is indeed the smallest, as predicted.

The eigenvalues of A are $\lambda_1 \approx -0.6631$, $\lambda_2 \approx -5.168 + 0.6579i$ and $\lambda_3 \approx -5.168 - 0.6579i$.

The figures below show the *Gerschgorin row discs* (left) and *column discs* (right) of A.



In the left figure, the row discs are:

- from row 1 of A, centre $a_{11} = -4$ and radius $|a_{12}| + |a_{13}| = 6$;
- from row 2 of A, centre $a_{22} = -2$ and radius $|a_{21}| + |a_{23}| = 2$;
- from row 3 of A, centre $a_{33} = -5$ and radius $|a_{31}| + |a_{32}| = 1$.

In the right figure, the column discs are:

- from column 1 of A, centre $a_{11} = -4$ and radius $|a_{21}| + |a_{31}| = 1$;
- from column 2 of A, centre $a_{22} = -2$ and radius $|a_{12}| + |a_{32}| = 4$;
- from column 3 of A, centre $a_{33} = -5$ and radius $|a_{13}| + |a_{23}| = 4$.

In both figures, the eigenvalues (black dots, •) of A are contained within the union of the circles, in accordance with Brauer's theorem. Similar figures for any 3×3 matrix can be generated using the `Python` script `3474_3.1.py`. □

3.1.3 Sparse systems of equations

We consider methods for solving *sparse* systems of n simultaneous algebraic equations, such as (2.40), written in the form

$$A\boldsymbol{x} = \boldsymbol{b}. \tag{3.5}$$

Here A is a known $n\times n$ square sparse matrix. A sparse matrix is one in which most coefficients are zero. $\boldsymbol{x}$ and $\boldsymbol{b}$ are respectively unknown and known $n \times 1$ vectors. Direct solution methods, such as Gaussian elimination, are methods that gives the exact answer (ignoring rounding errors) in a finite number of steps. They are generally not suited to solving (3.5) when n is large. In the special case that A is *tridiagonal*, as in (2.40), the direct method of §3.2.1 is used.

In general, *iterative* methods are used in which trial components of the unknown $\boldsymbol{x}$ are initially chosen as $\boldsymbol{x}^{(0)}$, say, which is improved by a sequence of iterative corrections $\{\boldsymbol{x}^{(1)},\ \boldsymbol{x}^{(2)},\ \boldsymbol{x}^{(3)},\ \ldots\}$. The iteration *converges* when there is some measure of agreement (quantified in §3.2.3) between successive *iterates*, i.e. when the modulus of the *correction* drops below some tolerance. The type of correction technique distinguishes the method, and the most common ones will be discussed in §3.2.2. Iterative methods may be further divided into *stationary* and *gradient* methods, of which only the former are considered here.

We begin by splitting the $n \times n$ matrix A into its "DLU decomposition"

$$A = D - L - U, \tag{3.6}$$

in which

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix} \quad \text{and} \quad D = \begin{pmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ 0 & 0 & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn} \end{pmatrix}$$

are the matrix A and its *diagonal* component D (N.B. all $a_{ii} \neq 0$), and

$$L = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 \\ -a_{21} & 0 & \cdots & 0 & 0 \\ -a_{31} & -a_{32} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & -a_{n,n-1} & 0 \end{pmatrix} \quad \text{and} \quad U = \begin{pmatrix} 0 & -a_{12} & -a_{13} & \cdots & -a_{1n} \\ 0 & 0 & -a_{23} & \cdots & -a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & -a_{n-1,n} \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

are the *lower-* and *upper-triangular* components of A. Assuming that all $a_{ii} \neq 0$ (achieved by changing the order of the equations in (3.5), if necesarry), we divide the i^{th} row of the n simultaneous equations (3.5) by a_{ii}. Then the non-zero elements in D are all 1, whence D is the $n \times n$ *identity* matrix, I. We shall henceforth assume that this scaling has taken place, so that (3.6) is replaced by its "ILU decomposition"

$$A = I - L - U. \tag{3.7}$$

3.2 Solution of sparse systems

3.2.1 Direct method: LU-factorisation for tridiagonal systems

In the event that A is tridiagonal,

$$A = \begin{pmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & b_n \end{pmatrix},$$

we write it as the product of a lower-triangular matrix L^* and upper-triangular matrix U^* (N.B. the notation reflects that these have no connection with L and U in (3.7)), where

$$L^* = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ l_2 & 1 & 0 & 0 & \cdots & 0 \\ 0 & l_3 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \quad \text{and} \quad U^* = \begin{pmatrix} v_1 & w_1 & 0 & 0 & \cdots & 0 \\ 0 & v_2 & w_2 & 0 & \cdots & 0 \\ 0 & 0 & v_3 & w_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & v_n \end{pmatrix}.$$

so that

$$A = L^*U^* = \begin{pmatrix} v_1 & w_1 & 0 & 0 & \cdots & 0 \\ v_1 l_2 & v_2 + l_2 w_1 & w_2 & 0 & \cdots & 0 \\ 0 & v_2 l_3 & v_3 + l_3 w_2 & w_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & v_n \end{pmatrix}.$$

Comparing elements in A and L^*U^* gives $v_1 = b_1$ and $w_1 = c_1$ from row 1, then (after a little algebra),

$$l_i = \frac{a_i}{v_{i-1}}, \quad v_i = b_i - l_i w_{i-1} \quad \text{and} \quad w_i = c_i, \quad i = 2(1)n.$$

This determines all of the entries in L^* and U^*. Let x_i and β_i be the i^{th} components of $\boldsymbol{x}$ and $\boldsymbol{b}$ respectively, both of which vectors are now used in a two-step solution process.

We first solve $L^*\boldsymbol{y} = \boldsymbol{b}$ for $\boldsymbol{y}$. Comparing elements row by row now gives $y_1 = \beta_1$ from row 1, then

$$y_i = \beta_i - l_i y_{i-1}, \quad i = 2(1)n;$$

$\boldsymbol{y}$ is found by *forward substitution*.

Next, we solve $U^*\boldsymbol{x} = \boldsymbol{y}$ for the required vector $\boldsymbol{x}$, since we have

$$U^*\boldsymbol{x} = \boldsymbol{y} \quad \Rightarrow \quad L^*U^*\boldsymbol{x} = L^*\boldsymbol{y} \quad \Rightarrow \quad A\boldsymbol{x} = \boldsymbol{b}.$$

This time, the n^{th} row yields $x_n = y_n/v_n$, then

$$x_i = \frac{y_i - w_i x_{i+1}}{v_i}, \quad i = n-1(-1)1;$$

$\boldsymbol{x}$ is found by *backward substitution*. The method is fast and efficient for tridiagonal systems, requiring the storage of only the $4n-2$ values of a_2 to a_n, b_1 to b_n, c_1 to c_{n-1} and β_1 to β_n. The number of arithmetic operations is proportional to n, whereas it would be n^2 in Gaussian elimination.

Note finally that matrices arising through FD approximations often have

$$a_2 = \cdots = a_n, \quad b_1 = \cdots = b_n, \quad \text{and} \quad c_1 = \cdots = c_{n-1},$$

in which case A is said to be a *Toeplitz* matrix (c.f. (2.40)). Note that Toeplitz matrices do not necessarily have to be tridiagonal.

3.2.2 Iterative stationary methods: Jacobi, Gauss-Seidel and SOR

These are used in the event that A is not tridiagonal. Equation (3.7) gives

$$A\boldsymbol{x} = \boldsymbol{b} \quad \Rightarrow \quad (I - L - U)\boldsymbol{x} = \boldsymbol{b} \quad \Rightarrow \quad \boldsymbol{x} = (L + U)\boldsymbol{x} + \boldsymbol{b}, \tag{3.8}$$

so that the simplest iterative scheme possible has

$$\boldsymbol{x}^{(m+1)} = (L + U)\boldsymbol{x}^{(m)} + \boldsymbol{b}, \tag{3.9}$$

in which $\boldsymbol{x}^{(m+1)} = \left(x_1^{(m+1)},\, x_2^{(m+1)}, \ldots,\, x_n^{(m+1)}\right)^T$ and $\boldsymbol{x}^{(m)} = \left(x_1^{(m)},\, x_2^{(m)}, \ldots,\, x_n^{(m)}\right)^T$. Equation (3.9) is called the *Jacobi* (J) scheme, which is a *simultaneous correction* method since the i^{th} row of (3.9) gives $x_i^{(m+1)}$ in terms of *all* of $x_1^{(m)}$ to $x_n^{(m)}$ from the previous iterate.

The Jacobi scheme is easy but inefficient: after $x_1^{(m+1)}$ is found from row 1 in (3.9), this "fresh" value should be used to update $x_1^{(m)}$ in the *remaining* rows 2 to n. Similarly, the fresh value of $x_2^{(m+1)}$ should similarly be used to replace $x_2^{(m)}$ in rows 3 to n, etc. We therefore have

$$\boldsymbol{x}^{(m+1)} = L\boldsymbol{x}^{(m+1)} + U\boldsymbol{x}^{(m)} + \boldsymbol{b}\,, \tag{3.10}$$

and hence

$$\boldsymbol{x}^{(m+1)} = (I-L)^{-1}U\boldsymbol{x}^{(m)} + (I-L)^{-1}\boldsymbol{b}. \tag{3.11}$$

Equation (3.11) is the *Gauss-Seidel* (GS) method, which we expect to converge faster than (3.9) since it is a *successive-correction* method that uses the most up-to-date information available.

Note that adding and subtracting $\boldsymbol{x}^{(m)}$ to the RHS of (3.10) yields GS in the form

$$\underbrace{\boldsymbol{x}^{(m+1)}}_{\text{new}} = \underbrace{\boldsymbol{x}^{(m)}}_{\text{old}} + \underbrace{L\boldsymbol{x}^{(m+1)} + (U-I)\boldsymbol{x}^{(m)} + \boldsymbol{b}}_{\text{correction}}\,. \tag{3.12}$$

We postulate that the iteration may converge more quickly if the correction term in (3.12) is scaled by a so-called *relaxation parameter* ω:

$$\boldsymbol{x}^{(m+1)} = \boldsymbol{x}^{(m)} + \omega\left[L\boldsymbol{x}^{(m+1)} + (U-I)\boldsymbol{x}^{(m)} + \boldsymbol{b}\right]. \tag{3.13}$$

Equation (3.13) can also be written as

$$\boldsymbol{x}^{(m+1)} = (1-\omega)\boldsymbol{x}^{(m)} + \omega\left[L\boldsymbol{x}^{(m+1)} + U\boldsymbol{x}^{(m)} + \boldsymbol{b}\right] \tag{3.14}$$

where the right-hand side is a mixture of $\boldsymbol{x}^{(m)}$ and GS of (3.10) with the proportion of each determined by ω. Comparison of (3.10) and (3.14) reveals that $\omega \equiv 1$ for GS. On the basis that the correction in (3.12) does not change sign (which, for elliptic problems, it does not), we expect to improve the convergence rate of GS by taking $\omega > 1$. Then (3.13) gives

$$(I-\omega L)\boldsymbol{x}^{(m+1)} = \left[(1-\omega)I + \omega U\right]\boldsymbol{x}^{(m)} + \omega\boldsymbol{b},$$

so that

$$\boldsymbol{x}^{(m+1)} = (I-\omega L)^{-1}\left[(1-\omega)I + \omega U\right]\boldsymbol{x}^{(m)} + \omega(I-\omega L)^{-1}\boldsymbol{b}. \tag{3.15}$$

Equation (3.15) defines the *successive over-relaxation* (SOR) method, the term "over" referring to the fact that ω is greater than 1. In certain cases, we can theoretically determine the value of ω that yields the fastest rate of convergence of $\boldsymbol{x}^{(m)}$ to $\boldsymbol{x}$.

3.2.3 Convergence of iterative schemes

By convergence of an iterative scheme we mean that, given a pre-specified *tolerance*, ϵ, say, there exists an iteration number k such that (most commonly)

$$\left\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\right\|_\infty \le \epsilon,$$

although any other norm could be used. It is to be noted that the Jacobi (3.9), Gauss-Seidel (3.11) and SOR (3.15) iterative schemes are all of the form

$$\boldsymbol{x}^{(m+1)} = H\boldsymbol{x}^{(m)} + \boldsymbol{v}, \tag{3.16}$$

where H is referred to as the *iteration matrix*, with

$$H_J = L + U, \quad H_{GS} = (I - L)^{-1}U; \qquad \text{and} \qquad H_{SOR} = (I - \omega L)^{-1}\left[(1-\omega)I + \omega U\right].$$

Suppose now that $\boldsymbol{x}$ is the *exact* solution of (3.5) (or (3.16)), i.e.

$$\boldsymbol{x} = H\boldsymbol{x} + \boldsymbol{v}.$$

If the error in the m^{th} iterate is

$$\boldsymbol{e}^{(m)} = \boldsymbol{x}^{(m)} - \boldsymbol{x},$$

then, using (3.16), we find

$$\boldsymbol{e}^{(m+1)} = H\boldsymbol{e}^{(m)} = H^2\boldsymbol{e}^{(m-1)} = \cdots = H^{m+1}\boldsymbol{e}^{(0)}, \quad \text{i.e.} \quad \boldsymbol{e}^{(m)} = H^m\boldsymbol{e}^{(0)}. \tag{3.17}$$

It can be shown (see Problem Set 5) that the necessary condition which ensures $\boldsymbol{e}^{(m)} \to \boldsymbol{0}$ is simply

$$\rho(H) < 1.$$

On the other hand, using the properties (3.1) of matrix norms,

$$\left\|\boldsymbol{e}^{(m)}\right\| = \left\|H^m\boldsymbol{e}^{(0)}\right\| \le \|H^m\|\left\|\boldsymbol{e}^{(0)}\right\| \le \|H\|^m\left\|\boldsymbol{e}^{(0)}\right\|.$$

Since $\boldsymbol{e}^{(0)}$ is an arbitrary initial error, a sufficient condition for convergence is therefore

$$\|H\| < 1.$$

Then, since we recall $\rho(H) \le \|H\|$ for all norms and for all H, we may have: (i) $\rho(H) \le \|H\| < 1$ (convergence); (ii) $\rho(H) \le 1 \le \|H\|$ (convergence or divergence), or; (iii) $1 < \rho(H) \le \|H\|$ (divergence).

Two useful results are: (i) if the matrix A is diagonally dominant, both Jacobi and Gauss-Seidel converge, and; (ii) $\|H_J\|_\infty < 1 \Longleftrightarrow A$ is strictly row-diagonally dominant.

□ **Example 3.2** *Convergence affected by re-ordering equations*
If

$$A = \begin{pmatrix} 3 & 1 & 2 \\ -1 & 3 & -2 \\ -2 & 2 & 3 \end{pmatrix},$$

then (after scaling by the diagonal terms), we find

$$\rho(H_J) = 1 \quad \text{and} \quad \|H_J\|_2 = 1,$$

and

$$\rho(H_{GS}) \simeq 0.6083 \quad \text{and} \quad \|H_{GS}\|_2 \simeq 1.1234,$$

thus Gauss-Seidel converges (the necessary condition $\rho(H_{GS}) < 1$ satisfied) and Jacobi marginally diverges. Note that if the 3's on the diagonal are replaced by $3+\epsilon$ we have $\rho(H_J) = 3/(3+\epsilon) <$

1, i.e. Jacobi converges for *any* $\epsilon > 0$. If the second and third rows, and then the second and third columns, are swapped, we have

$$A = \begin{pmatrix} 3 & 2 & 1 \\ -2 & 3 & 2 \\ -1 & -2 & 3 \end{pmatrix}.$$

After scaling, once again we find

$$\rho(H_J) = 1 \quad \text{and} \quad \|H_J\|_2 = 1,$$

but now

$$\rho(H_{GS}) \simeq 1.2652 \quad \text{and} \quad \|H_{GS}\|_2 \simeq 1.4715,$$

so that Jacobi's convergence is unchanged whereas Gauss-Seidel diverges rapidly. This example demonstrates that Jacobi is not automatically to be considered as being inferior to Gauss-Seidel. □

3.2.4 The optimum relaxation parameter for SOR

Recall that the SOR iteration matrix in (3.15) depends upon the *(over-)relaxation* (or *acceleration*) parameter ω according to

$$H_\omega \equiv H_{SOR} = (I - \omega L)^{-1}\left[(1-\omega)I + \omega U\right], \tag{3.18}$$

and hence so does the spectral radius $\rho(H_\omega)$ of H_ω. In many applications, one find that $\left\|\boldsymbol{e}^{(m+1)}\right\|_\infty / \left\|\boldsymbol{e}^{(m)}\right\|_\infty \lesssim \rho(H_\omega)$. Therefore, we seek the *optimum relaxation parameter* ω^* that minimises $\rho(H_\omega)$, i.e. that maximises the rate of SOR convergence. When the matrix A has a general structure, ω^* can be found only numerically. However, when A has certain properties, ω^* (and other related results) can be found theoretically.

Specifically, when the (diagonally scaled) matrix A can be *partitioned* (possibly after permutation of rows and/or columns) into the form (*cf.* (3.7))

$$A = \left(\begin{array}{c|c} I_n & -U \\ \hline -L & I_m \end{array}\right), \tag{3.19}$$

in which $I_n \in \mathbb{R}^n \times \mathbb{R}^n$ and $I_m \in \mathbb{R}^m \times \mathbb{R}^m$ are identity matrices, and $U \in \mathbb{R}^n \times \mathbb{R}^m$ and $L \in \mathbb{R}^m \times \mathbb{R}^n$, it is said to be *2-cyclic*. Note that U and L in (3.19) are rectangular, whereas they are triangular in (3.7). Also, (3.19) reflects the fact that, in the system $A\boldsymbol{x} = \boldsymbol{b}$, the unknown vector $\boldsymbol{x}$ can be split into two subvectors $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ in such a way that the equations link every component of $\boldsymbol{x}_a$ to only those in $\boldsymbol{x}_b$, and vice-versa.

When A is 2-cyclic, and when the eigenvalues of the Jacobi iteration matrix $H_J = L+U$ satisfy certain properties (see §3.2.5), it can be proved that $\rho(H_\omega) < 1$ provided that $\omega \in (0, 2)$; it can also be shown that $\omega^* \in (1, 2)$. Moreover, the value of ω^* is given by

$$\omega^* = \frac{2}{1 + \sqrt{1 - \rho(H_J)^2}} \tag{3.20}$$

(N.B. spectral radius is that of Jacobi matrix), whence

$$\rho(H_{\omega^*}) = \omega^* - 1. \tag{3.21}$$

Also, Gauss-Seidel converges (or diverges) precisely twice as fast as Jacobi, i.e.

$$\rho(H_{GS}) = \left\{\rho(H_J)\right\}^2. \tag{3.22}$$

For 2-cyclic matrices with marginally-convergent Jacobi (i.e $\rho(H_J)$ just below 1), the *relative* iteration efficiency of SOR over Jacobi and Gauss-Seidel can be impressive. Specifically, If $0 < \varepsilon \ll 1$, then (3.21)–(3.22) yield

$$\rho(H_J) = 1 - \varepsilon\,, \qquad \rho(H_{GS}) \simeq 1 - 2\varepsilon\,, \qquad \rho(H_{\omega^*}) \simeq 1 - 2\sqrt{2\varepsilon}\,, \tag{3.23}$$

so that the *relative efficiency* of the *optimal* SOR over Jacobi is

$$\frac{\ln\rho(H_{\omega^*})}{\ln\rho(H_J)} \simeq \frac{\ln(1-2\sqrt{2\varepsilon})}{\ln(1-\varepsilon)} \simeq 2\sqrt{\frac{2}{\varepsilon}} + 4 + O(\sqrt{\varepsilon}),$$

i.e. approximately 13, 32, 93 and 287 when $\rho(H_J) = 0.9$, 0.99, 0.999 and 0.9999 respectively.

□ **Example 3.3** *Theoretically determined* ω^*
Consider the 2-cyclic, symmetric matrix

$$A = \left(\begin{array}{cc|cc} -4 & 0 & 1 & 1 \\ 0 & -4 & 1 & 1 \\ \hline 1 & 1 & -4 & 0 \\ 1 & 1 & 0 & -4 \end{array}\right).$$

Using the `Python` script `3474_3.3.py`, we find, after scaling the diagonal to 1,

$$L = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1/4 & 1/4 & 0 & 0 \\ 1/4 & 1/4 & 0 & 0 \end{pmatrix}, \qquad U = \begin{pmatrix} 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

and hence,

$$H_J = \begin{pmatrix} 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 1/4 & 1/4 \\ 1/4 & 1/4 & 0 & 0 \\ 1/4 & 1/4 & 0 & 0 \end{pmatrix}, \qquad H_{GS} = \begin{pmatrix} 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 1/8 & 1/8 \\ 0 & 0 & 1/8 & 1/8 \end{pmatrix},$$

from which we obtain

$$\rho(H_J) = 0.5 \quad \text{and} \quad \rho(H_{GS}) = 0.25.$$

We calculate ω^* using (3.20) and construct H_{ω^*} to obtain

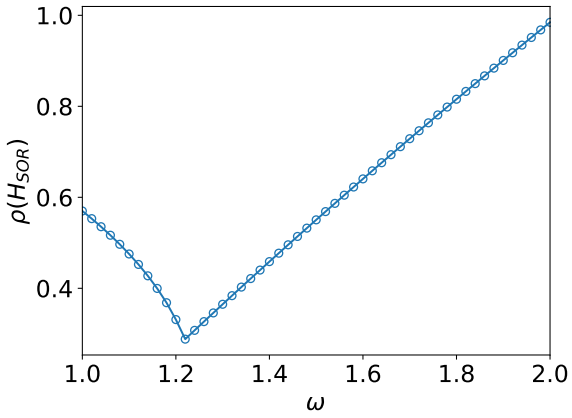
$$\omega^* = 1.071796770 \quad \text{and} \quad \rho(H_{\omega^*}) = 0.071796770.$$

The above results are in complete accordance with (3.21)–(3.22). □

□ **Example 3.4** *Experiments with numerically determined* ω^*
We now consider the following 4×4 Toeplitz matrix

$$A = \left(\begin{array}{cc|cc} -4 & 1 & 1 & 1 \\ 1 & -4 & 1 & 1 \\ \hline 1 & 1 & -4 & 1 \\ 1 & 1 & 1 & -4 \end{array}\right)$$

which is *not* 2-cyclic and therefore ω^* cannot be obtained theoretically. Instead, using the `Python` script `3474_3.4.py`, we construct H_{SOR} for different $\omega \in [1, 2]$ to obtain the following plot of $\rho(H_{SOR})$ as a function of ω.



An increasingly refined search reveals $\omega^* \simeq 1.216218$, whence $\rho(H_{SOR}) \simeq 0.293707$. Note in particular that $\rho(H_{\omega^*}) \neq \omega^* - 1$ since (3.21) does not hold for non-2-cyclic matrices.

Also, for this system, $\rho(H_J) = 0.75$ and $\rho(H_{GS}) \simeq 0.569945$, (i.e. (3.20) does not hold) so that

$$\frac{\ln \rho(H_{SOR})}{\ln \rho(H_{GS})} \simeq 2.179 \quad \text{and} \quad \frac{\ln \rho(H_{SOR})}{\ln \rho(H_J)} \simeq 4.259.$$

Thus the (optimised) SOR should converge about twice as quickly as Gauss-Seidel, which in turn should converge about twice as quickly as Jacobi. Using a FORTRAN 77 implementation of the three schemes, one finds via experiments that fastest convergence for SOR occurs at $\omega \simeq 1.23$, close to the value of ω^*. With a convergence tolerance of $\left\|\boldsymbol{e}^{(k)}\right\|_\infty < 10^{-11}$, the number of iterations are found to be 20 (SOR), 43 (GS) and 79 (J), in broad agreement with the theoretical predictions; the agreement is not perfect since the actual number of iterations is dependent upon the initial vector $\boldsymbol{x}^{(0)}$. □

3.2.5 The optimum SOR parameter for 2-cyclic matrices

Let λ be an eigenvalue of $H_J = L + U$ with eigenvector $\boldsymbol{q}$. Following (3.19), $H_J\boldsymbol{q} = \lambda\boldsymbol{q}$ is written in the partitioned form

$$\left(\begin{array}{c|c} 0 & U \\ \hline L & 0 \end{array}\right)\left(\begin{array}{c} \boldsymbol{q}_a \\ \hline \boldsymbol{q}_b \end{array}\right) = \lambda\left(\begin{array}{c} \boldsymbol{q}_a \\ \hline \boldsymbol{q}_b \end{array}\right), \tag{3.24}$$

from which

$$\left(\begin{array}{c|c} 0 & U \\ \hline L & 0 \end{array}\right)\left(\begin{array}{c} \boldsymbol{q}_a \\ \hline -\boldsymbol{q}_b \end{array}\right) = -\lambda\left(\begin{array}{c} \boldsymbol{q}_a \\ \hline -\boldsymbol{q}_b \end{array}\right),$$

i.e. the (non-zero) eigenvalues of H_J occur in pairs of the form $\pm\lambda$.

Now let μ be an eigenvalue of H_ω with eigenvector $\boldsymbol{p}$, so that $H_\omega\boldsymbol{p} = \mu\boldsymbol{p}$. Then (3.18) gives

$$(I-\omega L)^{-1}\left\{(1-\omega)I+\omega U\right\}\boldsymbol{p} = \mu\boldsymbol{p} \quad\Rightarrow\quad (\mu L+U)\,\boldsymbol{p} = \frac{\mu+\omega-1}{\omega}\boldsymbol{p},$$

which can be written in partitioned form as

$$\left(\begin{array}{c|c} 0 & U \\ \hline \mu L & 0 \end{array}\right)\left(\begin{array}{c} \boldsymbol{p}_a \\ \hline \boldsymbol{p}_b \end{array}\right) = \frac{\mu+\omega-1}{\omega}\left(\begin{array}{c} \boldsymbol{p}_a \\ \hline \boldsymbol{p}_b \end{array}\right),$$

from which

$$\left(\begin{array}{c|c} 0 & U \\ \hline L & 0 \end{array}\right)\left(\begin{array}{c} \sqrt{\mu}\boldsymbol{p}_a \\ \hline \boldsymbol{p}_b \end{array}\right) = \frac{\mu+\omega-1}{\omega\sqrt{\mu}}\left(\begin{array}{c} \sqrt{\mu}\boldsymbol{p}_a \\ \hline \boldsymbol{p}_b \end{array}\right). \tag{3.25}$$

Via (3.24) and (3.25), the eigenvalues of the SOR and Jacobi matrices are related by

$$\lambda = \frac{\mu+\omega-1}{\omega\sqrt{\mu}},$$

which is a quadratic equation in $\sqrt{\mu}$ with solutions

$$\sqrt{\mu} = \frac{\omega\lambda}{2} \pm \sqrt{\frac{\omega^2\lambda^2}{4}-\omega+1} \quad\Rightarrow\quad \mu = \frac{1}{4}\left[\omega\lambda \pm \sqrt{\omega^2\lambda^2-4\omega+4}\right]^2 \tag{3.26}$$

in which μ may be complex.

We now assume that $\lambda \in \mathbb{R}$, which is guaranteed when L is the conjugate transpose of U, $L_{ij} = \overline{U_{ji}}$, as it is in the common event that A is real and symmetric. [Counterexamples can easily be found that violate (3.20) and (3.21) when $\lambda \in \mathbb{C}$.] Since we are comparing the performances of convergent SOR and convergent Jacobi, we also require $\rho(H_\omega) < 1$ and $\rho(H_J) < 1$, respectively giving $|\mu| < 1$ and $\lambda \in (-1, 1)$. We also note that $\omega \in \mathbb{R}$.

We first consider the case $\omega \equiv 1$, when SOR is equivalent to Gauss-Seidel. Then, irrespective of $\lambda \in \mathbb{R}$, (3.26) gives (ignoring the zero eigenvalues after the first step)

$$\mu = \left\{\frac{\lambda}{2} \pm \sqrt{\frac{\lambda^2}{4}}\right\}^2 = 0 \text{ or } \lambda^2 \;\Rightarrow\; |\mu| = |\lambda^2| = |\lambda|^2 \;\Rightarrow\; \rho(H_{GS}) = \left\{\rho(H_J)\right\}^2,$$

so that Gauss-Seidel converges precisely twice as quickly as Jacobi, as asserted in (3.22).

Our goal now is to calculate $\rho(H_\omega)$ as a function of ω. We consider only $\omega \in (0, 2)$ and shall come back to justify this later. The first step is to recognise that μ switches from being real to complex when the discriminant in (3.26) becomes negative. Specifically, for a given λ, μ is real when $0 < \omega \le \tilde{\omega}(\lambda)$ where $\tilde{\omega}(\lambda)$ satisfies

$$\lambda^2\tilde{\omega}^2 - 4\tilde{\omega} + 4 = 0$$

$$\Longrightarrow \qquad \tilde{\omega}(\lambda) = \frac{2}{1+\sqrt{1-\lambda^2}} < 2 \tag{3.27}$$

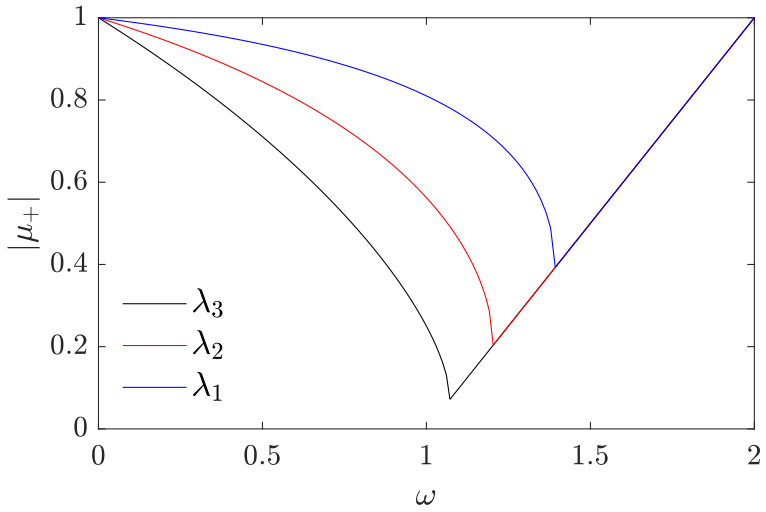
(the other root that leads to $\tilde{\omega} \geq 2$ is rejected). As our concern is the spectral radius, we are only interested in the larger root in (3.26)

$$\mu_+ = \frac{1}{4}\left[\omega\lambda + \sqrt{\omega^2\lambda^2 - 4\omega + 4}\right]^2 , \tag{3.28}$$

which decreases strictly monotonically with $\omega \in (0, \tilde{\omega})$ for $|\lambda| < 1$ (prove this by considering the function $f(\omega) = \omega\lambda + \sqrt{\omega^2\lambda^2 - 4\omega + 4}$ and show that $f'(\omega) < 0$). Next we consider $\tilde{\omega}(\lambda) < \omega < 2$ where μ is complex. From (3.26),

$$\begin{aligned} |\mu| &= \frac{1}{4}\left|\omega\lambda \pm i\sqrt{-\omega^2\lambda^2 + 4\omega - 4}\right|^2 \\ \implies |\mu_+| &= |\mu_-| = \omega - 1. \end{aligned} \tag{3.29}$$

Thus $|\mu_+|$ is independent of λ and increases strictly monotonically with $\omega \in (\tilde{\omega}, 2)$. (3.28) and (3.29) together gives $|\mu_+|$ in the full range of $\omega \in (0, 2)$ at a fixed λ. The following figure plots $|\mu_+|$ as a function of ω for three different λ where we have assumed $\lambda_1 > \lambda_2 > \lambda_3 > 0$. Since eigenvalues of H_J occur in pair $\pm\lambda$ and from (3.26), both $\pm\lambda$ lead to the same $|\mu_+|$, we focus only on $\lambda > 0$.



We note two important features in the above figure, both follow from (3.28). First, at a fixed $\omega \in (0, \tilde{\omega})$, $\mu_+ \in \mathbb{R}$ increases monotonically with $\lambda \in [0, 1)$ and second, $\mu_+(\omega)$ in (3.28) ends at $\omega = \tilde{\omega}$ onto the straight line $|\mu_+| = \omega - 1$. As a consequence, $|\mu_+(\omega; \lambda_1)| \geq |\mu_+(\omega; \lambda_2)|$ if $\lambda_1 > \lambda_2$ for $\omega \in (0, 2)$. It thus follows from the definition of spectral radius that

$$\rho(H_\omega) = \begin{cases} \dfrac{1}{4}\left[\omega\rho(H_J) + \sqrt{\omega^2\rho(H_J)^2 - 4\omega + 4}\right]^2, & \omega \in (0, \omega^*) \\ \omega - 1, & \omega \in [\omega^*, 2), \end{cases} \tag{3.30}$$

with the minimum of $\rho(H_\omega)$ occurs at $\omega^* \equiv \tilde{\omega}(\rho(H_J))$. Hence, we arrive at the results (3.20) and (3.21) that for a 2-cyclic matrix,

$$\omega^* = \frac{2}{1 + \sqrt{1 - \rho(H_J)^2}} ,$$

$$\rho(H_{\omega^*}) = \omega^* - 1 .$$

Since $0 < \rho(H_J) < 1$ for convergent Jocabi scheme, it immediately follows from the above expression for ω^*, or (3.20), that $\omega^* \in (1, 2)$.

We now explain our reason for focusing on the range $\omega \in (0, 2)$. For $\omega \leq \tilde{\omega}$, μ_+ in (3.28) increases monotonically with $\lambda \in (-1, 1)$, the requirement $\mu_+ < 1$, and hence $\rho(H_\omega) < 1$ for all $|\lambda| < 1$ implies

$$\mu_+(\omega; \lambda = 1) = \frac{1}{4}\left[\omega + \sqrt{\omega^2 - 4\omega + 4}\right] = (\omega - 1)^2 < 1$$

$$\implies \omega(\omega - 2) < 0 \implies 0 < \omega < 2.$$

For $\omega > \tilde{\omega}$, $|\mu_+| = \omega - 1 < 1$ gives $\omega < 2$ again. Therefore, given $\rho(H_J) < 1$, the necessary condition of $\rho(H_\omega) < 1$ for the SOR scheme to converge implies $\omega \in (0, 2)$.