MA20222: Numerical Analysis

Department of Mathematical Sciences
University of Bath

Contents

1   What is numerical analysis?

Much of today's science and engineering depends on large-scale calculations performed with computers. These calculations find solutions or approximate solutions to mathematical models and enable scientists and engineers to predict behaviours of interest. A prime example is the weather: PDE models are used to predict the weather based on recent observations. The accuracy of the predictions depend on many things, including the accuracy of the numerical solution to the PDEs. In this unit, we explore the theoretical basis for these numerical methods, especially their reliability and efficiency. The name given to this subject is *Numerical Analysis.*

Numerical analysis is half theory and half practice. We want to prove that algorithms work with rigorous mathematical analysis and to implement them. An essential part of the course will be for *you* to implement and use the methods yourself.

1.1   Examples

Quadratic-equation formula

The following formula for the two roots of a quadratic equation is well known:

$$x_\pm = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

gives the solutions x of

$$a\,x^2 + b\,x + c = 0, \qquad \text{for given } a, b, c.$$

For a pure mathematician, this could be the end of the story: Python provides built-in routines for evaluating such expressions and hence we can input a, b, c and find the roots.

It is not always so simple: computers work with finite-precision arithmetic and only store finitely many different numbers. Any number that is too long (π, $\sqrt{2}$,... written in base 10) or too big (above 10^{308} or too small 10^{-323} on my machine) causes problems. We focus on numbers that are too long: in Python and many other computing environments, real numbers are stored to 16-significant figures. That is, any number with more than 16 digits (excluding the exponent) in base 10 is rounded (by chopping or choosing the nearest) to 16 digits. In Python, typing

```
import numpy as np
np.pi,
```

we see 3.141592653589793. Of course, computers work in base 2 and the principle there is similar.

At first, this appears like a minor irritation as the error caused is so small relative to the size of π. However, when performing long computations in finite-precision arithmetic, the effects can accumulate to cause a catastrophic loss of accuracy. For example, consider computing $x_\pm$ in the case $b = 10^6$, $a = 10^{-3}$ and $c = 10^{-3}$:

$$x_\pm = \frac{-10^6 \pm \sqrt{10^{12} - 4 \times 10^{-6}}}{2 \times 10^{-3}}.$$

Working to 16-significant figures, the square root evaluates to $\pm 10^6$ because $10^{12} - 4 \times 10^{-6} = 10^{12}$ to 16 s.f. (s.f. denotes significant figures). Hence, the computed values of the roots are $x_\pm^c = -10^9, 0$. In fact, the exact answers are $x_\pm = -10^9, -10^{-9}$ (to 16 s.f.). There are no correct digits in the x_+^c. Indeed,

$$\text{the absolute error in } x_+ \text{ is } \left| x_+ - x_+^c \right| \approx 10^{-9},$$

but

$$\text{the relative error in } x_+ \text{ is } \frac{\left| x_+ - x_+^c \right|}{|x_+|} \approx 1.$$

Using the relative error, we scale the error relative to what we are trying to compute and find the error in computing x_+ is unacceptably large.

How can the quadratic-equation formula be evaluated accurately in this case? No, the answer is not "Get a bigger computer or use more digits in your calculations". Numerical analysis provides ways of improving algorithms so they are less sensitive to the effects of rounding error, without the need to change computing environment. Indeed, now that single-processor computing speed is no longer increasing dramatically, it is becoming more important to exploit good algorithms. Sixteen figures is enough to represent the answer and the algorithm can be adjusted to avoid the problematic cancellation $-10^6 + \sqrt{10^{12} + \text{neglected}}$ and find the correct answer in Python. In this case, we note that one of the roots $x_\pm$ is evaluated accurately and the second root can be computed accurately by exploiting the identity $x_+ x_- = c/a$ for the product of the roots.

Linear equations

To convince you that the previous example is not overly contrived, consider the linear system of equations:
$$\begin{bmatrix} \epsilon & 1 \\ 0 & 1 \end{bmatrix} \boldsymbol{x} = \boldsymbol{b}, \qquad \boldsymbol{b} := \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

for a known small number $0 < \epsilon \ll 1$. We are interested in determining $\boldsymbol{x} \in \mathbb{R}^2$ and it is easy to show that
$$\boldsymbol{x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Imagine that there has been rounding error and the vector $\boldsymbol{b}$ is actually stored as $[1 + \delta, 1]^\mathsf{T}$ (the T denotes transpose):
$$\begin{bmatrix} \epsilon & 1 \\ 0 & 1 \end{bmatrix} \boldsymbol{x} = \boldsymbol{b}, \qquad \boldsymbol{b} := \begin{bmatrix} 1 + \delta \\ 1 \end{bmatrix}. \tag{1.1}$$

Again solving the linear system, we find

$$\boldsymbol{x} = \begin{bmatrix} \delta/\epsilon \\ 1 \end{bmatrix}.$$

In the case that $0 < \epsilon \ll \delta$ (ϵ is much smaller than δ), there is a large change to the solution $\boldsymbol{x}$. This system and its solution $\boldsymbol{x}$ is highly sensitive to small changes in input data (as modelled by δ). This is a simple example of an *ill-conditioned* system

of equations and these arise widely in mathematical modelling and are particularly hard to solve accurately using numerical methods. The perturbation represented by δ always occurs in numerical simulations due to rounding error.

In contrast to the quadratic-equation example, the ill-conditioning here is fundamental to the underlying equations and is not a consequence of the method of solution. Numerical analysis can help identify numerically stable algorithms that are less susceptible to the effects of rounding error, but, if there is an instability in the underlying model, even a good algorithm will produce wrong answers. This is why well posedness (existence, uniqueness, and continuity of solutions with respect to parameters) is studied in modules on differential equations.
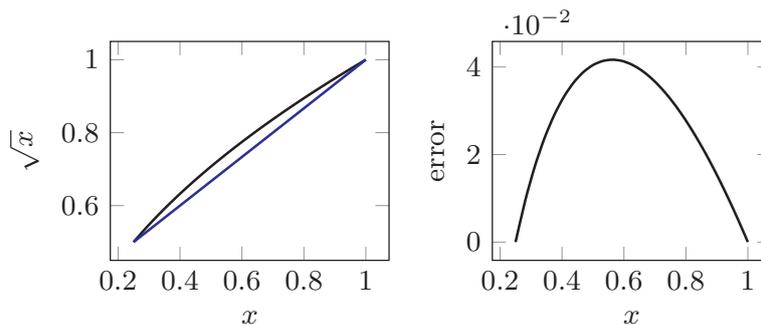
Figure 1: Linear interpolation of $f(x) = \sqrt{x}$ at $x_0 = 1/4$ and $x_1 = 1$. Left: f plotted next to the linear interpolant p_1. Right: plot of the error $|f(x) - p_1(x)|$.

2 Interpolation

Problem Suppose that a function $f: [a, b] \to \mathbb{R}$ is specified only by its values $f(x_i)$ at the $N + 1$ distinct points $x_0, x_1, \ldots, x_N$. How can we approximate $f(x)$ for all x?

In *polynomial interpolation*, we do this by constructing a polynomial p_N of degree N such that

$$p_N(x_i) = f(x_i), \qquad i = 0, \ldots, N. \tag{2.1}$$

2.1 Linear interpolation

Linear interpolation is the case $N = 1$. We are given two points $x_0 \neq x_1$ and values $f(x_0)$ and $f(x_1)$. The interpolant $p_1(x)$ is simply the straight line given by

$$p_1(x) = f(x_0) + \left(\frac{f(x_1) - f(x_0)}{x_1 - x_0} \right)(x - x_0). \tag{2.2}$$

You should check that p_1 is linear and $p_1(x_i) = f(x_i)$ for $i = 0, 1$.

Example 2.1. Approximate $f(x) = \sqrt{x}$ by linear interpolation at $x_0 = 1/4$ and $x_1 = 1$. By Eq. (2.2),

$$p_1(x) = \sqrt{\frac{1}{4}} + \left(\frac{1 - \sqrt{1/4}}{1 - \frac{1}{4}} \right)\left(x - \frac{1}{4} \right) = \frac{2}{3}x + \frac{1}{3}.$$

Figure 1 shows graphs of $f(x)$ and $p_1(x)$ for $x_0 = 1/4$ and $x_1 = 1$, and the error at each point. Figure 2 shows the same graphs for $x_0 = 0$ and $x_1 = 3/4$. The error in the second example is much bigger! Let us explain why.

Corollary 2.1 (Rolle's theorem)**.** *Let* $f: [a, b] \to \mathbb{R}$ *be smooth with* $f(a) = f(b)$. *There exists* $\xi \in (a, b)$ *such that* $f'(\xi) = 0$.

Proof. This is a special case of the mean-value theorem using $f(a) = f(b)$. $\qquad\square$

To analyse the *error in linear interpolation*, set

$$e(x) := f(x) - p_1(x).$$

Note that $e(x_0) = 0 = e(x_1)$.
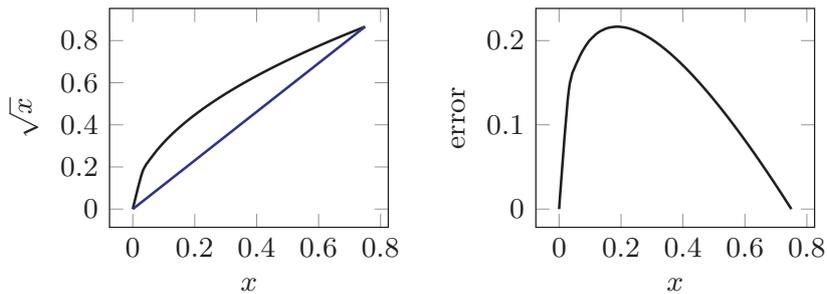
Figure 2: Linear interpolation of $f(x) = \sqrt{x}$ at $x_0 = 0$ and $x_1 = 3/4$. Left: f plotted alongside the linear interpolant p_1. Right: plot of the error $|f(x) - p_1(x)|$. Notice the range of the y-axis for the error in comparison to Figure 1.

Theorem 2.1. *Suppose that $f \colon [x_0, x_1] \to \mathbb{R}$ is smooth. For all $x \in (x_0, x_1)$, there exists $\xi \in (x_0, x_1)$ such that*

$$e(x) = \frac{f''(\xi)}{2}w_2(x), \quad where \quad w_2(x) := (x - x_0)(x - x_1). \tag{2.3}$$

Proof. Fix $x \in (x_0, x_1)$ and note that $w_2(x) \neq 0$. Define

$$g(t) := e(t) - \frac{e(x)}{w_2(x)}w_2(t) \qquad \text{for } t \in [x_0, x_1]. \tag{2.4}$$

Observe that

$$g(x_0) = e(x_0) - \frac{e(x)}{w_2(x)}w_2(x_0) = 0 - 0 = 0.$$

Similarly $g(x_1) = 0$.

Also,

$$g(x) = e(x) - \frac{e(x)}{w_2(x)}w_2(x) = e(x) - e(x) = 0.$$

The smoothness assumption of f implies smoothness of g and Rolle's theorem applies. Hence, applying Rolle's theorem twice,

$$\exists \eta_1 \in (x_0, x), \eta_2 \in (x, x_1) \quad \text{such that } g'(\eta_1) = 0 = g'(\eta_2).$$

Now consider $g'(t)$:

$$g'(t) = e'(t) - \frac{e(x)}{w_2(x)}w_2'(t).$$

Hence, Rolle's theorem implies again and

$$\exists \xi \in (\eta_1, \eta_2) \quad \text{such that } g''(\xi) = 0. \tag{2.5}$$

By Eq. (2.4),

$$g(t) = f(t) - p_1(t) - \frac{e(x)}{w_2(x)}(t - x_0)(t - x_1).$$

Since p_1 is a linear function, $p_1''(t) \equiv 0$ and so

$$g''(t) = f''(t) - 0 - 2\frac{e(x)}{w_2(x)}.$$

6

As $g''(\xi) = 0$ by Eq. (2.5),

$$0 = f''(\xi) - 2\frac{e(x)}{w_2(x)}$$

and finally this can be rearranged to show that Eq. (2.3) holds. □

We note that ξ in Eq. (2.3) depends on x and we do not know it explicitly in general. We only know that it exists. This makes the above formula for the error difficult to use. To get around this, we replace the term involving ξ by something that is easier to understand.

Definition 2.1. For any interval I and $f \colon I \to \mathbb{R}$, we define "the sup norm"

$$\left\|f\right\|_{\infty,I} := \sup_{x \in I}\left|f(x)\right|.$$

Here, sup means supremum or least upper bound and, in many cases, it is the same as finding the maximum of $|f|$ on I.

Corollary 2.2. *Under the assumptions of Theorem 2.1,*

$$\left\|e\right\|_{\infty,[x_0,x_1]} \leq \frac{(x_1 - x_0)^2}{8}\left\|f''\right\|_{\infty,[x_0,x_1]}. \tag{2.6}$$

Proof. From Theorem 2.1, for all $x \in (x_0, x_1)$,

$$\left|e(x)\right| = \frac{|f''(\xi)|}{2}|w_2(x)|, \quad \text{for some } \xi \in (x_0, x_1).$$

Also,

$$|w_2(x)| = |(x - x_0)(x - x_1)| = (x - x_0)(x_1 - x).$$

Simple calculus (see Problem E2.1) shows that

$$|w_2(x)| \leq \frac{(x_1 - x_0)^2}{4}.$$

Hence,

$$|e(x)| \leq \frac{(x_1 - x_0)^2}{8}\left\|f''\right\|_{\infty,[x_0,x_1]}.$$

This inequality also holds for $x = x_0$ and x_1 (since the left-hand side vanishes) and we have derived Eq. (2.6). □

The smoothness of f affects the quality of the approximation and we see that the error is proportional to $f''(\xi)$. The size of the derivatives of f is one way to quantify the smoothness of a function. In Figure 1, $f(x) = \sqrt{x}$ on $[1/4, 1]$ and in Figure 2 the interval is $[0, 3/4]$. Because $f''(x) \to -\infty$ as $x \to 0$, $\|f''\|_{\infty,[0,3/4]}$ is infinite and the error is much larger in the second example.

See Problem E2.4 for a computational example.
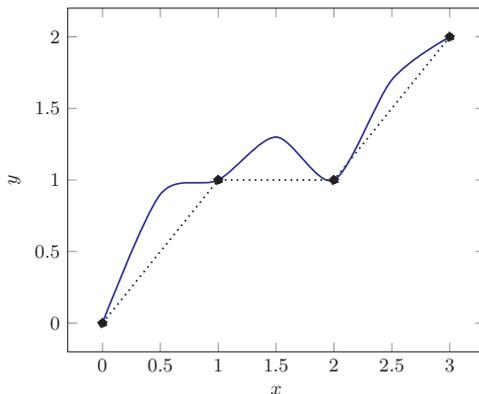
Figure 3: Piecewise-linear interpolation (solid line is f; dotted line is the piecewise-linear interpolant). In this case, the mesh is $y_0 = 0$, $y_1 = 1$, $y_2 = 2$ and $y_3 = 3$.

Piecewise-linear interpolation

To get a (more) accurate approximation to $f \colon [a, b] \to \mathbb{R}$, we subdivide $[a, b]$ into a *mesh* of points
$$a = y_0 < y_1 < \cdots < y_J = b$$
and use linear interpolation on each subinterval $[y_{j-1}, y_j]$.

Let $p_{1,J}$ denote the *piecewise-linear function* on $[a, b]$ that interpolates f at all the points y_j of the mesh and let $h_j := y_j - y_{j-1}$. By Corollary 2.2,

$$\left\| f - p_{1,J} \right\|_{\infty,[y_{j-1},y_j]} \le \frac{1}{8} h_j^2 \left\| f'' \right\|_{\infty,[y_{j-1},y_j]}.$$

Clearly,

$$\left\| f - p_{1,J} \right\|_{\infty,[a,b]} = \max_{j=1,\ldots,J} \left\| f - p_{1,J} \right\|_{\infty,[y_{j-1},y_j]}.$$

Hence, in terms of the mesh width $h := \max_{j=1,\ldots,J} h_j$,

$$\left\| f - p_{1,J} \right\|_{\infty,[a,b]} \le \frac{1}{8} h^2 \max_{j=1,\ldots,J} \left\| f'' \right\|_{\infty,[y_{j-1},y_j]} = \frac{1}{8} h^2 \left\| f'' \right\|_{\infty,[a,b]}. \tag{2.7}$$

Convergence is achieved as $h \to 0$ and the error is $\mathcal{O}(h^2)$.

At the points y_j, the function f is only required to be continuous! This convergence analysis does not actually require f to be twice differentiable at y_j: If all discontinuities in f' are resolved by the mesh, we can deal with less smooth functions in piecewise interpolation.

Example 2.2. Let $f(x) = \exp(x^2)$ on $[a, b] = [0, 1]$ and let $y_j = jh, j = 0, \ldots, J$, where $h = 1/J$ (this is called a *uniform mesh*). In Problem E3.1, you will write a program to compute

$$e_h := \max_{j=1,\ldots,J} \left| (f - p_{1,J})(z_j) \right|$$

where $z_j := (y_{j-1} + y_j)/2$ (the midpoint of $[y_{j-1}, y_j]$). A discrete set of points is used for the maximum instead of the whole interval, to allow easy computation. *Results:*

h	e_h	$(e_h)/(e_{h/2})$	bound(h)
1/8	2.60e-2	3.62	3.18e-2
1/16	7.19e-3	3.80	7.95e-3
1/32	1.89e-3	3.90	1.99e-3
1/64	4.85e-4		4.97e-4

To estimate the rate of convergence, we conjecture that $e_h = Ch^\alpha$. Then,

$$(e_h)/(e_{h/2}) = 2^\alpha.$$

The third column suggests α approaches 2. To prove this rigorously, note that

$$f''(x) = (4x^2 + 2)\exp(x^2) \qquad \Rightarrow \qquad \left\|f''\right\|_{\infty,[0,1]} \le 6\exp(1).$$

Hence, from Eq. (2.7),

$$e_h \le \left\|f - p_{1,J}\right\|_{\infty,[0,1]} \le \frac{1}{8}h^2\left\|f''\right\|_{\infty,[0,1]} \le \frac{3}{4}\exp(1)h^2 =: \text{bound}(h).$$

Note how sharp the theoretical bound is (in the table)!

2.2   Degree-N interpolation

When f is smooth, better accuracy is possible by choosing the *degree-N* polynomial that interpolates at $N + 1$ distinct points instead. Let $\mathcal{P}_N$ denote the polynomials of degree N or less.

Problem   Given $N + 1$ distinct points $x_0, \ldots, x_N$ and values $f(x_0), \ldots, f(x_N)$, compute a polynomial $p_N \in \mathcal{P}_N$ with the property that

$$p_N(x_i) = f(x_i), \qquad i = 0, \ldots, N. \tag{2.8}$$

We finish this chapter with two theorems:

Theorem 2.2 (existence and uniqueness)**.** *Let $x_0, x_1, \cdots, x_N$ be distinct points in $[a, b]$ and suppose $f\colon [a, b] \to \mathbb{R}$ is continuous. Then, there exists a unique $p_N \in \mathcal{P}_N$ satisfying $p_N(x_i) = f(x_i)$ for $i = 0, \ldots, N$.*

Proof. Let

$$L_j(x) := \frac{\prod_{i \neq j}(x - x_i)}{\prod_{i \neq j}(x_j - x_i)},$$

where $\prod$ denotes the product of each term. These are known as *Lagrange basis functions* and are degree-N polynomials (i.e., $L_j \in \mathcal{P}_N$). Note that $L_j(x_k) = \delta_{jk}$ (the Kronecker delta function), with $\delta_{jk} = 0$ if $j \neq k$ and $\delta_{jk} = 1$ if $j = k$. Define

$$p_N(x) = \sum_{j=0}^{N} f(x_j)L_j(x).$$

Then, evaluating at $x = x_i$, we have

$$p_N(x_i) = \sum_{j=0}^{N} f(x_j)\delta_{ij} = f(x_i).$$

This is a degree-N interpolant of f and we have proved existence.

To show uniqueness, let $p, q \in \mathcal{P}_N$ both satisfy Eq. (2.8). Then p, q agree at $N + 1$ distinct points and $r := p - q$ is a degree-N polynomial with $N + 1$ distinct roots. This can only happen if $r \equiv 0$ and the polynomials p and q are identical. $\qquad\square$

The following theorem generalises Theorem 2.1 to general N. We assume the points are well ordered so that $x_i < x_{i+1}$.

Theorem 2.3. *Suppose the conditions of Theorem 2.2 hold and that f is smooth. Then, for all $x \in [x_0, x_N]$, there exists $\xi \in (x_0, x_N)$ such that*

$$(f - p_N)(x) = \frac{f^{(N+1)}(\xi)}{(N+1)!} w_{N+1}(x).$$

where $w_{N+1}(x) = (x - x_0) \times \cdots \times (x - x_N)$.

Proof. Not covered. $\qquad\square$

Notice that now derivatives of order $N+1$ determine the quality of the approximation. It is easy to show that $w_{N+1}(x) = \mathcal{O}(h^{N+1})$ if $|x_i - x_j| \leq h$. Hence, if f is $(N+1)$-times continuously differentiable, $\|f - p_N\|_{\infty, [x_0, x_N]} = \mathcal{O}(h^{N+1})$.

Care is needed to apply high-degree polynomial interpolation, especially with uniformly spaced points; see Figure 4.

2.3 Newton's divided-difference formulae

Newton provided an elegant way of writing interpolation formulae, which is especially useful when adding more interpolation points. Let's derive his divided differences and the associated formulae for polynomial interpolation. We approximate a function $f \colon \mathbb{R} \to \mathbb{R}$.

One data point We are given $(x_0, f(x_0))$ and the constant interpolation function is
$p_0(x) = f(x_0)$.

Two data points Given a second point $(x_1, f(x_1))$, we wish to update the interpolation function by increasing the polynomial degree. We write

$$p_1(x) = p_0(x) + A_1(x - x_0),$$

where A_1 is a coefficient to be determined. Notice that $p_1(x_0) = f(x_0)$ for any choice of A_1; we automatically satisfy the first interpolation condition. To determine A_1, apply the second interpolation condition $p_1(x_1) = f(x_1)$, to find

$$A_1 = \frac{f(x_1) - p_0(x)}{x_1 - x_0} = \frac{f(x_1) - f(x_0)}{x_1 - x_0}.$$

This quantity is Newton's first divided-difference and usually denoted $f[x_0, x_1]$ (or $f[x_1, x_0]$ as order does not matter here). The linear interpolant is $p_1(x) = f(x_0) + f[x_0, x_1](x - x_0)$.
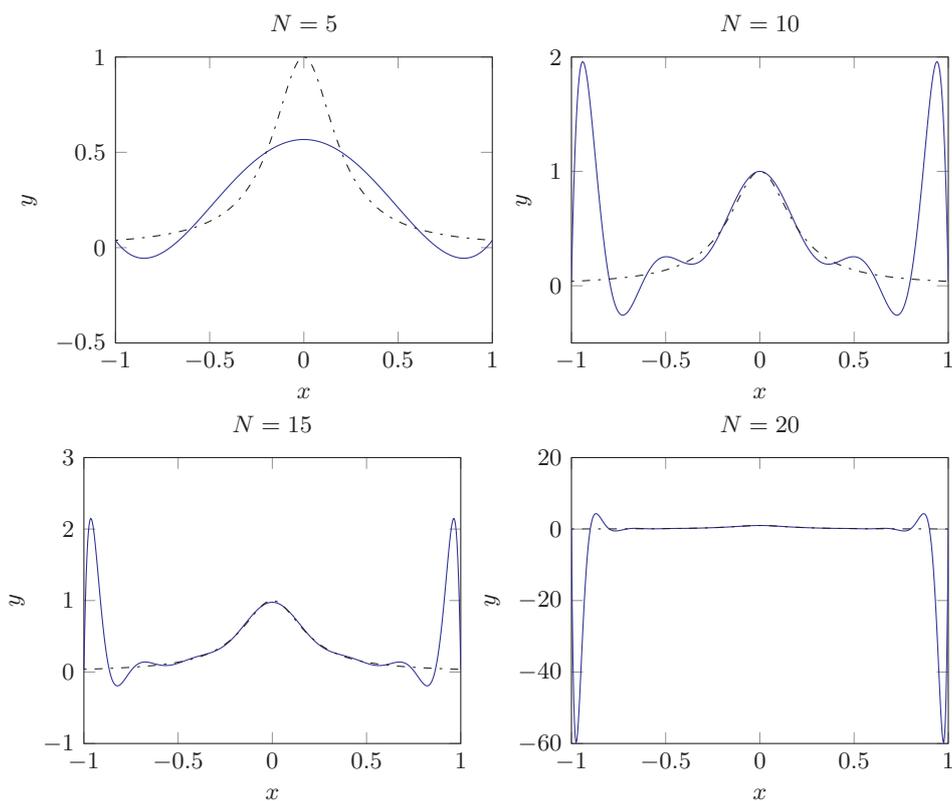
Figure 4: The solid line shows the interpolant $p_N \in \mathcal{P}_N$ of $f(x) = 1/(25x^2 + 1)$ based on $N+1$ uniformly spaced points on the interval $[-1, 1]$. Note how the oscillations near the end points become wilder as N is increased. The difficulty with this choice of $f(x)$ is its derivatives, which become larger as roughly speaking each derivative increases by a factor of 25 and Theorem 2.3 requires the derivatives to be well behaved.

Three data points Add another data point and build the degree-two interpolant $p_2(x)$ by adding to the already-known degree-one interpolation. Write

$$p_2(x) = p_1(x) + A_2(x - x_0)(x - x_1)$$

for some A_2 to be determined. Notice that $p_2(x)$ satisfies the first two interpolation conditions, and A_2 is determined by $p_2(x_2) = f(x_2)$. That is,

$$f(x_2) = p_1(x_2) + A_2(x_2 - x_0)(x_2 - x_1)$$

so that

$$A_2 = \frac{f(x_2) - f(x_0) - f[x_0, x_1](x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)} = \frac{f[x_2, x_0] - f[x_0, x_1]}{(x_2 - x_1)} =: f[x_0, x_1, x_2],$$

which is the second Newton divided-difference. You should verify that permuting $[x_0, x_1, x_2]$ leaves its definition unchanged. The quadratic interpolant $p_2(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1)$.

Four data points The pattern continues. For example, if we add a fourth point $x_4, f(x_4)$, the degree-three interpolant is written

$$\begin{aligned} p_3(x) = {} & f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ & + f[x_0, x_1, x_2, x_3](x - x_0)(x - x_1)(x - x_2), \end{aligned}$$

where the Newton divided-difference is defined by

$$f[x_0, \ldots, x_n] := \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0},$$

which is invariant to permutation of its arguments.

3 Numerical integration

Problem Given a function $f \colon [a, b] \to \mathbb{R}$, compute approximations of $\int_a^b f(x)\,dx$ using only samples of f at some points in the interval $[a, b]$.

Numerical integration is important since, for many functions f, the integral cannot be found exactly, but point values of f are relatively easy to compute (e.g., $f(x) = \exp(x^2)$). Initially, we work on a reference interval $[a, b] = [0, 1]$ and consider first

$$\int_0^1 f(x)\,dx. \tag{3.1}$$

We approximate this by *quadrature rules* of the form

$$Q(f) = \sum_{i=0}^{N} w_i f(x_i) \tag{3.2}$$

where x_i are distinct points in $[0, 1]$ and w_i are some suitable *weights*. We study ways to fix the x_i and the w_i independently of any particular choice of f, so that Eq. (3.2) can be computed easily for any f. How to choose x_i and w_i?

Idea: Replace f with an interpolating polynomial. Polynomials are simple to integrate and lead to easy-to-use quadrature rules with a set of weights w_i for any given set of points x_i.

3.1 Newton–Cotes rules

Newton–Cotes over the interval $[0, 1]$:

Start with equally spaced points $x_i = i/N$, $i = 0, \ldots, N$, and find suitable weights w_i. We construct the rule (3.2) by integrating the degree-N interpolating polynomial $p_N(x)$ for f at x_i.

Two points: For $N = 1$, we have two points $x_0 = 0$ and $x_1 = 1$. Then (see §2), the linear interpolant to f is

$$p_1(x) = f(0) + (f(1) - f(0))\,x$$

and

$$\int_0^1 p_1(x)\,dx = f(0) + (f(1) - f(0)) \int_0^1 x\,dx$$
$$= f(0) + \frac{1}{2}(f(1) - f(0)) = \frac{1}{2}\Big(f(0) + f(1)\Big).$$

We have derived the *trapezoidal or trapezium rule*

$$Q_1(f) := \frac{1}{2}\Big(f(0) + f(1)\Big). \tag{3.3}$$

Here the weights $w_0 = w_1 = \frac{1}{2}$. It is called the *trapezoidal rule* because it approximates $\int_0^1 f(x)\,dx$ by the area of the trapezium under the straight line $p_1(x)$ that interpolates f between 0 and 1. It is exact (i.e., $Q_1(f) = \int_0^1 f(x)\,dx$) for any polynomial f of degree 1 (i.e., $f \in \mathcal{P}_1$), since in that case $p_1 = f$ by the uniqueness of the linear interpolant (recall Theorem 2.2).
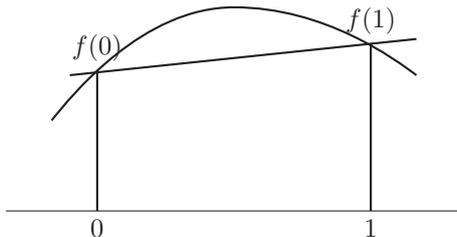
Figure 5: The trapezoidal rule

Three points: For $N = 2$, we have three points $x_0 = 0$, $x_1 = 1$, and $x_2 = 1/2$. Using the Newton divided-difference form of the quadratic interpolant,

$$p_2(x) = f(0) + f[0, 1]x + f[0, 1, 1/2]x(x - 1).$$

Then,

$$\int_0^1 p_2(x)\,dx = f(0) + \frac{1}{2}f[0, 1] + f[0, 1, 1/2]\underbrace{\int_0^1 (x^2 - x)\,dx}_{=-1/6}$$

$$= f(0) + \frac{1}{2}\frac{f(1) - f(0)}{1} - \frac{1}{6}\frac{f[0, 1/2] - f[0, 1]}{1/2 - 1},$$

where $f[0, 1/2] = 2(f(1/2) - f(0))$ and $f[0, 1] = f(1) - f(0)$. Thus,

$$\int_0^1 p_2(x)\,dx = f(0) + \frac{1}{2}(f(1) - f(0)) + \frac{1}{3}\Big(2f(1/2) - 2f(0) + f(0) - f(1)\Big)$$

$$= \frac{1}{6}\Big[f(0) + 4f\left(\tfrac{1}{2}\right) + f(1)\Big].$$

This is called *Simpson's rule.* We write

$$Q_2(f) := \frac{1}{6}f(0) + \frac{4}{6}f\left(\tfrac{1}{2}\right) + \frac{1}{6}f(1), \tag{3.4}$$

with weights $w_0 = w_1 = \frac{1}{6}$ and $w_2 = \frac{4}{6}$. This rule is exact for all $f \in \mathcal{P}_2$, since again in that case $p_2 = f$.

In general, *Newton–Cotes quadrature rules* are of the form

$$Q_N(f) := \sum_{i=0}^{N} w_i f(x_i)$$

where $x_i = i/N$ for $i = 0, \ldots, N$, and the *weights* w_i can be found by integrating the degree-N interpolating polynomial. By writing $p_N(x)$ using the Lagrange basis functions (see the proof of Theorem 2.2),

$$p_N(x) = \sum_{i=0}^{N} L_i(x)f(x_i),$$

we can show that

$$w_i = \int_0^1 L_i(x)\,dx.$$

Newton–Cotes rules over a general interval $[a, b]$: Suppose now we have a rule

$$Q(f) = \sum_{i=0}^{N} w_i f(x_i) \qquad \text{that approximates} \qquad \int_0^1 f(x) \, dx. \qquad (3.5)$$

To approximate $\int_a^b f(t) \, dt$, we make a change of variable $x = \dfrac{t - a}{b - a}$ and $dx = \frac{1}{b-a} dt$. Then, $t \in [a, b]$ is mapped to $x \in [0, 1]$ and

$$\int_a^b f(t) \, dt = \int_0^1 \underbrace{(b - a) f\Big(a + x(b - a)\Big)}_{=: \, g(x)} \, dx = \int_0^1 g(x) \, dx.$$

We now use $Q(g)$ to approximate the right-hand side, to obtain the rule

$$Q^{[a,b]}(f) := \sum_{i=0}^{N} w_i g(x_i) = (b - a) \sum_{i=0}^{N} w_i f\Big(a + (b - a)x_i\Big). \qquad (3.6)$$

Notation. We use superscript $[a, b]$ to denote a rule over $[a, b]$. We omit the superscript for $[0, 1]$. The lowest-order Newton–Cotes rules on a general interval $[a, b]$ are

$$Q_1^{[a,b]}(f) = \frac{b - a}{2}\Big(f(a) + f(b)\Big), \qquad\qquad \text{trapezoidal}$$

$$Q_2^{[a,b]}(f) = \frac{b - a}{6}\Big(f(a) + 4f\Big(\frac{b + a}{2}\Big) + f(b)\Big), \qquad \text{Simpson's.}$$

Example 3.1. Let us apply these to the integral of $f(x) = \sqrt{x}$ over $[1/4, 1]$. Then

$$Q_1^{[1/4,1]}(f) = \frac{1 - \frac{1}{4}}{2}\Big(\sqrt{1/4} + 1\Big) = \frac{3}{8}\frac{3}{2} = 9/16 = 0.5625, \qquad \text{trapezoidal}$$

$$Q_2^{[1/4,1]}(f) = \frac{1 - \frac{1}{4}}{6}\Big(\sqrt{1/4} + 4\sqrt{5/8} + 1\Big) = 0.582785 \quad \text{(6 s.f.)}, \quad \text{Simpson's.}$$

The exact value

$$\int_{1/4}^1 \sqrt{x} \, dx = \frac{2}{3}\Big[x^{3/2}\Big]_{1/4}^1 = \frac{7}{12} = 0.583333 \quad \text{(6 s.f.)},$$

so the absolute value of the error in Simpson's rule is $|0.582785 - 0.583333| = 5.48 \times 10^{-4}$, which is about 38 times smaller than the error in the trapezoidal rule $|9/16 - 0.583333| = 2.083 \times 10^{-2}$.

Composite rules

As for the piecewise interpolation in §2, instead of increasing the accuracy of quadrature by choosing higher-order interpolating polynomials, we can also split the integration domain into subintervals by introducing a mesh and applying low-order rules on the sub-intervals.

Consider a general quadrature rule (e.g., from §3.1)

$$Q(f) = \sum_{i=0}^{N} w_i f(x_i) \quad \text{that approximates} \quad \int_0^1 f(x)\, dx, \tag{3.7}$$

for some N, weights w_i and distinct points x_i. To approximate $\int_a^b f(t)\, dt$, we introduce the mesh $a = y_0 < y_1 < \cdots < y_J = b$ and write

$$\int_a^b f(t)\, dt = \int_{y_0}^{y_1} f(t)\, dt + \int_{y_1}^{y_2} f(t)\, dt + \cdots + \int_{y_{J-1}}^{y_J} f(t)\, dt = \sum_{j=1}^{J} \int_{y_{j-1}}^{y_j} f(t)\, dt.$$

Now use the quadrature rule given by Eq. (3.7) on each subinterval, to obtain the approximation

$$\int_a^b f(t)\, dt \approx \sum_{j=1}^{J} Q^{[y_{j-1}, y_j]}(f) = \sum_{j=1}^{J} h_j \sum_{i=0}^{N} w_i f(y_{j-1} + h_j x_i), \tag{3.8}$$

where $h_j = y_j - y_{j-1}$. The approximation Eq. (3.8) is known as the *composite version* of Eq. (3.7).

As in the case of interpolation, we expect that accuracy will increase when the mesh width $\max_j h_j \to 0$.

Example 3.2 (composite trapezoidal rule).

$$Q_{1,J}^{[a,b]}(f) := \sum_{j=1}^{J} Q_1^{[y_{j-1}, y_j]}(f) = \sum_{j=1}^{J} \frac{h_j}{2} \Big(f(y_{j-1}) + f(y_j) \Big). \tag{3.9}$$

For a uniform mesh, $h_j = h := (b - a)/J$, $j = 1, \ldots, J$, this can be written compactly as

$$Q_{1,J}^{[a,b]}(f) := h \left(\frac{f(a)}{2} + \sum_{j=1}^{J-1} f(y_j) + \frac{f(b)}{2} \right).$$

Example 3.3 (composite Simpson's rule).

$$Q_{2,J}^{[a,b]}(f) := \sum_{j=1}^{J} Q_2^{[y_{j-1}, y_j]}(f) = \sum_{j=1}^{J} \frac{h_j}{6} \Big(f(y_{j-1}) + 4f(m_j) + f(y_j) \Big), \tag{3.10}$$

where the midpoints $m_j = \dfrac{y_{j-1} + y_j}{2}$. For a uniform mesh, this can again be written more compactly as

$$Q_{2,J}^{[a,b]}(f) = \frac{h}{6} \left[f(a) + 2 \sum_{j=1}^{J-1} f(y_j) + 4 \sum_{j=1}^{J} f(m_j) + f(b) \right].$$

Please implement the general formulae Eq. (3.9) and Eq. (3.10) (see Problems E4.2 and E4.3), so that we can apply the rules on any mesh.

3.2 Error analysis

Non-composite rules over $[0, 1]$

Given a rule

$$Q(f) := \sum_{i=0}^{N} w_i f(x_i) \quad \text{that approximates} \quad I(f) := \int_0^1 f(x)\, dx, \qquad (3.11)$$

we define the error in the quadrature rule to be

$$E(f) := I(f) - Q(f). \qquad (3.12)$$

A way into the analysis is via the following concept.

Definition 3.1 (DOP)**.** The rule $Q(f)$ in Eq. (3.11) has *degree of precision (DOP)* $d \in \mathbb{N}$ if

$$E(x^r) = 0 \text{ for all } r \in \mathbb{N} \text{ with } 0 \le r \le d \qquad \text{and} \qquad E(x^{d+1}) \ne 0.$$

Example 3.4 (trapezoidal rule)**.** $Q_1(f) = \frac{1}{2}(f(0) + f(1))$ and $E_1(f) := I(f) - Q_1(f)$.

r	x^r	$I(x^r)$	$Q_1(x^r)$	$E_1(x^r)$
0	1	1	$\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 1 = 1$	0
1	x	$\frac{1}{2}$	$\frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 1 = \frac{1}{2}$	0
2	x^2	$\frac{1}{3}$	$\frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 1 = \frac{1}{2}$	$-\frac{1}{6}$

Hence the DOP of the trapezoidal rule is 1.

Example 3.5 (Simpson's rule)**.** $Q_2(f) = \frac{1}{6}\left[f(0) + 4f\left(\frac{1}{2}\right) + f(1)\right]$ and $E_2(f) := I(f) - Q_2(f)$.

r	x^r	$I(x^r)$	$Q_2(x^r)$	$E_2(x^r)$
0	1	1	$\frac{1}{6}(1 \cdot 1 + 4 \cdot 1 + 1 \cdot 1) = 1$	0
1	x	$\frac{1}{2}$	$\frac{1}{6}(1 \cdot 0 + 4 \cdot \frac{1}{2} + 1 \cdot 1) = \frac{1}{2}$	0
2	x^2	$\frac{1}{3}$	$\frac{1}{6}(1 \cdot 0 + 4 \cdot (\frac{1}{2})^2 + 1 \cdot 1) = \frac{1}{3}$	0
3	x^3	$\frac{1}{4}$	$\frac{1}{6}(1 \cdot 0 + 4(\frac{1}{2})^3 + 1 \cdot 1) = \frac{1}{4}$	0
4	x^4	$\frac{1}{5}$	$\frac{1}{6}(1 \cdot 0 + 4(\frac{1}{2})^4 + 1 \cdot 1) = \frac{5}{24}$	$-\frac{1}{120}$

Hence the DOP of Simpson's rule is 3.

Q_1 is found by integrating p_1 and it is not surprising that its DOP is 1. Q_2 is found by integrating p_2 and we'd expect a DOP of at least 2. A DOP of 3 is a surprise. In fact, it turns out that for all $N \in \mathbb{N}$ the Newton–Cotes rule $Q_N(f)$ is of DOP N if N is odd and of DOP $N + 1$ if N is even.

Proposition 3.1. *If Eq. (3.11) has DOP d, then*

$$E(p) = 0, \quad \text{for all} \quad p \in \mathcal{P}_d.$$

Proof. For any $f_1, f_2 \colon [0,1] \to \mathbb{R}$ and $\alpha, \beta \in \mathbb{R}$,

$$I(\alpha f_1 + \beta f_2) \;=\; \alpha I(f_2) + \beta I(f_2) \quad \text{and} \quad Q(\alpha f_1 + \beta f_2) \;=\; \alpha Q(f_2) + \beta Q(f_2), \quad (3.13)$$

since I and Q are both linear transformations.

If $p \in \mathcal{P}_d$, p is a polynomial of degree less or equal to d and can be written

$$p(x) = \sum_{r=0}^{d} a_r x^r, \quad \text{for some coefficients } a_r \in \mathbb{R}.$$

Hence, using Eq. (3.13),

$$I(p) - Q(p) = \sum_{r=0}^{d} a_r \Big[I(x^r) - Q(x^r) \Big] = 0,$$

since the DOP equals d. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \square$

With significant further work, we can show that, for the trapezoidal rule, there exists $\xi \in [0,1]$ such that

$$E_1(f) = \left[\frac{E_1(x^2)}{2!} \right] f''(\xi) = -\frac{1}{12} f''(\xi), \qquad \text{for all } f \in C^2[0,1]. \qquad (3.14)$$

For $p \in \mathcal{P}_1$, $p'' \equiv 0$ and $E_1(p) = 0$ as expected from the DOP calculation.

For Simpson's rule, there exists $\xi \in [0,1]$ such that

$$E_2(f) = \left[\frac{E_2(x^4)}{4!} \right] f^{(4)}(\xi) = -\frac{1}{2880} f^{(4)}(\xi), \qquad \text{for all } f \in C^4[0,1].$$

Notice the error for Simpson's rule depends on the fourth derivative of f while that for the trapezoidal rule depends on the second derivative. Here $C^k[a,b]$ is the set of functions $f \colon [a,b] \to \mathbb{R}$ that are k-times continuously differentiable.

This leads to estimates over $[a,b]$ instead of $[0,1]$ by a change of variables.

Example 3.6 (errors on $[a,b]$). The error for the trapezoidal rule over $[a,b]$ is

$$E_1^{[a,b]}(f) := \int_a^b f(t)\, dt - Q_1^{[a,b]}(f).$$

To determine the error, we recall that

$$\int_a^b f(t)\, dt = \int_0^1 g(x)\, dx, \qquad Q_1^{[a,b]}(f) = Q_1(g),$$

from Eq. (3.6) with $g(x) = (b-a)f(a + (b-a)x)$. Now $g'(x) = (b-a)^2 f'(a + (b-a)x)$ and $g''(x) = (b-a)^3 f''(a + (b-a)x)$. By Eq. (3.14),

$$E_1(g) = \int_0^1 g(x)\, dx - Q_1(g) = -\frac{1}{12} g''(\xi) = -\frac{1}{12}(b-a)^3 f''(\eta)$$

for some $\xi \in [0,1]$ and $\eta := a + (b-a)\xi$. Then,

$$E_1^{[a,b]}(f) = -\frac{(b-a)^3}{12} f''(\eta) \qquad \text{for some} \quad \eta \in [a,b].$$

A similar calculation shows that the error for Simpson's rule over $[a,b]$ is

$$E_2^{[a,b]}(f) := \int_a^b f(t)\, dt - Q_2^{[a,b]}(f) = -\frac{(b-a)^5}{2880} f^{(4)}(\eta) \qquad \text{for some } \eta \in [a,b].$$

Notice the fifth power of $(b-a)$, which comes from the change of coordinates from t to x.

Composite Newton–Cotes rules

Let Q_N be a Newton–Cotes rule on $[0, 1]$ with degree of precision d. The composite version with J subintervals on $[a, b]$ is (as in Eq. (3.8)):

$$Q_{N,J}^{[a,b]}(f) = \sum_{j=1}^{J} Q_N^{[y_{j-1}, y_j]}(f).$$

The error can be expressed in terms of the error made in approximating the sub-integrals.

$$E_{N,J}^{[a,b]}(f) := \int_a^b f(x)\, dx - Q_{N,J}^{[a,b]}(f) = \sum_{j=1}^{J} \underbrace{\left[\int_{y_{j-1}}^{y_j} f(x)\, dx - Q_N^{[y_{j-1}, y_j]}(f) \right]}_{= E_N^{[y_{j-1}, y_j]}(f)}.$$

We have formula for $E_N^{[a,b]}$ for $N = 1, 2$, which lead to the following error estimates for the composite trapezoidal and Simpson's rule.

Example 3.7. Composite trapezoidal rule: If $f \in C^2[a, b]$, then there exists $\eta_j \in [y_{j-1}, y_j]$ such that

$$E_{1,J}^{[a,b]}(f) = -\frac{1}{12} \sum_{j=1}^{J} h_j^3 f^{(2)}(\eta_j) \qquad \text{and} \qquad \left| E_{1,J}^{[a,b]}(f) \right| \le \frac{b-a}{12} \left\| f^{(2)} \right\|_{\infty, [a,b]} h^2,$$

since $\sum_{j=1}^{J} h_j = b - a$.

Example 3.8. Composite Simpson's rule: If $f \in C^4[a, b]$, then there exists $\eta_j \in [y_{j-1}, y_j]$ such that

$$E_{2,J}^{[a,b]}(f) = -\frac{1}{2880} \sum_{j=1}^{J} h_j^5 f^{(4)}(\eta_j) \qquad \text{and} \qquad \left| E_{2,J}^{[a,b]}(f) \right| \le \frac{b-a}{2880} \left\| f^{(4)} \right\|_{\infty, [a,b]} h^4.$$

If $f(x)$ fails to be sufficiently differentiable on all of $[a, b]$, but is sufficiently differentiable on subintervals of $[a, b]$, we can apply error estimates there.

Example 3.9. Consider $f(x) = \sqrt{x}$ on $[0, 1]$, which has infinitely many derivatives on subintervals that do not contain 0, but no bounded derivatives on $[0, 1]$. Consider the composite trapezoidal rule for $\int_0^1 f(x)\, dx$ on the mesh $0 = y_0 < y_1 < \cdots < y_J = 1$. Then

$$\begin{aligned} E_{1,J}^{[0,1]}(f) &= E_{1,J}^{[0,y_1]}(f) + E_{1,J}^{[y_1,1]}(f) \\ &= \left(\int_0^{y_1} f(x)\, dx - h_1 \frac{\sqrt{y_1}}{2} \right) - \frac{1}{12} \sum_{j=2}^{J} h_j^3 f^{(2)}(\eta_j). \end{aligned} \qquad (3.15)$$

Now, we can estimate each of the terms in Eq. (3.15) separately (see Problem E4.3).

3.3 Gaussian quadrature

The only examples of quadrature rules so far have been Newton–Cotes rules with equally spaced points. Can we do better with other points?

To take advantage of symmetry and simplify calculations, we work on the interval $[-1, 1]$ rather than $[0, 1]$ (which can of course be transformed onto $[0, 1]$). Let x_i, $i = 0, \ldots, N$, be *arbitrary points* in $[-1, 1]$, and let $p_N(x)$ be the degree-N interpolating polynomial for a function f at these points. Consider the rule:

$$Q_{\text{Gauss},N}^{[-1,1]}(f) := \int_{-1}^{1} p_N(x) \, dx \qquad (3.16)$$

as an approximation to

$$\int_{-1}^{1} f(x) \, dx. \qquad (3.17)$$

This rule has DOP at least N. Can we do better with a clever choice of points? There are $2N + 2$ degrees of freedom (given by choice of x_i and w_i for $i = 0, \ldots, N$) and $d + 1$ conditions to achieve of a DOP of d. By equating the number of conditions to the number of degrees of freedom, $d + 1 = 2N + 2$, we hope that a DOP $d = 2N + 1$ can be achieved by careful choice of x_i and weights w_i.

Example 3.10. One point ($N = 0$). To achieve a DOP of $d = 2N + 1 = 1$, we demand that

$$Q(1) = \int_{-1}^{1} 1 \, dx = 2 \qquad \text{and} \qquad Q(x) = \int_{-1}^{1} x \, dx = 0. \qquad (3.18)$$

As $p_0(x)$ is a constant, we must have $p_0(x) = f(x_0)$ and

$$Q(f) = w_0 f(x_0).$$

Then, $w_0 = 2$ and $x_0 = 0$ gives Eq. (3.18). Therefore, the *one-point Gauss rule* (or *midpoint rule*), obtained by integrating the degree-0 interpolant p_0 at $x_0 = 0$ over $[-1, 1]$ is

$$Q_{\text{Gauss},0}^{[-1,1]}(f) := \int_{-1}^{1} p_0(x) \, dx = 2f(0).$$

This is exact when $f = 1$ and $f = x$, so it is a one-point rule with DOP $= 1$. Any other one-point rule has only DOP $= 0$.

Example 3.11. Two points ($N = 1$). To achieve a DOP $d = 2N + 1 = 3$, we demand that

$$Q(1) = 2, \qquad Q(x) = 0, \qquad Q(x^2) = \frac{2}{3}, \qquad \text{and} \qquad Q(x^3) = 0.$$

As

$$Q(f) = w_0 f(x_0) + w_1 f(x_1),$$

we must have $w_0 + w_1 = 2$ and $w_0 x_0 + w_1 x_1 = 0$ and $w_0 x_0^2 + w_1 x_1^2 = 2/3$ and $w_0 x_0^3 + w_1 x_1^3 = 0$. By symmetry considerations, we must have $x_0 = -x_1$ and $w_0 = w_1$. Then $w_0 = w_1 = 1$ and $2x_0^2 = 2/3$, so that $x_0 = \sqrt{1/3}$. We obtain the *two-point Gauss rule*

$$Q_{\text{Gauss},1}^{[-1,1]}(f) := \int_{-1}^{1} p_1(x) \, dx = f\left(\frac{1}{\sqrt{3}}\right) + f\left(-\frac{1}{\sqrt{3}}\right).$$

Although only a two-point rule, its DOP is $2N + 1 = 3$. Compare with the trapezoidal rule which uses two points and has DOP only 1!

Remark 3.1.
- Composite Gauss rules can also be derived and are highly effective.

- High-frequency integrands, where f oscillates rapidly and where the derivatives of f are large, are particularly difficult and arise, for example, in high-frequency scattering applications (e.g., radio waves). This requires special techniques such as *Filon* quadrature.

- Multivariate integrals are also important. For low-dimensional problems, simple *tensor-product rules* (applying one-dimensional rules in each dimension) work fine and our theory carries over easily. For high-dimensional integrals, the only feasible methods currently are *Monte Carlo-type methods*.

4 Nonlinear equations and IVPs for ODEs

Root-finding problem   For a given function $f \colon \mathbb{R} \to \mathbb{R}$, determine a solution x of the equation $f(x) = 0$. A solution x is known as a root of f.

For most f, there is no formula to give x explicitly and numerical methods are required. For example, we may use the bisection method and choose an interval $[a, b]$ that contains the root by checking that $f(a) > 0$ and $f(b) < 0$ or vice versa (f changes sign and has a root in the interval if it is continuous). Bisecting the interval and choosing one subinterval where f changes signs at the endpoints gives a new interval containing a root. Then, we iterate to find successively smaller intervals and better approximations to the root. Given a suitable initial interval, this bisection method is simple to apply but does not generalise easily to higher dimension. Instead, we focus on fixed-point iterations.

4.1 Fixed-point iteration

Definition 4.1 (root, fixed point). We say x is a root of a function f if $f(x) = 0$, and x is a fixed point (FP) of a function g if $g(x) = x$.

Often, root-finding problems can be replaced by fixed-point (FP) problems:

Example 4.1. Let $f(x) = x^3 + 4x^2 - 10$. There are many ways of posing this as a FP problem $g(x) = x$.

- Let $g_1(x) = x - x^3 - 4x^2 + 10$. Then, it is easy to check that $g_1(x) = x$ if and only if $f(x) = 0$.

- Let $g_2(x) = \frac{1}{2}(10 - x^3)^{1/2}$ (positive root). For $x > 0$, $g_2(x) = x$ if and only if $f(x) = 0$.

- Let $g_3(x) = \left(\frac{10}{4+x}\right)^{1/2}$, which is again a FP problem for the root-finding problem for f.

Define the sequence x_n by $x_{n+1} = g(x_n)$, given an initial condition x_0. Under what conditions does x_n converge to a fixed point of g and can this be used for computing the root? To answer this question, we have the fixed-point theorem. We use the term 'smooth' to mean the function has enough continuous derivatives.

Theorem 4.1 (convergence of FP iteration). *Let $g \colon [a, b] \to \mathbb{R}$ be a smooth function. Then, if (i) $g(x) \in [a, b]$ for $x \in [a, b]$ and (ii) $|g'(x)| \leq \lambda < 1$ for $x \in [a, b]$, then the sequence x_n defined by $x_{n+1} = g(x_n)$ for any $x_0 \in [a, b]$ converges to the unique fixed point x of g. Further,*
$$|x_n - x| \leq \lambda^n |x - x_0|.$$

As well as convergence of the FP iteration, this theorem also gives existence and uniqueness of the FP of g in $[a, b]$.

Example 4.2. We look back at the fixed point problems in Example 4.1:

- $g_1(x) = x - x^3 - 4x^2 + 10$ and $[a, b] = [1, 2]$. Then $g_1(1) = 6$ and condition (i) fails. The FP theorem does not apply to the iteration based on g_1.

- $g_2(x) = \frac{1}{2}(10 - x^3)^{1/2}$ and

$$g_2'(x) = \frac{1}{4}\left(10 - x^3\right)^{-1/2}(-3x^2) = \frac{-3x^2}{4(10 - x^3)^{1/2}},$$

$$g_2'(2) = \frac{-3 \cdot 4}{4(10 - 8)^{1/2}} = \frac{-3}{\sqrt{2}} \approx -2.12.$$

  Hence, $|g_2'(x)| > 1$ and (ii) fails. Again, the FP theorem does not apply.

- $g_3(x) = \left(\frac{10}{4+x}\right)^{1/2}$ and

$$g_3'(x) = \frac{1}{2}\left(\frac{10}{4+x}\right)^{-1/2}\left(\frac{-10}{(4+x)^2}\right) = \frac{-5}{(4+x)^{3/2}\sqrt{10}}.$$

  As g_3 is decreasing and $g_3(2) = \sqrt{10/6} \in [1, 2]$ and $g_3(1) = \sqrt{2} \in [1, 2]$, we see that (i) holds.

  Further,

$$|g_3'(x)| \leq \frac{5}{\sqrt{10}}\frac{1}{5^{3/2}} < 1 \qquad \text{for } x \in [1, 2].$$

  Hence (ii) holds. The FP theorem applies and $x_n \to x$, the unique fixed point of g, and the root of f: try it,

$$x_1 = 1.5, \qquad x_2 = g_3(1.5) \approx 1.3484, \qquad x_3 \approx 1.3674, \qquad x_4 \approx 1.365.$$

  We see that the first three digits of x_n have already converged and that $f(1.365) = -0.0038$, indicating that 1.365 is close to the root of f.

For the proof of the FP theorem, we use the mean-value theorem from MA10207.

Theorem 4.2 (mean-value theorem). *Let $f\colon [a, b] \to \mathbb{R}$ be smooth. There exists $\xi \in (a, b)$ such that*

$$\frac{f(b) - f(a)}{b - a} = f'(\xi).$$

Proof of Theorem 4.1. Let $f(x) = g(x) - x$. Then, by (i), $f(a) = g(a) - a \geq 0$ and $f(b) = g(b) - b \leq 0$. By the intermediate-value theorem, there exists $x \in [a, b]$ such that $f(x) = 0$. In other words, there exists $x \in [a, b]$ so that $g(x) = x$.

Consider the iteration $x_{n+1} = g(x_n)$ and the fixed-point equation $x = g(x)$. Then,

$$x_{n+1} - x = g(x_n) - g(x).$$

By the mean-value theorem, there exists $\xi \in (a, b)$ so that

$$x_{n+1} - x = g'(\xi)(x_n - x)$$

(as g is smooth). Now $|g'(\xi)| \leq \lambda$ and

$$|x_{n+1} - x| \leq \lambda|x_n - x|.$$

By a simple induction argument, this implies that $|x_n - x| \leq \lambda^n|x_0 - x|$.

Finally, to show uniqueness, consider two fixed-points x, y of g. Then $g(x) = x$ and $g(y) = y$ and hence

$$x - y = g(x) - g(y) = g'(\xi)(x - y).$$

As $|g'(\xi)| \leq \lambda$, we see that

$$|x - y| \leq \lambda|x - y|.$$

As $\lambda < 1$, it must hold that $x = y$ and there is only one fixed point of g in $[a, b]$. $\qquad \square$

4.2 Newton's method

The most well-known example of a fixed-point iteration is Newton's method. This is the iteration

$$x_{n+1} = g(x_n), \qquad g(x) := x - \frac{f(x)}{f'(x)},$$

where we assume that $f'(x) \neq 0$. Clearly, $f(x) = 0$ if and only if $g(x) = x$. See Figure 6.
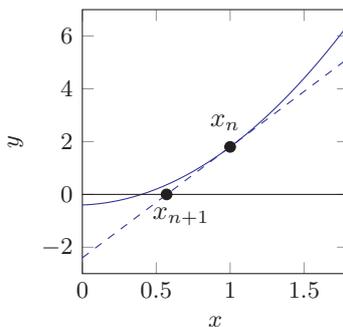
Figure 6: The solid line shows $y = f(x)$ and the dashed line is the tangent $y = f(x_n) + (x - x_n)f'(x_n)$ at $x = x_n$. Newton's method finds the point x_{n+1} where the tangent line crosses the x-axes.

We show the FP theorem applies:

Theorem 4.3 (local convergence of Newton's method). *Suppose that f is smooth and that $f(x) = 0$ and $f'(x) \neq 0$. Then, there exists $\epsilon > 0$ so that Newton's method converges to the root x of f if the initial guess $x_0 \in [x - \epsilon, x + \epsilon]$.*

Proof. With $g(x) = x - f(x)/f'(x)$, we have

$$g'(x) = 1 - \frac{f'(x)}{f'(x)} + \frac{f(x)f''(x)}{f'(x)^2} = 0 \tag{4.1}$$

as $f(x) = 0$. As f and g are smooth, we have $|g'(y)| \leq \lambda := \frac{1}{2}$ for $y \in [x - \epsilon, x + \epsilon]$ by choosing ϵ sufficiently small. This gives (ii) of the FP theorem for $a = x - \epsilon$ and $b = x + \epsilon$. For (i), note that

$$|g(y) - x| = |g(y) - g(x)| \leq |g'(\xi)| \, |x - y| \leq \frac{1}{2}\epsilon,$$

for any $y \in [a, b]$ and some $\xi \in (a, b)$ by the mean-value theorem. Clearly, then $g(y) \in [x - \epsilon, x + \epsilon] = [a, b]$ and (i) of the FP theorem holds. We conclude that Newton's method converges for initial conditions close (as given by ϵ) to x. $\qquad \square$

This theorem is problematic for the practitioner: it says that Newton's method converges if we can start close enough to the root! We don't usually know the root and we don't usually know what ϵ is (i.e., what close enough means). However, Newton's method is often effective and, when it works, it is often very fast.

To quantify the speed of convergence, we define *the order of convergence*.

Definition 4.2 (order of convergence). Consider a sequence x_n approximating x. Let $e_n := |x_n - x|$, the error in the approximation. We say that x_n converges to x with *order r* if

   (a) case $r = 1$ (linear convergence): $e_{n+1} \le K e_n$ for all $n \in \mathbb{N}$, for some $K < 1$;

   (b) case $r > 1$: $e_{n+1} \le K e_n^r$ for all $n \in \mathbb{N}$, for some $K > 0$. The case $r = 2$ is known as quadratic convergence.

   One of the most useful tools in numerical analysis is Taylor's theorem from MA10207:

Theorem 4.4 (Taylor's theorem). *Suppose that $f \colon (a, b) \to \mathbb{R}$ is $(m + 1)$-times continuously differentiable and suppose $x_0, x \in (a, b)$ with $x_0 \ne x$. Then,*

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2}(x - x_0)^2 + \cdots + \frac{f^{(m)}(x_0)}{m!}(x - x_0)^m + R_m(x),$$

where $f^{(m)}$ denotes the mth derivative of f and

$$R_m(x) = \frac{f^{(m+1)}(\xi)}{(m + 1)!}(x - x_0)^{m+1},$$

for some ξ lying strictly between x and x_0.

Theorem 4.5 (Newton's method). *If f is smooth and the initial guess x_0 is sufficiently close to the root x, then Newton's method converges quadratically; that is, for some $K > 0$,*

$$e_{n+1} = |x_{n+1} - x| \le K|x_n - x|^2 = K e_n^2,$$

where $e_n = |x_n - x|$ represents the error at step n.

Proof. Use Taylor's theorem, to write

$$g(y) = g(x) + g'(x)(y - x) + \frac{1}{2}g''(\xi)(y - x)^2$$

for some ξ. We know from the calculation in Eq. (4.1) that $g'(x) = 0$ and hence

$$g(y) - g(x) = g(y) - x = \frac{1}{2}g''(\xi)(y - x)^2.$$

We know that the FP theorem applies in some interval $[a, b] = [x - \epsilon, x + \epsilon]$. Hence, if $x_0 \in [a, b]$ then so does x_n for $n \in \mathbb{N}$. Hence, it is enough to take $y \in [a, b]$ and also $\xi \in [a, b]$. Let $K := \frac{1}{2}\max_{\xi \in [a,b]} |g''(\xi)|$, which is finite as g is smooth. Then, with $y = x_n$, we have

$$\left|x_{n+1} - x\right| \le K\left|x_n - x\right|^2. \qquad \square$$

Example 4.3. Note that $f(\pi) = 0$ for $f(x) = \sin(x)$. Newton's method is the iteration

$$x_{n+1} = x_n - \frac{\sin(x_n)}{\cos(x_n)} = x_n - \tan(x_n).$$

Then, we take an initial condition $x_0 = 3$ and

$$x_1 = 3.142546543074278, \quad x_2 = 3.141592653300477, \quad x_3 = 3.141592653589793$$

by iterating `x=x-tan(x)`. The example illustrates nicely quadratic convergence and we see the number of correct digits increases rapidly (3, 10, 11 digits; the last one is affected by rounding error).

4.3   Solution of Initial-Value Problems (IVPs)

An *IVP in standard form* is a system of N differential equations:

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(\boldsymbol{y}, t), \quad t \geq 0, \tag{4.2}$$

where $\boldsymbol{f} \colon \mathbb{R}^{N+1} \to \mathbb{R}^N$ is a given function and $\boldsymbol{y}(t) \in \mathbb{R}^N$ is the solution (for $t > 0$), to be found subject to initial conditions of the form:

$$\boldsymbol{y}(0) = \boldsymbol{y}_0, \tag{4.3}$$

with initial data $\boldsymbol{y}_0 \in \mathbb{R}^N$ given.

Example 4.4 ($N = 1$). Let $f(y, t) = y^2$ and $y_0 = 1$. Then

$$\frac{dy}{dt} = y^2, \quad \text{subject to} \quad y(0) = 1.$$

It is easy to see that the exact solution is $y(t) = 1/(1 - t)$. (Please check!) Usually the exact solution is not so easy to find.

Not all problems are presented as Eq. (4.2).

Example 4.5. A simple pendulum is released from rest at angle α. The angle $\theta = \theta(t)$ of the pendulum satisfies the second-order equation:

$$\frac{d^2\theta}{dt^2} = \frac{g}{a}\sin\theta, \tag{4.4}$$

with g and a constants, subject to two initial conditions:

$$\theta(0) = \alpha, \quad \frac{d\theta}{dt}(0) = 0.$$

To put in standard form, introduce the new variable ϕ defined by $\phi = d\theta/dt$. Then Eq. (4.4) becomes

$$\frac{d\theta}{dt} = \phi,$$
$$\frac{d\phi}{dt} = \frac{g}{a}\sin\theta.$$

which has the form Eq. (4.2) with

$$\boldsymbol{y} = \begin{bmatrix} \theta \\ \phi \end{bmatrix} \quad \text{and} \quad \boldsymbol{f}(\boldsymbol{y}, t) = \begin{bmatrix} \phi \\ \frac{g}{a}\sin\theta \end{bmatrix}.$$

4.4 Euler's method

Given Eqs. (4.2) and (4.3), suppose we want to find $\boldsymbol{y}(t)$ for $t = h$ for some small h. Integrating each side of Eq. (4.2) over $[0, h]$ gives

$$\boldsymbol{y}(h) - \boldsymbol{y}(0) = \int_0^h \frac{d\boldsymbol{y}}{dt}\, dt = \int_0^h f(\boldsymbol{y}(t), t)\, dt \tag{4.5}$$

The integral on the right-hand side is still unknown, but we can approximate it, for example, by replacing the integrand by its value at $t = 0$ (which will be OK if h is small). This yields

$$\boldsymbol{y}(h) \approx \boldsymbol{y}(0) + h\, \mathbf{f}(\boldsymbol{y}(0), 0) = \boldsymbol{y}_0 + h\, \mathbf{f}(\boldsymbol{y}_0, 0).$$

Everything on the right-hand side is known so we can compute an approximation of $\boldsymbol{y}(h)$. Then $\boldsymbol{y}(2h)$ can be approximated by taking another step, and so on. This is *Euler's Method*, which computes a sequence of approximations $\boldsymbol{Y}_j$ to $\boldsymbol{y}(t_j)$, where $t_j = jh$, $j = 1, 2, \ldots$ by

$$\boldsymbol{Y}_0 = \boldsymbol{y}_0$$
$$\boldsymbol{Y}_j = \boldsymbol{Y}_{j-1} + h\, \mathbf{f}(\boldsymbol{Y}_{j-1}, t_{j-1}), \qquad \text{for all } j \geq 1.$$

Example 4.6. Solve

$$\frac{dy}{dt} = y^2, \qquad y(0) = 1 \tag{4.6}$$

using Euler's method with $h = 0.1$; that is,

$$Y_0 = y_0 = 1$$
$$Y_1 = Y_0 + hf(Y_0) = 1 + 0.1(1^2) = 1.1$$
$$Y_2 = Y_1 + hf(Y_1) = 1.1 + 0.1(1.1)^2 = 1.221.$$

Using the program, we approximate the solution of Eq. (4.6) at the time $T = 1/2$ for various h. The number of steps taken in Euler's method is then $n = T/h$. Since we know $y(1/2) = 2$ in this case (see Eq. (4.2)), we can find the error exactly.

Results:

$n = T/h$	h	$\lvert y(1/2) - Y_n \rvert$	Ratio
4	1/8	0.2338	0.59
8	1/16	0.1389	0.55
16	1/32	0.07696	0.53
32	1/64	0.04073	

The ratios for the errors approach $1/2$, which suggests that $\lvert Y_n - y(t) \rvert \cong \mathcal{O}(h)$ as $h \to 0$. This is what we prove in the next section.

4.5 Convergence of one-step methods

We now restrict to the simplified version of Eq. (4.2) with $N = 1$ and $f(y, t) = f(y)$:

$$\frac{dy}{dt} = f(y) \tag{4.7}$$

subject to the initial condition

$$y(0) = y_0. \tag{4.8}$$

We shall consider *"one step"* methods of the form :

$$\begin{aligned} Y_0 &= y_0 \\ Y_j &= Y_{j-1} + h\, F_h(Y_{j-1}), \qquad \text{for all } j \geq 1 \end{aligned} \tag{4.9}$$

where F_h is a function to be specified.

An example is Euler's method, where $F_h \equiv f$. The error at the jth step is defined to be

$$e_j = y(t_j) - Y_j \quad \text{where} \quad t_j = jh.$$

Definition 4.3. If y solves Eqs. (4.7) and (4.8), the *local truncation error* for Eq. (4.9) is defined to be

$$\tau_j = \frac{y(t_j) - y(t_{j-1})}{h} - F_h(y(t_{j-1}))\,, \qquad \text{where} \quad t_j = jh.$$

(i.e., $h\tau_j$ is the discrepancy when the true solution is substituted into Eq. (4.9)).

The convergence analysis proceeds by (i) bounding the error in the computed solutions Y_j in terms of the local truncation errors and (ii) estimating the local truncation error using Taylor's theorem.

Definition 4.4. A continuous function $g \colon \mathbb{R} \to \mathbb{R}$ is called *Lipschitz continuous* with Lipschitz constant $L > 0$ if

$$|g(Y) - g(Z)| \leq L\,|Y - Z| \qquad \text{for all } Y, Z \in \mathbb{R}.$$

Theorem 4.6. *Suppose F_h is Lipschitz continuous with Lipschitz constant L independent of h. Then the error e_j in Eq. (4.9) satisfies*

$$|e_j| \leq (1 + hL)|e_{j-1}| + h|\tau_j|, \quad j = 1, 2, 3, \ldots. \tag{4.10}$$

Moreover, for all fixed T and all $n \in \mathbb{N}$ satisfying $nh \leq T$,

$$|e_n| \leq \frac{\exp(TL) - 1}{L} \max_{1 \leq j \leq n} |\tau_j|.$$

Proof. By definition of τ_j,

$$y(t_j) = y(t_{j-1}) + h\, F_h(y(t_{j-1})) + h\,\tau_j\,. \tag{4.11}$$

So with $e_j = y(t_j) - Y_j$, we have by subtracting Eq. (4.9) from Eq. (4.11):

$$e_j = e_{j-1} + h\left(F_h(y(t_{j-1})) - F_h(Y_{j-1}) \right) + h\,\tau_j.$$

By the triangle inequality and the Lipschitz continuity of F_h,

$$\begin{aligned} |e_j| &\leq |e_{j-1}| + h\,|F_h(y(t_{j-1})) - F_h(Y_{j-1})| + h\,|\tau_j| \\ &\leq (1 + hL)|e_{j-1}| + h\,|\tau_j|. \end{aligned}$$

We see that Eq. (4.10) holds.

Now we shall prove by induction that, for all $n \geq 1$,

$$|e_n| \leq h \sum_{j=0}^{n-1} (1 + hL)^j |\tau_{n-j}|. \qquad (4.12)$$

Clearly (4.12) holds for $n = 1$, since the first equation in Eq. (4.9) implies $|e_0| = 0$ and (4.10) then gives $|e_1| \leq h|\tau_1|$. Now if (4.12) holds for some n, then (4.10) implies

$$|e_{n+1}| \leq h \sum_{j=0}^{n-1} (1 + hL)^{j+1} |\tau_{n-j}| + h |\tau_{n+1}|$$

$$= h \sum_{j=1}^{n} (1 + hL)^j |\tau_{n+1-j}| + h |\tau_{n+1}|$$

$$= h \sum_{j=0}^{n} (1 + hL)^j |\tau_{n+1-j}|.$$

Hence, (4.12) holds for $n + 1$, and for all n by induction.

Finally, from (4.12) we have, since $\sum_{j=0}^{n-1} q^j = \dfrac{q^n - 1}{q - 1}$,

$$|e_n| \leq \frac{(1 + hL)^n - 1}{L} \max_{1 \leq j \leq n} |\tau_j| = \frac{\exp(nhL) - 1}{L} \max_{1 \leq j \leq n} |\tau_j|.$$

since $1 + x \leq \exp x$, for all $x \geq 0$. And so the result follows for all $nh \leq T$. $\qquad \square$

Remark: This theorem shows that the error in the approximation to the solution computed by Eq. (4.9) at the point $t_n = nh$ will approach 0 if all the local truncation errors τ_j, $j = 1, \ldots, n$, approach 0, as $h \to 0$.

Normally the local truncation error is estimated by applying Taylor's theorem.

Example 4.7. Euler's method is Eq. (4.9) with $F_h(Y) := f(Y)$. If we assume that f is Lipschitz continuous, then Theorem 4.6 applies and to show convergence we have to estimate τ_j.

To do this we write (using the definition of τ_j):

$$\tau_j = \frac{y(t_j) - y(t_{j-1})}{h} - f(y(t_{j-1})) = \frac{y(t_{j-1} + h) - y(t_{j-1})}{h} - f(y(t_{j-1}))$$

which we can expand via Taylor's Theorem, with $\xi_j \in (t_{j-1}, t_j)$, such that

$$\tau_j = \frac{\left(y(t_{j-1}) + h\dfrac{dy}{dt}(t_{j-1}) + \dfrac{h^2}{2}\dfrac{d^2 y}{dt^2}(\xi_j) - y(t_{j-1}) \right)}{h} - f(y(t_{j-1}))$$

$$= \left[\frac{dy}{dt}(t_{j-1}) - f(y(t_{j-1})) \right] + \frac{h}{2}\frac{d^2 y}{dt^2}(\xi_j).$$

Now, since $y(t)$ is the solution of Eq. (4.7), we have, for all $j = 1, \ldots, n$:

$$|\tau_j| \leq \frac{1}{2} \max_{t \in [0,T]} \left| \frac{d^2 y}{dt^2}(t) \right| h.$$

Hence if

$$\left|\frac{d^2y}{dt^2}(t)\right| \quad \text{is bounded for } t \in \mathbb{R}, \tag{4.13}$$

then Theorem 4.6 implies that

$$|e_n| \leq C(T)h, \tag{4.14}$$

where $C(T)$ is a constant depending on T; and, for fixed T, we have convergence (i.e., $e_n \to 0$ as $h \to 0$).

Often one just assumes Eq. (4.13) and then concludes Eq. (4.14). To show Eq. (4.13) rigorously, we need to make some assumptions on the given function f. In particular, assume that

$$|f(x)| \leq M, \quad x \in \mathbb{R},$$

$$\text{and} \quad |f'(x)| \leq L, \quad x \in \mathbb{R}.$$

Then f is Lipschitz continuous with Lipschitz constant L, and by Eq. (4.7) and the chain rule,

$$\left|\frac{d^2y}{dt^2}(t)\right| = \left|\frac{d}{dt}f(y(t))\right| = \left|f'(y(t))\frac{dy}{dt}(t)\right| = |f'(y(t))||f(y(t))| \leq LM.$$

4.6 Higher-order methods

We saw in Eq. (4.14) that Euler's Method converges with $\mathcal{O}(h)$. This is relatively slow. Higher-order methods can be found by employing higher-order quadrature in Eq. (4.5). For simplicity, restrict to the case the case $N = 1$ and $f(y,t) = f(y)$ again: Then Eq. (4.5) is

$$y(h) - y(0) = \int_0^h f(y(t)) \, dt.$$

Instead of approximating the right-hand side by the one-point quadrature rule at 0, consider using instead the trapezoidal rule, to obtain

$$y(h) \approx y(0) + \frac{h}{2}\Big[f(y(0)) + f(y(h))\Big]. \tag{4.15}$$

This motivates the *Crank–Nicolson (or Trapezoidal) Method* for solving Eq. (4.2):

$$\begin{aligned} Y_0 &= y_0 \\ Y_j &= Y_{j-1} + \frac{h}{2}\Big[f(Y_{j-1}) + f(Y_j)\Big], \qquad \text{for all } j \geq 1. \end{aligned} \tag{4.16}$$

This method has a big disadvantage, since to find Y_j from Y_{j-1} in Eq. (4.16), we have to solve a (possibly nonlinear) equation

$$Y_j - \frac{h}{2}f(Y_j) = Y_{j-1} + \frac{h}{2}f(Y_{j-1}).$$

If there are N differential equations, then there are N (possibly nonlinear) equations to solve at each timestep. Despite this extra cost such *"implicit" methods* are often preferred because of their good *stability* properties. (Stability is an advanced topic.)

To estimate the local truncation error for Eq. (4.16), write the method as

$$\frac{Y_j - Y_{j-1}}{h} = \frac{1}{2}\Big[f(Y_{j-1}) + f(Y_j)\Big].$$

Then using Eq. (4.7) and applying Taylor's theorem to both y and dy/dt we get:

$$
\begin{aligned}
\tau_j &= \frac{y(t_j) - y(t_{j-1})}{h} - \frac{1}{2}[f(y(t_{j-1})) + f(y(t_j))] \\
&= \frac{dy}{dt}(t_{j-1}) + \frac{h}{2}\frac{d^2y}{dt^2}(t_{j-1}) + \frac{h^2}{6}\frac{d^3y}{dt^3}(t_{j-1}) - \frac{1}{2}\left[\frac{dy}{dt}(t_{j-1}) + \frac{dy}{dt}(t_j)\right] + \mathcal{O}\!\left(h^3\right) \\
&= \left(\frac{dy}{dt} + \frac{h}{2}\frac{d^2y}{dt^2} + \frac{h^2}{6}\frac{d^3y}{dt^3} - \frac{dy}{dt} - \frac{h}{2}\frac{d^2y}{dt^2} - \frac{h^2}{4}\frac{d^3y}{dt^3}\right)(t_{j-1}) + \mathcal{O}\!\left(h^3\right) \\
&= -\frac{1}{12}\frac{d^3y}{dt^3}(t_{j-1})\,h^2 + \mathcal{O}\!\left(h^3\right),
\end{aligned}
$$

Hence, provided y has three bounded derivatives, $|\tau_j| = \mathcal{O}(h^2)$. Note that

$$\frac{d^3y}{dt^3} = \frac{d^2}{dt^2}f(y) = \frac{d}{dt}\Big(f'(y)y'\Big) = \frac{d}{dt}\Big(f'(y)f(y)\Big) = f''(y)(f(y))^2 + (f'(y))^2 f(y)\,.$$

Theorem 4.7. *Suppose f is Lipschitz continuous with Lipschitz constant L independent of h. If $hL \leq 1$, then the error $e_j = y(t_j) - Y_j$ for the Crank–Nicolson method in Eq. (4.16) satisfies*

$$|e_j| \leq (1 + hL)^2 \,|e_{j-1}| + h(1 + hL)\,|\tau_j|, \quad j = 1, 2, 3, \ldots. \tag{4.17}$$

Moreover, for all fixed T and all $n \in \mathbb{N}$ satisfying $nh \leq T$,

$$|e_n| \leq \frac{\exp(2TL) - 1}{L} \max_{1 \leq j \leq n} |\tau_j|.$$

Proof. From Eq. (4.16),

$$Y_j = Y_{j-1} + \frac{h}{2}\left[f(Y_{j-1}) + f(Y_j)\right]$$

and by definition of the truncation error

$$y(t_j) = y(t_{j-1}) + \frac{h}{2}\Big[f(y(t_{j-1})) + f(y(t_j))\Big] + h\tau_j\,.$$

Hence, subtracting

$$|e_j| = \left|e_{j-1} + \frac{h}{2}\Big[f(y(t_{j-1})) - f(Y_{j-1})\Big] + \frac{h}{2}\Big[f(y(t_j)) - f(Y_j)\Big] + h\tau_j\right|$$

and, using the triangle inequality and Lipschitz continuity of f, we have

$$|e_j| \leq |e_{j-1}| + \frac{hL}{2}\,|e_{j-1}| + \frac{hL}{2}\,|e_j| + h\,|\tau_j|.$$

Rearrange as follows.

$$\left(1 - \frac{hL}{2}\right)|e_j| \leq \left(1 + \frac{hL}{2}\right)|e_{j-1}| + h\,|\tau_j|$$

$$\Rightarrow \quad |e_j| \leq \left(1 - \frac{hL}{2}\right)^{-1}\left[\left(1 + \frac{hL}{2}\right)|e_{j-1}| + h\,|\tau_j|\right].$$

ow, for any $0 \leq x \leq \frac{1}{2}$,

$$(1 - x)^{-1} \leq 1 + 2x \qquad (\textbf{Check!})$$

and so for $hL \leq 1$ (since $(1 - \frac{hL}{2})^{-1} \leq 1 + hL$ and $1 + \frac{hL}{2} \leq 1 + hL$)

$$|e_j| \leq (1 + hL)^2|e_{j-1}| + h(1 + hL)\,|\tau_j|.$$

The rest of the proof is Problem E7.3. $\qquad\qquad\qquad\qquad\qquad\qquad\square$

Higher-order explicit methods

There are ways of achieving higher order without using implicitness. Assume that we have computed an approximation Y_{j-1} to $y(t_{j-1})$. The *improved Euler method* uses first the standard Euler method to get an approximation $\widehat{Y}_j$ to $y(t_j)$ and then the trapezoidal rule to improve it:

$$\widehat{Y}_j = Y_{j-1} + hf(Y_{j-1}) \qquad\qquad \text{"prediction"} \qquad\qquad (4.18)$$

$$Y_j = Y_{j-1} + \frac{h}{2}\Big[f(Y_{j-1}) + f(\widehat{Y}_j)\Big] \qquad \text{"correction"} \qquad (4.19)$$

This method fits into the framework of Theorem 4.6 because it can be written:

$$Y_j = Y_{j-1} + \frac{h}{2}\Big(f(Y_{j-1}) + f\big(Y_{j-1} + hf(Y_{j-1})\big)\Big),$$

which is of the form Eq. (4.9) with $F_h(Y) := \frac{1}{2}f(Y) + \frac{1}{2}f\big(Y + hf(Y)\big)$. The truncation error is

$$\tau_j = \frac{y(t_j) - y(t_{j-1})}{h} - \frac{1}{2}\Big[f(y(t_{j-1})) + f\big(y(t_{j-1}) + h\,f(y(t_{j-1}))\big)\Big],$$

and it turns out that $|\tau_j| = \mathcal{O}(h^2)$ for sufficiently smooth f (see Problem E7.2).

Moreover if f is Lipschitz then so is F_h, since

$$|F_h(Y) - F_h(Z)| \leq \frac{1}{2}|f(Y) - f(Z)| + \frac{1}{2}|f(Y + hf(Y)) - f(Z + hf(Z))|$$

$$\leq \frac{L}{2}|Y - Z| + \frac{L}{2}|(Y + hf(Y)) - (Z + hf(Z))|$$

$$\leq L|Y - Z| + \frac{hL}{2}|f(Y) - f(Z)| \leq \left(L + \frac{1}{2}hL^2\right)|Y - Z|.$$

So under these conditions, Theorem 4.6 implies that the improved Euler method converges with order $\mathcal{O}(h^2)$, at the small additional cost of an extra evaluation of f at each time step.

Higher-order methods can be built up using more evaluations of f. In general these methods are called *Runge–Kutta methods*.

Example 4.8. *Let*

$$
\left.
\begin{aligned}
K_1 &= f(Y_0), \\
K_2 &= f(Y_0 + \frac{h}{2}K_1), \\
K_3 &= f(Y_0 + \frac{h}{2}K_2), \\
K_4 &= f(Y_0 + hK_3), \\
Y_1 &= Y_0 + \frac{h}{6}[K_1 + 2K_2 + 2K_3 + K_4].
\end{aligned}
\right\}
\tag{4.20}
$$

This is an order-4 Runge–Kutta method (requires some analysis).

5 Solution of linear system of equations: $A\boldsymbol{x} = \boldsymbol{b}$

Problem   For a given $d \times d$ matrix A and vector $\boldsymbol{b} \in \mathbb{R}^d$, find $\boldsymbol{x} \in \mathbb{R}^d$ such that $A\boldsymbol{x} = \boldsymbol{b}$. If the entries of A are a_{ij} (row i, column j) and the entries of $\boldsymbol{x}$ and $\boldsymbol{b}$ are x_j and b_j, this means

$$\sum_{j=1}^{d} a_{ij} x_j = b_i, \qquad i = 1, \ldots, d.$$

You have studied already row-reduction techniques. In numerical analysis, these are developed in a way to improve numerical stability into the standard technique "Gaussian elimination with partial pivoting". This is an example of a *direct method* and this means the solution $\boldsymbol{x}$ is found by a finite number of arithmetic operations. Packages such as `scipy.linalg` use Gaussian elimination to find an LU factorisation.

Definition 5.1. A matrix P is a permutation matrix if each row and column has exactly one non-zero entry equal to one; multiplication by a permutation matrix P in PA permutes the rows of A.
   A matrix L is lower triangular if $L_{ij} = 0$ for $i < j$ and unit lower triangular if additionally $L_{ii} = 1$. A matrix U is upper triangular if $U_{ij} = 0$ for $i > j$.

The LU factorisation consists of a permutation matrix P, a unit lower-triangular matrix L, and an upper-triangular matrix U such that $PA = LU$; this can be found in Python using `P, L, U = scipy.linalg.lu(A)`. For example,

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1/2 & 1/2 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 4 & -1 & 1 \\ 0 & -1 & 2 \\ 0 & 0 & -3/2 \end{bmatrix}, \quad P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

is an LU factorisation of

$$A = \begin{bmatrix} 2 & -1 & 0 \\ 4 & -1 & 1 \\ 0 & -1 & 2 \end{bmatrix}.$$

You should verify that $PA = LU$. The entries of L and U express the row reductions normally performed in Gaussian elimination. We do not show how to compute L and U in detail by hand. There are many matrix factorisation in numerical linear algebra and this one is useful for solving linear systems of equations.

Example 5.1 (LU factorisation). To solve a linear system of equations with the LU factorisation, substitute $PA = LU$ into the linear system $A\boldsymbol{x} = \boldsymbol{b}$:

$$PA\boldsymbol{x} = LU\boldsymbol{x} = P\boldsymbol{b}.$$

Let $\boldsymbol{y} := U\boldsymbol{x}$. Given $\boldsymbol{b} \in \mathbb{R}^d$, we solve the triangular system

$$L\boldsymbol{y} = P\boldsymbol{b}, \qquad \text{to find } \boldsymbol{y} \in \mathbb{R}^d$$

and

$$U\boldsymbol{x} = \boldsymbol{y}, \qquad \text{to find } \boldsymbol{x} \in \mathbb{R}^d.$$

We have replaced the problem of solve $A\boldsymbol{x} = \boldsymbol{b}$ by solving two much simpler linear systems. Both matrices are triangular and their solution is easily found by forward or backward substitution.

We spend most of our time studying *iterative methods*, where $\boldsymbol{x}$ occurs as the limit of approximations $\boldsymbol{x}_n$ as $n \to \infty$. In general, direct methods (such as the LU factorisation or Gaussian elimination) are good for dense matrices (where $a_{ij} \neq 0$ for nearly all i, j) and the complexity of such a linear solve is $\mathcal{O}(d^3)$. If d is very large, it may be impossible to store A in memory and to perform row reductions. On the other hand, the matrix may be *sparse* ($a_{ij} = 0$ for many i, j) and it may be easy to compute matrix–vector products $A\boldsymbol{x}$. In this case, iterative methods are valuable.

We will work with the following example of a sparse matrix throughout.

Example 5.2 (finite-difference matrix)**.** Suppose that u is a smooth real-valued function on $[0, 1]$ and we want to approximate its second derivative based on evaluations on the mesh $x_i = ih$ for some mesh spacing $h = 1/(d + 1)$. By Taylor's theorem,

$$u(x + h) = u(x) + hu'(x) + \frac{1}{2}h^2 u''(x) + \frac{1}{6}h^3 u'''(x) + \mathcal{O}\big(h^4\big),$$

$$u(x - h) = u(x) - hu'(x) + \frac{1}{2}h^2 u''(x) - \frac{1}{6}h^3 u'''(x) + \mathcal{O}\big(h^4\big).$$

Then,

$$u''(x) = \frac{u(x + h) - 2u(x) + u(x - h)}{h^2} + \mathcal{O}\big(h^2\big).$$

By dropping the $\mathcal{O}(h^2)$ term, we have a *finite-difference* approximation to the second derivative. This can be used to find an approximate solution to the two-point boundary value problem: for a given function $f \colon [0, 1] \to \mathbb{R}$, find $u(x)$ such that

$$-u''(x) = f(x), \qquad u(0) = u(1) = 0.$$

Using the finite-difference approximation,

$$-\begin{bmatrix} u''(x_1) \\ u''(x_2) \\ \vdots \\ u''(x_d) \end{bmatrix} = \frac{1}{h^2} A \begin{bmatrix} u(x_1) \\ u(x_2) \\ \vdots \\ u(x_d) \end{bmatrix} + \mathcal{O}\big(h^2\big)$$

for

$$A := \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 2 \end{bmatrix}. \qquad (5.1)$$

Then $u''(x) = -f(x)$ gives $\frac{1}{h^2} A\boldsymbol{u} = \boldsymbol{f}$ if we neglect the $\mathcal{O}(h^2)$ term, for

$$\boldsymbol{f} := \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_d) \end{bmatrix}, \qquad \boldsymbol{u} := \begin{bmatrix} u(x_1) \\ u(x_2) \\ \vdots \\ u(x_d) \end{bmatrix}.$$

We have a linear system of equations that can be solved to determine an approximate solution of the boundary-value problem.

Only the main and two off-diagonals of A are non-zero. All other entries are zero and the matrix is sparse. We will use the finite-difference matrix A as a prototype example as we develop iterative methods. The matrix is typical of the ones that arise in the numerical solution of PDEs.

5.1 Iterative methods

Suppose we wish to solve $A\boldsymbol{x} = \boldsymbol{b}$ for $\boldsymbol{x} \in \mathbb{R}^d$ given a $d \times d$ matrix A and $\boldsymbol{b} \in \mathbb{R}^d$. Write $A = A_1 - A_2$ so that

$$A_1 \boldsymbol{x} = A_2 \boldsymbol{x} + \boldsymbol{b}.$$

This motivates the following iterative method: find $\boldsymbol{x}_{n+1}$ such that

$$A_1 \boldsymbol{x}_{n+1} = A_2 \boldsymbol{x}_n + \boldsymbol{b}.$$

When $\boldsymbol{x}_n$ is known and A_1 is well chosen, we easily find $\boldsymbol{x}_{n+1}$ and generate a sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ that we hope converges to the solution $\boldsymbol{x}$. We can interpret this as a fixed-point iteration and

$$\boldsymbol{x}_{n+1} = \boldsymbol{g}(\boldsymbol{x}_n), \qquad \boldsymbol{g}(\boldsymbol{x}) := A_1^{-1}(A_2 \boldsymbol{x} + \boldsymbol{b}),$$

where we assume the inverse matrix A_1^{-1} exists.

Example 5.3 (Jacobi). Let A_1 denote the diagonal part of A and $A_2 = A_1 - A = -(L + U)$ (for the lower- and upper-triangular parts L and U of A). Take for example

$$A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}, \qquad A_1 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

The Jacobi iteration is

$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \boldsymbol{x}_{n+1} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \boldsymbol{x}_n + \boldsymbol{b}$$

or

$$\boldsymbol{x}_{n+1} = \begin{bmatrix} 0 & 1/2 \\ 1/2 & 0 \end{bmatrix} \boldsymbol{x}_n + \frac{1}{2}\boldsymbol{b}.$$

Notice how simple it is to evaluate the right-hand side given $\boldsymbol{x}_n$ and $\boldsymbol{b}$.

For the finite-difference example (5.1), the Jacobi method is

$$\boldsymbol{x}_{n+1} = \frac{1}{2} \begin{bmatrix} 0 & 1 & & & \\ 1 & \ddots & 1 & & \\ & \ddots & & \ddots & \\ & & & & 1 \\ & & & 1 & \end{bmatrix} \boldsymbol{x}_n + \frac{1}{2}\boldsymbol{b}. \tag{5.2}$$

Even when d is large, the right-hand side is cheap to compute and the matrix–vector product is easy to evaluate *without* storing the sparse matrix.

Example 5.4 (Gauss–Seidel). Let $A_1 = D + L$ (the diagonal and lower-triangular part) and $A_2 = -U$. For the finite-difference matrix,

$$\underbrace{\begin{bmatrix} 2 & & & \\ -1 & 2 & & \\ & -1 & 2 & \\ & & \ddots & \ddots \end{bmatrix}}_{=A_1} \boldsymbol{x}_{n+1} = \underbrace{\begin{bmatrix} 0 & 1 & & \\ & \ddots & 1 & \\ & & & 1 \\ & & & \ddots \end{bmatrix}}_{=A_2} \boldsymbol{x}_k + \boldsymbol{b}$$

This time a linear solve is required. As the matrix on the left-hand side is lower triangular, this can be done efficiently to find $\boldsymbol{x}_{n+1}$.

We now develop some tools for understanding the convergence of iterative methods.

5.2 Vector and matrix norms

To understand convergence of $\boldsymbol{x}_n$, we introduce a way to measure distance on $\mathbb{R}^d$. It turns out to be very convenient to have more than one measurement of distance.

Definition 5.2 (vector norm)**.** A vector norm on $\mathbb{R}^d$ is a real-valued function $\|\cdot\|$ on $\mathbb{R}^d$ such that

(a) $\|\boldsymbol{x}\| \geq 0$ for all $\boldsymbol{x} \in \mathbb{R}^d$,

(b) $\|\boldsymbol{x}\| = 0$ if and only if $\boldsymbol{x} = \boldsymbol{0}$,

(c) $\|\alpha\boldsymbol{x}\| = |\alpha|\,\|\boldsymbol{x}\|$ for $\alpha \in \mathbb{R}$, and

(d) $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ (the triangle inequality).

The standard Euclidean norm $\|\boldsymbol{x}\|_2 := (\boldsymbol{x}^\mathsf{T}\boldsymbol{x})^{1/2} \equiv (x_1^2 + \cdots + x_d^2)^{1/2}$ satisfies these conditions and is a vector norm. The conditions (a–c) are easy to verify. The last one follows from the Cauchy–Schwarz inequality, which says that $\boldsymbol{x}^\mathsf{T}\boldsymbol{y} \leq \|\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2$ and so

$$\left\|\boldsymbol{x} + \boldsymbol{y}\right\|_2^2 = (\boldsymbol{x} + \boldsymbol{y})^\mathsf{T}(\boldsymbol{x} + \boldsymbol{y}) = \boldsymbol{x}^\mathsf{T}\boldsymbol{x} + 2\boldsymbol{x}^\mathsf{T}\boldsymbol{y} + \boldsymbol{y}^\mathsf{T}\boldsymbol{x} \leq \left\|\boldsymbol{x}\right\|_2^2 + 2\left\|\boldsymbol{x}\right\|_2\left\|\boldsymbol{y}\right\|_2 + \left\|\boldsymbol{y}\right\|_2^2$$
$$= \left(\left\|\boldsymbol{x}\right\|_2 + \left\|\boldsymbol{y}\right\|_2\right)^2.$$

We will make use of two more examples:

$$\left\|\boldsymbol{x}\right\|_\infty := \max_{j=1,\ldots,d}\left|x_j\right|, \qquad \left\|\boldsymbol{x}\right\|_1 := \sum_{j=1}^d\left|x_j\right|.$$

Don't forget the absolute-value signs on the right-hand side! Verification of the norm axioms is straightforward here.

These give different numbers and, for $\boldsymbol{x} = (-1, 1, 2)^\mathsf{T}$, we get

$$\left\|\boldsymbol{x}\right\|_2 = \sqrt{6}, \qquad \left\|\boldsymbol{x}\right\|_1 = 4, \qquad \left\|\boldsymbol{x}\right\|_\infty = 2.$$

We also need to measure matrices. The obvious way to do this is to treat a $d \times d$ matrix as a d^2 vector and thereby inherit the norms defined for vectors. This does not say anything about the multiplicative nature of matrices and so we develop the following concept.

Definition 5.3 (matrix norm)**.** A matrix norm $\|A\|$ of a $d \times d$ matrix A is a real-valued function on $\mathbb{R}^{d \times d}$ such that

(a) $\|A\| \geq 0$ for all $A \in \mathbb{R}^{d \times d}$,

(b) $\|A\| = 0$ if and only if $A = 0$,

(c) $\|\alpha A\| = |\alpha|\,\|A\|$ for $\alpha \in \mathbb{R}$,

(d) $\|A + B\| \leq \|A\| + \|B\|$ (the triangle inequality) and

(e) $\|AB\| \leq \|A\|\,\|B\|$ for all $A, B \in \mathbb{R}^{d \times d}$.

Conditions (a–d) correspond to the ones for vector norms. The last, the sub-multiplicative condition, relates to matrix products.

Vector norms lead naturally to a corresponding matrix norm, known as the subordinate or operator norm.

Definition 5.4. The operator norm $\|A\|_{\mathrm{op}}$ with respect to a vector norm $\|\boldsymbol{x}\|$ is defined by

$$\|A\|_{\mathrm{op}} := \sup_{\boldsymbol{x} \neq \boldsymbol{0}} \frac{\|A\boldsymbol{x}\|}{\|\boldsymbol{x}\|}.$$

Equivalently, because of condition (c),

$$\|A\|_{\mathrm{op}} = \sup_{\|\boldsymbol{x}\|=1} \|A\boldsymbol{x}\|$$

(sup means least upper bound or roughly "the maximum"). The operator norm describes the maximum stretch that can be achieved by multiplication by A.

Theorem 5.1. *The operator norm $\|A\|_{\mathrm{op}}$ is a matrix norm*

Proof. We show (e). As $\|A\|_{\mathrm{op}} = \sup\|A\boldsymbol{x}\|/\|\boldsymbol{x}\|$, we have

$$\|A\boldsymbol{x}\| \leq \|A\|_{\mathrm{op}}\|\boldsymbol{x}\| \tag{5.3}$$

for any $\boldsymbol{x} \in \mathbb{R}^d$. Then, applying Eq. (5.3) twice,

$$\|AB\boldsymbol{x}\| \leq \|A\|_{\mathrm{op}}\|B\boldsymbol{x}\| \leq \|A\|_{\mathrm{op}}\|B\|_{\mathrm{op}}\|\boldsymbol{x}\|.$$

Hence,

$$\|AB\|_{\mathrm{op}} = \max_{\|\boldsymbol{x}\|=1}\|AB\boldsymbol{x}\| \leq \|A\|_{\mathrm{op}}\|B\|_{\mathrm{op}}. \qquad \square$$

Let $\|A\|_1$ be the operator norm associated to the vector norm $\|\boldsymbol{x}\|_1$, and $\|A\|_\infty$ be the operator norm associated to the vector norm $\|\boldsymbol{x}\|_\infty$.

Theorem 5.2. *Let A be an $d \times d$ matrix. Then,*

$$\|A\|_1 = \max_{j=1,\ldots,d} \sum_{i=1}^d |a_{ij}|, \qquad \textit{maximum column sum}$$

$$\|A\|_\infty = \max_{i=1,\ldots,d} \sum_{j=1}^d |a_{ij}|, \qquad \textit{maximum row sum.}$$

To remember which way round it is, $\|A\|_1$ is the maximum column sum and the subscript 1 looks like a column. Don't forget the absolute-value signs!

Proof. We prove that

$$\|A\|_\infty = \max_i \sum_j |a_{ij}| =: f(A).$$

The argument for $\|A\|_1$ is similar and addressed on the problem sheet. We divide and conquer, first showing that $\|A\|_\infty \leq f(A)$ and then $\|A\|_\infty \geq f(A)$.

To show that $\|A\|_\infty \leq f(A)$, consider $\boldsymbol{x} \in \mathbb{R}^d$ with $\|\boldsymbol{x}\|_\infty = 1$. Then, $|x_i| \leq 1$ for all $i = 1, \ldots, d$ and hence

$$\left| \sum_{j=1}^d a_{ij} x_j \right| \leq \sum_{j=1}^d \left| a_{ij} x_j \right| \leq \sum_{j=1}^d |a_{ij}| \leq f(A).$$

That is, the ith entry of $A\boldsymbol{x}$ is smaller (in absolute value) than $f(A)$. Hence, $\|A\boldsymbol{x}\|_\infty \leq f(A)$.

To show that $\|A\|_\infty \geq f(A)$, suppose that $f(A) = \sum_j |a_{ij}|$ (i.e., row i has the maximum sum). Let

$$x_j := \begin{cases} +1, & a_{ij} \geq 0, \\ -1, & a_{ij} < 0. \end{cases}$$

Clearly then $\|\boldsymbol{x}\|_\infty = 1$ and

$$A\boldsymbol{x} = \begin{bmatrix} \times \\ \times \\ \vdots \\ \sum_j a_{ij} x_j \\ \times \end{bmatrix} = \begin{bmatrix} \times \\ \times \\ \vdots \\ \sum_j |a_{ij}| \\ \times \end{bmatrix} = \begin{bmatrix} \times \\ \times \\ \vdots \\ f(A) \\ \times \end{bmatrix},$$

where we write the ith row only. The magnitude of the largest entry of $A\boldsymbol{x}$ is at least $f(A)$ and

$$\left\| A\boldsymbol{x} \right\|_\infty \geq f(A). \qquad \square$$

Both these operator norms are very easy to compute.

Example 5.5. Let

$$A = \begin{bmatrix} 3 & -8 & -9 \\ 1 & -2 & 0 \\ 9 & -14 & 6 \end{bmatrix}.$$

Then $\|A\|_1 = \max\{13, 24, 15\} = 24$ and $\|A\|_\infty = \max\{20, 3, 29\} = 29$.

The matrix 2-norm $\|A\|_2$ induced by the Euclidean vector norm $\|\boldsymbol{x}\|_2$ is much harder to understand. We quote the following theorem:

Theorem 5.3. *Let A be a $d \times d$ matrix; then*

$$\left\| A \right\|_2 = \sqrt{\rho(A^\mathsf{T} A)}, \tag{5.4}$$

where $\rho(B)$ is the spectral radius or size of the largest eigenvalue defined by

$$\rho(B) := \max\{|\lambda| : \lambda \text{ is an eigenvalue of } B \text{ so that } B\boldsymbol{u} = \lambda\boldsymbol{u} \text{ for some } \boldsymbol{u} \neq \boldsymbol{0}\}.$$

When A is symmetric, we have simply that

$$\left\| A \right\|_2 = \rho(A).$$

The proof of Eq. (5.4) is not covered. In the symmetric case, $A = A^{\mathsf{T}}$ and, if λ is an eigenvalue of A, then λ^2 is an eigenvalue of $A^{\mathsf{T}}A = A^2$. We expect $\rho(A^{\mathsf{T}}A) = \rho(A)^2$.

Eigenvalues of large matrices are difficult to compute, though we can easily do it for a 2×2 example.

Example 5.6. Let

$$A = \begin{bmatrix} 3 & 1 \\ 0 & 1 \end{bmatrix}, \qquad A^{\mathsf{T}}A = \begin{bmatrix} 9 & 3 \\ 3 & 2 \end{bmatrix}.$$

The eigenvalues λ are the solutions of

$$\det(A - \lambda I) = \det \begin{vmatrix} 9 - \lambda & 3 \\ 3 & 2 - \lambda \end{vmatrix} = (9 - \lambda)(2 - \lambda) - 9 = \lambda^2 - 11\lambda + 9 = 0.$$

The quadratic equation formula gives $\lambda = (11 \pm \sqrt{121 - 36})/2$ and $\|A\|_2 = \rho(A) = \sqrt{(11 + \sqrt{85})/2}$.

Example 5.7. Recall the $d \times d$ finite-difference matrix

$$A = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots \\ & & & \end{bmatrix}.$$

We find the eigenvalues of A, which allows us to find $\|A\|_2$. Let $h := 1/(d+1)$ and

$$\boldsymbol{u}_k := (\sin k\pi h, \sin 2k\pi h, \ldots, \sin dk\pi h)^{\mathsf{T}} \in \mathbb{R}^d.$$

We show that $\boldsymbol{u}_k$ is an eigenvector of A. The jth component of $A\boldsymbol{u}_k$ is

$$(A\boldsymbol{u}_k)_j = 2\sin(jk\pi h) - \sin((j-1)k\pi h) - \sin((j+1)k\pi h),$$

where we use $\sin((j-1)k\pi h) = 0$ for $j = 1$ and $\sin((j+1)k\pi h) = \sin(k\pi) = 0$ for $j = d$.

The trig identity $\sin(X + Y) = \cos X \sin Y + \cos Y \sin X$ gives

$$\begin{aligned} (A\boldsymbol{u}_k)_j &= 2\sin(jk\pi h) - \big(\cos(k\pi h)\sin(jk\pi h) - \cos(jk\pi h)\sin(k\pi h)\big) \\ &\quad - \big(\cos(k\pi h)\sin(jk\pi h) + \cos(jk\pi h)\sin(k\pi h)\big) \\ &= 2(1 - \cos k\pi h)\sin(jk\pi h) \\ &= \lambda_k \times \text{the } j\text{th component of } \boldsymbol{u}_k, \end{aligned}$$

where $\lambda_k := 2(1 - \cos(k\pi h))$. In other words, λ_k is an eigenvalue of A with eigenvector $\boldsymbol{u}_k$. This gives d distinct eigenvalues for A. We conclude that

$$\rho(A) = \max\big\{\lambda_k : k = 1, \ldots, d\big\} = 2\Big(1 - \cos\frac{d\pi}{d+1}\Big).$$

As A is symmetric, Theorem 5.3 gives

$$\big\|A\big\|_2 = \sqrt{\rho(A^{\mathsf{T}}A)} = \rho(A) = 2\Big(1 - \cos\frac{d\pi}{d+1}\Big).$$

5.3 Convergence of iterative methods

If $A = D + L + U$ (sum of diagonal and lower- and upper-triangular parts), the Jacobi method is

$$D\boldsymbol{x}_{n+1} = -(L+U)\boldsymbol{x}_n + \boldsymbol{b}$$

and the Gauss–Seidel method is

$$(L+D)\boldsymbol{x}_{n+1} = -U\boldsymbol{x}_n + \boldsymbol{b}.$$

If D is non-singular, both can be written

$$\boldsymbol{x}_{n+1} = T\boldsymbol{x}_n + \boldsymbol{c}$$

where

$$
\begin{aligned}
T = T_{\mathrm{J}} &:= -D^{-1}(L+U), & \boldsymbol{c} &= D^{-1}\boldsymbol{b}, & &\text{Jacobi,}\\
T = T_{\mathrm{G}S} &:= -(L+D)^{-1}U, & \boldsymbol{c} &= (L+D)^{-1}\boldsymbol{b}, & &\text{Gauss–Seidel.}
\end{aligned}
$$

Lemma 5.1. *Suppose that A and D are non-singular. Consider $T = T_{\mathrm{J}}$ or $T = T_{\mathrm{GS}}$. Then $\boldsymbol{x}$ is the solution of $A\boldsymbol{x} = \boldsymbol{b}$ if and only if $\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{x}) := T\boldsymbol{x} + \boldsymbol{c}$ (i.e., $\boldsymbol{x}$ is a fixed point of $\boldsymbol{g}$; see Definition 4.1).*

Proof. Elementary. □

Theorem 5.4. *Suppose that the conditions of Lemma 5.1 hold, so that $A\boldsymbol{x} = \boldsymbol{b}$ has a unique solution $\boldsymbol{x}$. Suppose that $\boldsymbol{x}_n$ is a sequence in $\mathbb{R}^d$ generated by*

$$\boldsymbol{x}_{n+1} = T\boldsymbol{x}_n + \boldsymbol{c}, \tag{5.5}$$

for some initial vector $\boldsymbol{x}_0$. Then,

$$\left\|\boldsymbol{x}_n - \boldsymbol{x}\right\| \le \left\|T\right\|_{\mathrm{op}}^n \left\|\boldsymbol{x}_0 - \boldsymbol{x}\right\|,$$

where $\|T\|_{\mathrm{op}}$ is the operator norm associated to a vector norm $\|\boldsymbol{x}\|$.

Proof. We have $\boldsymbol{x}_{n+1} = T\boldsymbol{x} + \boldsymbol{c}$ and $\boldsymbol{x} = T\boldsymbol{x} + \boldsymbol{c}$; then

$$\boldsymbol{x}_{n+1} - \boldsymbol{x} = \left(T\boldsymbol{x}_n - T\boldsymbol{x}\right) + \left(\boldsymbol{c} - \boldsymbol{c}\right).$$

Apply the vector norm:
$$\left\|\boldsymbol{x}_{n+1} - \boldsymbol{x}\right\| = \left\|T\boldsymbol{x}_n - T\boldsymbol{x}\right\|.$$

Using Eq. (5.3),
$$\left\|\boldsymbol{x}_{n+1} - \boldsymbol{x}\right\| \le \left\|T\right\|_{\mathrm{op}} \left\|\boldsymbol{x}_n - \boldsymbol{x}\right\|.$$

As simple induction argument shows that $\|\boldsymbol{x}_n - \boldsymbol{x}\| \le \|T\|_{\mathrm{op}}^n \|\boldsymbol{x}_0 - \boldsymbol{x}\|$. □

Corollary 5.1. *If $\|T\|_{\mathrm{op}} < 1$, then $\boldsymbol{x}_n$ converges to $\boldsymbol{x}$ as $n \to \infty$. The convergence is linear (see Definition 4.2); that is, $\|\boldsymbol{x}_{n+1} - \boldsymbol{x}\| \le K\|\boldsymbol{x}_n - \boldsymbol{x}\|$ for $K = \|T\|_{\mathrm{op}} < 1$.*

When $\|T\|_{\mathrm{op}}$ is small, the convergence is more rapid and this is desirable in numerical calculations.

In finite dimensions, it turns out that all norms are equivalent and any operator norm can be used as long as $\|T\|_{\mathrm{op}} < 1$.

An alternative characterisation of convergence can be given in terms of eigenvalues.

Corollary 5.2. *The sequence $\boldsymbol{x}_n$ given by $\boldsymbol{x}_{n+1} = T\boldsymbol{x}_n + \boldsymbol{c}$ converges to the fixed point $\boldsymbol{x}$ satisfying $\boldsymbol{x} = T\boldsymbol{x} + \boldsymbol{c}$ for any $\boldsymbol{x}_0 \in \mathbb{R}^d$ if $\rho(T) < 1$.*

Proof. This is a corollary of Theorem 5.4 when T is symmetric as $\rho(T) = \|T\|_2$.

Suppose for simplicity that T has d distinct eigenvalues λ_j with corresponding eigenvectors $\boldsymbol{u}_j$, so that $T\boldsymbol{u}_j = \lambda_j \boldsymbol{u}_j$. Then, $\boldsymbol{u}_j$ is a basis for $\mathbb{R}^d$ and

$$\boldsymbol{x}_0 - \boldsymbol{x} = \sum_{j=1}^{d} \alpha_j \boldsymbol{u}_j,$$

for some $\alpha_j \in \mathbb{R}$. Let $\boldsymbol{e}_n = \boldsymbol{x}_n - \boldsymbol{x}$. Then, $\boldsymbol{e}_{n+1} = T\boldsymbol{e}_n$ and

$$\boldsymbol{e}_n = T^n \boldsymbol{e}_0 = T^n \sum_{j=1}^{d} \alpha_j \boldsymbol{u}_j = \sum_{j=1}^{d} \alpha_j T^n \boldsymbol{u}_j = \sum_{j=1}^{d} \alpha_j \lambda^n \boldsymbol{u}_j.$$

If $\rho(T) < 1$ then all eigenvalues λ satisfy $|\lambda| < 1$. Hence, $\lambda^n \to 0$ as $n \to \infty$. Thus $\boldsymbol{e}_n \to 0$ as $n \to \infty$ and the iterative method converges.

The case where the eigenvalues of T are not distinct is omitted. $\qquad\square$

Example 5.8. Consider
$$\begin{bmatrix} 8 & -1 & 0 \\ 1 & 5 & 2 \\ 0 & 2 & 4 \end{bmatrix} \boldsymbol{x} = \boldsymbol{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

The Jacobi iteration is

$$\begin{bmatrix} 8 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 4 \end{bmatrix} \boldsymbol{x}_{n+1} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & -2 \\ 0 & -2 & 0 \end{bmatrix} \boldsymbol{x}_n + \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

and

$$T_{\mathrm{J}} = \begin{bmatrix} 8 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 4 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & -2 \\ 0 & -2 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1/8 & 0 \\ -1/5 & 0 & -2/5 \\ 0 & -1/2 & 0 \end{bmatrix}.$$

Then,

$$\left\| T_J \right\|_{\infty} = \max\left\{ \frac{1}{8}, \frac{3}{5}, \frac{1}{2} \right\} = \frac{3}{5}, \qquad \text{max row sum}$$

$$\left\| T_J \right\|_1 = \max\left\{ \frac{1}{5}, \frac{5}{8}, \frac{2}{5} \right\} = \frac{5}{8}, \qquad \text{max column sum.}$$

The matrix norms are less than one and the Jacobi iteration converges for this example.

Example 5.9 (finite-difference matrix)**.** Recall the Jacobi iteration for the finite-difference matrix Eq. (5.2). Then, $\boldsymbol{x}_{n+1} = T\boldsymbol{x}_n + \boldsymbol{c}$ for

$$T = \frac{1}{2}\begin{bmatrix} 0 & 1 & & & \\ 1 & \ddots & 1 & & \\ & \ddots & & \ddots & \\ & & & & 1 \\ & & & 1 & 0 \end{bmatrix}, \qquad \boldsymbol{c} = \frac{1}{2}\boldsymbol{b}.$$

Note that $\|T\|_\infty = \|T\|_1 = 1$ so that Corollary 5.1 does not apply. We work out the eigenvalues similarly to Example 5.7. Let $\boldsymbol{u}_k = [\sin(k\pi h), \sin(2k\pi h), \ldots, \sin(dk\pi h)]^\mathsf{T}$ for $h = 1/(d+1)$. Then, the jth component of $T\boldsymbol{u}_k$ is $\frac{1}{2}(\sin(j-1)k\pi h) + \sin(j+1)k\pi h)$ and, by applying a trig identity, this is $\cos(k\pi h)\sin(jk\pi h)$. Thus, $\lambda_k = \cos(k\pi h)$ for $k = 1, \ldots, d$ are the eigenvalues of T and, as $|\lambda_k| < 1$, the convergence of the Jacobi iteration follows for the finite-difference matrix.

Note however that $\rho(T) = \max\{\cos(k\pi h)\colon k = 1, \ldots, d\} \to 1$ as $h \to 0$ as $\cos(\pi h) \approx 1$ for $h \approx 0$. This means when h is small the Jacobi iteration converges slowly. This is a significant problem in applications where small h corresponds to accurate resolution of the underlying physics. In other words, the more accurate our discretisation the slower the Jacobi iteration is.

For convergence of Gauss–Seidel with the finite-difference matrix, see Problem E9.3.

5.4   Condition number

We'd like to estimate the error in the approximation when solving a linear system. Our methods provide a computed value $\boldsymbol{x}_c$ that we hope approximates the solution $\boldsymbol{x}$ of $A\boldsymbol{x} = \boldsymbol{b}$. We cannot evaluate a norm for $\boldsymbol{x}_c - \boldsymbol{x}$ as $\boldsymbol{x}$ is usually unknown. What we do have is the residual defined by

$$\boldsymbol{r} := A\boldsymbol{x}_c - \boldsymbol{b}.$$

Note that

$$\boldsymbol{r} := A\boldsymbol{x}_c - A\boldsymbol{x} = A(\boldsymbol{x}_c - \boldsymbol{x})$$

and $\boldsymbol{x}_c - \boldsymbol{x} = A^{-1}\boldsymbol{r}$. Apply Eq. (5.3) to get

$$\left\|\boldsymbol{x}_c - \boldsymbol{x}\right\| \le \left\|A^{-1}\right\|_{\mathrm{op}}\left\|\boldsymbol{r}\right\|.$$

Furthermore, $A\boldsymbol{x} = \boldsymbol{b}$ gives

$$\left\|\boldsymbol{b}\right\| = \left\|A\boldsymbol{x}\right\| \le \left\|A\right\|_{\mathrm{op}}\left\|\boldsymbol{x}\right\|.$$

Dividing the two expressions,

$$\frac{\|\boldsymbol{x}_c - \boldsymbol{x}\|}{\|A\|_{\mathrm{op}}\|\boldsymbol{x}\|} \le \frac{\|A^{-1}\|_{\mathrm{op}}\|\boldsymbol{r}\|}{\|\boldsymbol{b}\|}.$$

Rearranging we get

$$\underbrace{\frac{\|\boldsymbol{x}_c - \boldsymbol{x}\|}{\|\boldsymbol{x}\|}}_{\text{relative error}} \le \underbrace{\left\|A^{-1}\right\|_{\mathrm{op}}\left\|A\right\|_{\mathrm{op}}}_{\text{condition number}} \underbrace{\frac{\|\boldsymbol{r}\|}{\|\boldsymbol{b}\|}}_{\text{relative residual}}.$$

Thus, the relative error is bounded by a constant times the relative residual. The constant is known as the *condition number*:

Definition 5.5 (condition number)**.** The condition number of a non-singular matrix A is defined by
$$\text{Cond}(A) := \left\| A \right\|_{\text{op}} \left\| A^{-1} \right\|_{\text{op}}.$$

Each choice of operator norm $\|A\|_{\text{op}}$ gives a different condition corresponding to the choice of vector norm $\|\boldsymbol{x}\|$ above and $\text{Cond}_1(A)$, $\text{Cond}_2(A)$, $\text{Cond}_\infty(A)$ denote the condition numbers with respect to the 1-, 2-, and ∞-norms. Often $\kappa(A)$ is used to denote the condition number.

The condition number is a widely used measure of the difficulty of solving $A\boldsymbol{x} = \boldsymbol{b}$. When $\text{Cond}(A)$ is large, it may be impossible to get accurate results.

Example 5.10. Returning to Eq. (1.1), we have

$$A = \begin{bmatrix} \epsilon & 1 \\ 0 & 1 \end{bmatrix}, \qquad A^{-1} = \frac{1}{\epsilon} \begin{bmatrix} 1 & -1 \\ 0 & \epsilon \end{bmatrix}.$$

Suppose that $0 < \epsilon < 1$. Then, $\|A\|_1 = 2$ and $\|A^{-1}\|_1 = (1 + \epsilon)/\epsilon$ and hence $\text{Cond}_1(A) = 2(1 + \epsilon)/\epsilon$. This is large when ϵ is small and the matrix is ill-conditioned. This reflects the sensitivity of $\boldsymbol{x}$ to changes in $\boldsymbol{b}$ found when solving $A\boldsymbol{x} = \boldsymbol{b}$ in (1.1).

Example 5.11 (Hilbert matrix)**.** Refer back to Problem E1.2 where we performed experiments with the Hilbert matrix A, which is the $d \times d$ matrix with (i, j) entry $1/(i + j - 1)$. We found that it was difficult to solve the linear system of equations $A\boldsymbol{x} = \boldsymbol{b}$ for a given $\boldsymbol{b} \in \mathbb{R}^d$ if d is moderately large (e.g., $d = 10$). `numpy.linalg` provides the function `cond` for finding the condition number. We apply this to the Hilbert matrix

```
>> d=4; A=hilb(d) % let A be the 4x4 Hilbert matrix

ans =

    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    0.1429

>> cond(A,1) % give condition number wrt 1-norm

ans=

2.8375e+04
```

Repeating this experiment for $d = 6$ we find a condition number of 2.9070×10^7; for $d = 8$, the condition number is 3.3872×10^{10}; for $d = 10$, the condition number is 3.534×10^{13}. Even for small systems (problem in real-world applications can be easily have millions of unknowns), the condition number is extremely large.

Second derivation of Cond(A)

Suppose that

$$A\boldsymbol{x} = \boldsymbol{b}, \qquad (A + \Delta A)\boldsymbol{x}_c = \boldsymbol{b},$$

where ΔA can be thought as the effects of rounding error. Then,

$$\boldsymbol{x} = A^{-1}\boldsymbol{b} = A^{-1}(A + \Delta A)\boldsymbol{x}_c = \boldsymbol{x}_c + A^{-1}\Delta A\boldsymbol{x}_c.$$

Hence,

$$\boldsymbol{x} - \boldsymbol{x}_c = A^{-1}\Delta A\boldsymbol{x}_c.$$

Applying norms, we find that

$$\left\|\boldsymbol{x} - \boldsymbol{x}_c\right\| \leq \left\|A^{-1}\right\|_{\text{op}}\left\|\Delta A\right\|_{\text{op}}\left\|\boldsymbol{x}_c\right\|.$$

We can rewrite this in terms of the condition number:

$$\frac{\left\|\boldsymbol{x} - \boldsymbol{x}_c\right\|}{\left\|\boldsymbol{x}_c\right\|} \leq \text{Cond}(A)\frac{\left\|\Delta A\right\|_{\text{op}}}{\left\|A\right\|_{\text{op}}}.$$

The relative error is less than the condition number times the size of the relative change in A.

Example 5.12 (finite-difference matrix)**.** Let A be the $d \times d$ finite-difference matrix of Example 5.2. Then, $\text{Cond}_2(A)$ is easy to find, because we know all the eigenvalues of A. The eigenvalues of A^{-1} are simply λ^{-1} where λ is an eigenvalue of A. So, as A is symmetric,

$$\text{Cond}_2(A) = \left\|A\right\|_2\left\|A^{-1}\right\|_2 = \rho(A)\rho(A^{-1}).$$

The eigenvalues are $\lambda_k = 2(1 - \cos(k\pi h))$. Note that λ_k increases from $\lambda_1 = 2(1 - \cos(\pi h))$ to $\lambda_d = 2(1 - \cos(d\pi/(d+1)))$. Then, $\rho(A) = 2(1 - \cos(d\pi/(d+1)))$ and $\rho(A^{-1}) = 1/(2(1 - \cos(h\pi)))$, and

$$\text{Cond}_2(A) = \frac{1 - \cos(d\pi/(d+1))}{1 - \cos(\pi h)} \to \infty, \qquad \text{as } h \downarrow 0.$$

In other words, the finite-difference matrix becomes more ill-conditioned as we make h small and approximate the second derivate accurately. This is a common problem when approximating PDEs numerically.