MA22037: Numerical Analysis

Department of Mathematical Sciences, University of Bath

2026-02-02

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. You will use Python to do this.

1.1 Examples

1.1.1 Catastrophic cancellation and the 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

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

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. Then in Python, typing `import numpy as np` then `np.pi`, we see

$$\pi \quad \text{is replaced by } 3.141592653589793.$$

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? 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.

1.1.2 Linear equations

To convince you that the previous example is not overly contrived, consider the linear system of equations:

$$\begin{bmatrix} \epsilon & 1 \\ 0 & 1 \end{bmatrix} \mathbf{x} = \mathbf{b}, \qquad \mathbf{b} := \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

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

$$\mathbf{x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Imagine that there has been rounding error and the vector $\mathbf{b}$ is actually stored as $[1 + \delta, 1]^T$ (the T denotes transpose):

$$\begin{bmatrix} \epsilon & 1 \\ 0 & 1 \end{bmatrix} \mathbf{x} = \mathbf{b}, \qquad \mathbf{b} := \begin{bmatrix} 1 + \delta \\ 1 \end{bmatrix}. \tag{1}$$

Again solving the linear system, we find

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

In the case that $0 < \epsilon \ll \delta$ (ϵ is much smaller than δ), there is a large change to the solution $\mathbf{x}$. This system and its solution $\mathbf{x}$ 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.

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

$$p_N(x_i) = f(x_i), \qquad i = 0, \ldots, N. \tag{2}$$

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

$$p_1(x) = f(x_0) + \left(\frac{f(x_1) - f(x_0)}{x_1 - x_0} \right) (x - x_0). \tag{3}$$

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 (3),

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

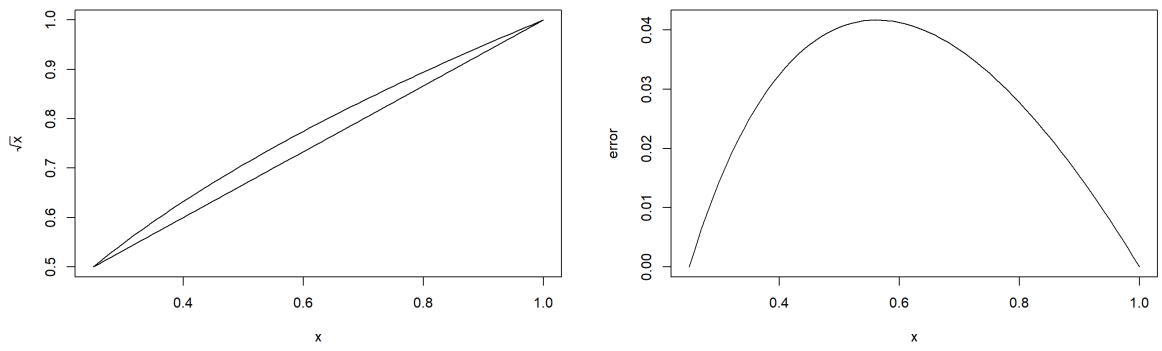

Figure 1: Linear interpolation of $f(x) = \sqrt{x}$ at $x_0 = 1/4$ and $x_1 = 1$

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.

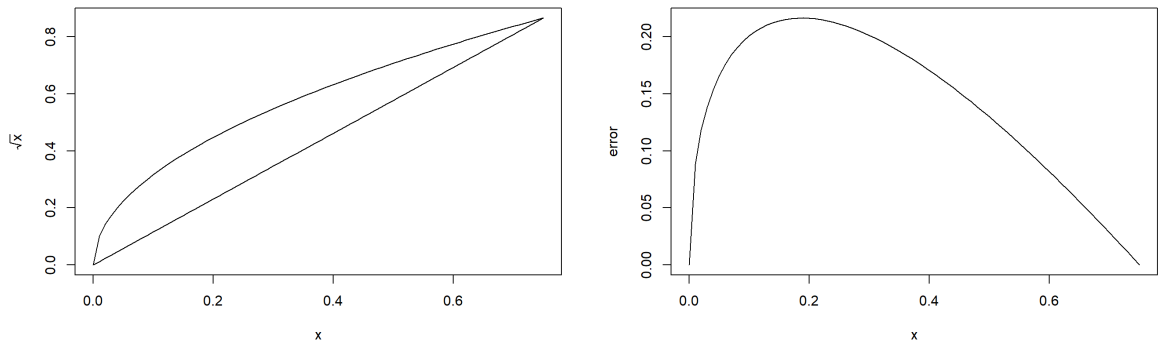

Figure 2: Linear interpolation of $f(x) = \sqrt{x}$ at $x_0 = 0$ and $x_1 = 3/4$

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 cause of the mean-value theorem using $f(a) = f(b)$. □

To analyse the *error in linear interpolation*, set

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

Note that $e(x_0) = 0 = e(x_1)$.

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

$$e(x) = \frac{f''(\xi)}{2} w_2(x), \quad \textit{where} \quad w_2(x) := (x - x_0)(x - x_1). \tag{4}$$

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

$$g(t) := e(t) - \frac{e(x)}{w_2(x)} w_2(t) \qquad \text{for } t \in [x_0, x_1]. \tag{5}$$

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

$$\exists \xi \in (\eta_1, \eta_2) \quad \text{ such that } g''(\xi) = 0. \tag{6}$$

By (5),

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

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

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

As $g''(\xi) = 0$ by (6),

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

and finally this can be rearranged to show that (4) holds. □

We note that ξ in (4) 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 $f : I \to \mathbb{R}$, we define the sup norm

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

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

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

$$\|e\|_{\infty,[x_0,x_1]} \le \frac{(x_1 - x_0)^2}{8} \|f''\|_{\infty,[x_0,x_1]}. \tag{7}$$

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

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

Also,

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

Simple calculus (see Problem E2.1) shows that

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

Hence,

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

This inequality also holds for $x = x_0$ and x_1 (since the left-hand side vanishes) and we have derived (7). □

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.

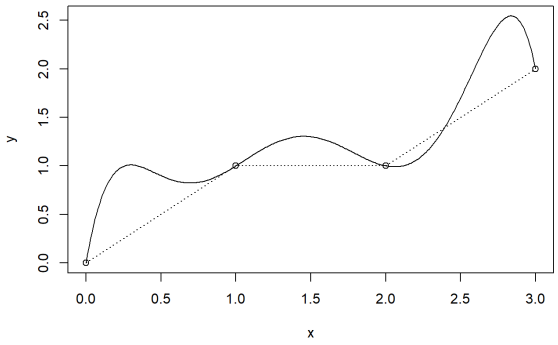

Figure 3: Piecewise-linear interpolation

2.1.1 Piecewise-linear interpolation

To get a (more) accurate approximation to $f : [a, b] \to \mathbb{R}$, 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,

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

Clearly,

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

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

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

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! It does not need to be twice continuously 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

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

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.

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

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

Hence, from (8),

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

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

$$p_N(x_i) = f(x_i), \qquad i = 0, \ldots, N. \tag{9}$$

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 $f : [a, b] \to \mathbb{R}$ 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 (9). 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. □

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 2.1 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. □

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.

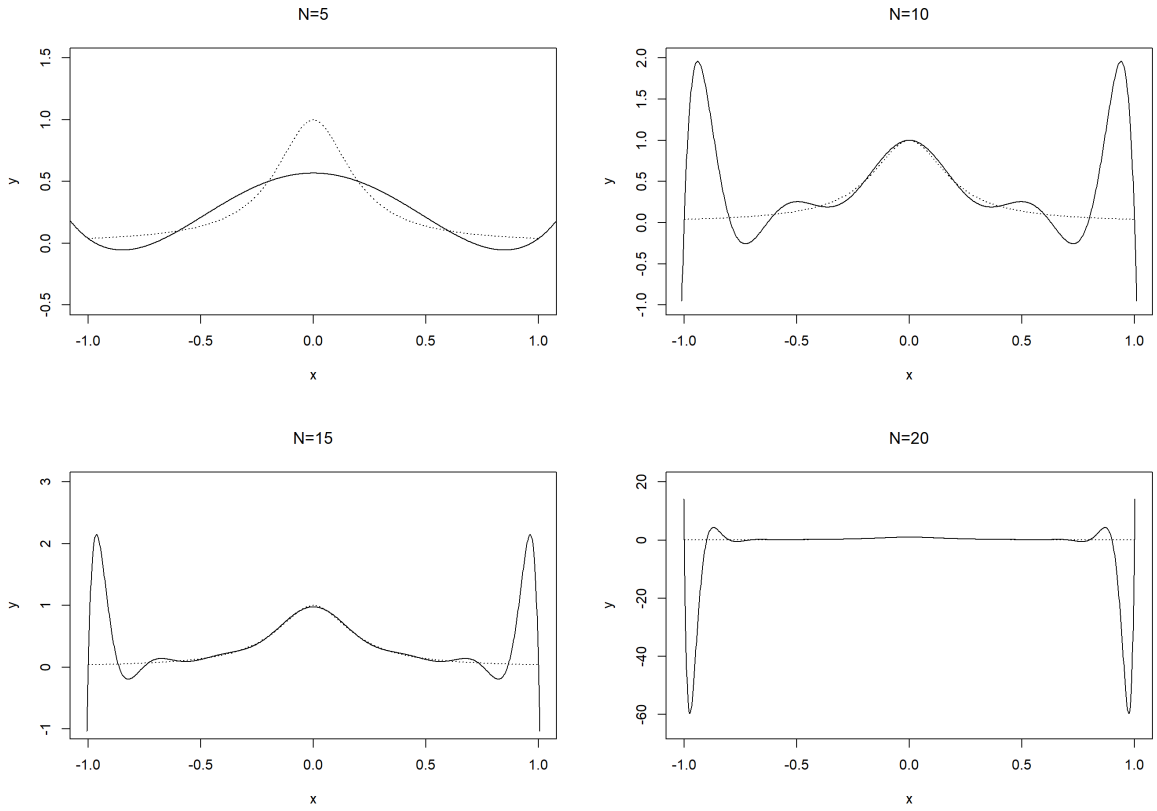

Figure 4: Runge's phenonmenon with the function $f(x) = 1/(25x^2 + 1)$

Here the solid lines show 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.

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 $f : \mathbb{R} \to \mathbb{R}$.

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 $p_1(x) = f(x_0) + f[x_0, x_1](x - x_0)$.

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 $x_3, f(x_3)$, 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 $f : [a, b] \to \mathbb{R}$, 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

$$\int_0^1 f(x)\,dx. \tag{10}$$

We approximate this by *quadrature rules* of the form

$$Q(f) = \sum_{i=0}^{N} w_i f(x_i) \tag{11}$$

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 (11) 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

3.1.1 Newton-Cotes rules 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 (11) 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 subsection 2), the linear interpolant to f is

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

and

$$\begin{aligned}\int_0^1 p_1(x)\,dx &= f(0) + (f(1) - f(0)) \int_0^1 x\,dx \\ &= f(0) + \frac{1}{2}(f(1) - f(0)) = \frac{1}{2}\left(f(0) + f(1)\right).\end{aligned}$$

We have derived the **trapezoidal** or **trapezium rule**

$$Q_1(f) := \frac{1}{2}\left(f(0) + f(1)\right) \quad \text{for approximating} \quad \int_0^1 f(x)dx. \tag{12}$$

Here the weights $w_0 = w_1 = \frac{1}{2}$. It is called the **trapezium 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).

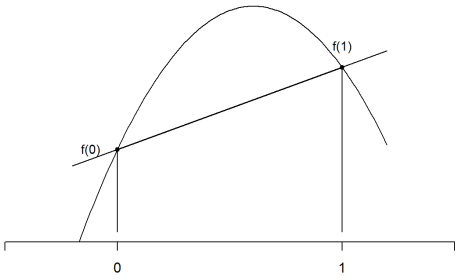

Figure 5: The trapezium rule

3.1.1.1 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,

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

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

$$\begin{aligned}\int_0^1 p_2(x)\,dx &= f(0) + \frac{1}{2}(f(1)-f(0)) + \frac{1}{3}\left(2f(1/2) - 2f(0) + f(0) - f(1)\right) \\ &= \frac{1}{6}\left[f(0) + 4f\left(\tfrac{1}{2}\right) + f(1)\right].\end{aligned}$$

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

$$Q_2(f) := \frac{1}{6}f(0) + \frac{4}{6}f\left(\tfrac{1}{2}\right) + \frac{1}{6}f(1), \tag{13}$$

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

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

3.1.2 Newton-Cotes rules over a general interval $[a, b]$

Suppose now we have a rule

$$Q(f) = \sum_{i=0}^{N} w_i f(x_i) \qquad \text{that approximates} \qquad \int_0^1 f(x)\, dx. \tag{14}$$

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

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

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

$$Q^{[a,b]}(f) := \sum_{i=0}^{N} w_i g(x_i) = (b-a) \sum_{i=0}^{N} w_i f\left(a + (b-a)x_i\right). \tag{15}$$

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

$$Q_1^{[a,b]}(f) = \frac{b-a}{2}\left(f(a) + f(b)\right), \qquad \text{trapezium rule}$$

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

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

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

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

The exact value

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

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 $|9/16 - 0.583333| = 2.083 \times 10^{-2}$.

3.1.3 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 subsection 3.1)

$$Q(f) = \sum_{i=0}^{N} w_i f(x_i) \quad \text{that approximates} \quad \int_0^1 f(x)\, dx, \tag{16}$$

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 (16) on each subinterval, to obtain the approximation

$$\int_a^b f(t)\, dt \approx \sum_{j=1}^{J} Q^{[y_{j-1},y_j]}(f) = \sum_{j=1}^{J} h_j \sum_{i=0}^{N} w_i f(y_{j-1} + h_j x_i), \tag{17}$$

where $h_j = y_j - y_{j-1}$. The approximation (17) is known as the **composite version** of (16).

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)**.**

$$Q_{1,J}^{[a,b]}(f) := \sum_{j=1}^{J} Q_1^{[y_{j-1},y_j]}(f) = \sum_{j=1}^{J} \frac{h_j}{2} \left(f(y_{j-1}) + f(y_j)\right). \tag{18}$$

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)**.**

$$Q_{2,J}^{[a,b]}(f) := \sum_{j=1}^{J} Q_2^{[y_{j-1},y_j]}(f) = \sum_{j=1}^{J} \frac{h_j}{6} \left(f(y_{j-1}) + 4f(m_j) + f(y_j)\right), \tag{19}$$

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 (18) and (19) in Python, so that we can apply the rules on any mesh.

3.2 Error analysis

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

Given a rule

$$Q(f) := \sum_{i=0}^{N} w_i f(x_i) \tag{20}$$

that approximates

$$I(f) := \int_0^1 f(x)\, dx,$$

we define the error in the quadrature rule to be

$$E(f) := I(f) - Q(f)\,. \tag{21}$$

Definition 3.1 (DoP)**.** The rule $Q(f)$ in (20) has **degree of precision (DoP)** $d \in \mathbb{N}$ if

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

Example 3.4 (trapezium rule DoP)**.** The trapezium rule $Q_1(f) = \frac{1}{2}(f(0) + f(1))$, and the error $E_1(f) := I(f) - Q_1(f)$. We create the following table:

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 DoP)**.** Simpson's rule $Q_2(f) = \frac{1}{6}\left[f(0) + 4f\left(\frac{1}{2}\right) + f(1)\right]$, and the error $E_2(f) := I(f) - Q_2(f)$. We create the following table:

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

Hence the DoP of Simpson's rule is 3.

The trapezium rule Q_1 is found by integrating p_1 and it is not surprising that its DoP is 1. Similarly, Q_2 is found by integrating p_2 and thus 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* (20) *has DoP* d*, then*

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

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

$$I(\alpha f_1 + \beta f_2) \;=\; \alpha I(f_2) + \beta I(f_2) \quad \text{and} \quad Q(\alpha f_1 + \beta f_2) \;=\; \alpha Q(f_2) + \beta Q(f_2), \tag{22}$$

since I and Q are both linear transformations.

If $p \in \mathcal{P}_d$, p is a polynomial of degree less than 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 (22),

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

since the DoP equals d. □

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

$$E_1(f) = \left[\frac{E_1(x^2)}{2!} \right] f''(\xi) = -\frac{1}{12} f''(\xi), \qquad \text{for all } f \in C^2[0,1]. \tag{23}$$

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 trapezium rule depends on the second derivative. Here $C^k[a,b]$ is the set of functions $f : [a,b] \to \mathbb{R}$ 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. 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 (15) 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 (23),

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

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.

3.2.2 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 subsection 3.1.3):

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

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 trapezium and Simpson's rule.

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

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

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

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

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

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

$$\begin{aligned} E_{1,J}^{[0,1]}(f) =& E_{1,J}^{[0,y_1]}(f) + E_{1,J}^{[y_1,1]}(f) \\ =& \left(\int_0^{y_1} f(x)\,dx - h_1\frac{\sqrt{y_1}}{2}\right) - \frac{1}{12}\sum_{j=2}^{J} h_j^3 f^{(2)}(\eta_j)\,. \end{aligned} \tag{24}$$

Now, we can estimate each of the terms in (24) separately (see Problem E5.2).

3.3 Gaussian quadrature rules

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:

$$Q^{[-1,1]}_{\text{Gauss},N}(f) := \int_{-1}^{1} p_N(x)\,dx \tag{25}$$

as an approximation to

$$\int_{-1}^{1} f(x)\,dx. \tag{26}$$

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

$$Q(1) = \int_{-1}^{1} 1\,dx = 2 \qquad \text{and} \qquad Q(x) = \int_{-1}^{1} x\,dx = 0. \tag{27}$$

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 (27). 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

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

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 $w_0x_0+w_1x_1 = 0$ and $w_0x_0^2+w_1x_1^2 = 2/3$ and $w_0x_0^3+w_1x_0^3 = 0$. 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**

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

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

Remark. 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 Solving nonlinear equations

4.1 Root finding

Root finding problem. For a given function $f : \mathbb{R} \to \mathbb{R}$, 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 bisubsection 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 bisubsection method is simple to apply but does not generalise easily to higher dimension. Instead, we focus on fixed-point iterations.

4.2 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 $g : [a, b] \to \mathbb{R}$ 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.
- $g_2(x) = \frac{1}{2}\left(10 - x^3\right)^{1/2}$ 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 condition (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 condition (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 fixed point theorem, we use the mean-value theorem from MA12001.

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

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

We now prove Theorem 4.1:

Proof. 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 $[a, b]$. $\square$

4.3 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 $g(x) = x$.

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

$$g'(x) = 1 - \frac{f'(x)}{f'(x)} + \frac{f(x)f''(x)}{f'(x)^2} = 0 \tag{28}$$

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. □

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

- case $r = 1$ (linear convergence): $e_{n+1} \leq K e_n$ for all $n \in \mathbb{N}$, for some $K < 1$;
- case $r > 1$: $e_{n+1} \leq K e_n^r$ 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 MA12001:

Theorem 4.4 (Taylor's theorem)**.** *Suppose that $f : (a, b) \to \mathbb{R}$ is $(m + 1)$-times continuously differentiable and suppose $x_0, x \in (a, b)$ with $x_0 \neq x$. 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 m-th 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 $K > 0$,*

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

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 (28) 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

$$|x_{n+1} - x| \leq K|x_n - x|^2,$$

thus concluding the proof. □

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 the Python code `x = x - np.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).

5 Numerical linear algebra

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

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

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 $\mathbf{x}$ 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 $A\vec{x} = \vec{b}$:

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

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

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

and

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

We have replaced the problem of solve $A\vec{x} = \vec{b}$ 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 $\vec{x}$ occurs as the limit of approximations $\vec{x}_n$ 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 $A\vec{x}$. 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,

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

Then,

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

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 $f : [0, 1] \to \mathbb{R}$, find $u(x)$ such that

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

Using the finite-difference approximation,

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

for

$$A := \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 2 \end{bmatrix}. \tag{29}$$

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

$$f := \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_d) \end{bmatrix}, \quad \vec{u} := \begin{bmatrix} u(x_1) \\ u(x_2) \\ \vdots \\ u(x_d) \end{bmatrix}. \tag{30}$$

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 $A\vec{x} = \vec{b}$ for $\vec{x} \in \mathbb{R}^d$ given a $d \times d$ matrix A and $\vec{b} \in \mathbb{R}^d$. Write $A = A_1 - A_2$ so that

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

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

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

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

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

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

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

or

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

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

For the finite-difference example (29), the Jacobi method is

$$\vec{x}_{n+1} = \frac{1}{2}\begin{bmatrix} 0 & 1 & & & \\ 1 & \ddots & 1 & & \\ & \ddots & & \ddots & \\ & & & & 1 \\ & & & 1 & \end{bmatrix} \vec{x}_n + \frac{1}{2}\vec{b}. \tag{31}$$

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,

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

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 $\vec{x}_{n+1}$.

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

5.2 Vector and matrix norms

To understand convergence of $\vec{x}_n$, 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) $\|\vec{x}\| \geq 0$ for all $\vec{x} \in \mathbb{R}^d$,
b) $\|\vec{x}\| = 0$ if and only if $\vec{x} = \vec{0}$,
c) $\|\alpha\vec{x}\| = |\alpha| \, \|\vec{x}\|$ for $\alpha \in \mathbb{R}$, and
d) $\|\vec{x} + \vec{y}\| \leq \|\vec{x}\| + \|\vec{y}\|$ for all $\vec{x}, \vec{y} \in \mathbb{R}^d$ (the triangle inequality).

The standard Euclidean norm $\|\vec{x}\|_2 := \left(\vec{x}^{\mathsf{T}}\vec{x}\right)^{1/2} \equiv \left(x_1^2 + \cdots + x_d^2\right)^{1/2}$ 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 $\vec{x}^{\mathsf{T}}\vec{y} \le \|\vec{x}\|_2\|\vec{y}\|_2$ and so

$$\begin{aligned}\|\vec{x}+\vec{y}\|_2^2 &= (\vec{x}+\vec{y})^{\mathsf{T}}(\vec{x}+\vec{y}) \\ &= \vec{x}^{\mathsf{T}}\vec{x} + 2\vec{x}^{\mathsf{T}}\vec{y} + \vec{y}^{\mathsf{T}}\vec{x} \\ &\le \|\vec{x}\|_2^2 + 2\|\vec{x}\|_2\|\vec{y}\|_2 + \|\vec{y}\|_2^2 \\ &= \Big(\|\vec{x}\|_2 + \|\vec{y}|_2\Big)^2.\end{aligned}$$

We will make use of two more examples:

$$\|\vec{x}\|_\infty := \max_{j=1,\dots,d} |x_j|, \qquad \|\vec{x}\|_1 := \sum_{j=1}^{d} |x_j|.$$

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 $\vec{x} = (-1,1,2)^{\mathsf{T}}$, we get

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

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) $\|A\| \ge 0$ 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) $\|A+B\| \le \|A\| + \|B\|$ (the triangle inequality) and
e) $\|AB\| \le \|A\|\,\|B\|$ 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 $\|A\|_{\text{op}}$ with respect to a vector norm $\|\vec{x}\|$ is defined by

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

Equivalently, because of condition (c),

$$\|A\|_{\text{op}} = \sup_{\|\vec{x}\|=1} \|A\vec{x}\|$$

(sup means least upper bound or roughly "the maximum". However, since the hypersphere $\{\vec{x} : \|\vec{x}\| = 1\}$ is *compact*, the supremum in this case can be replaced by a maximum.).

The operator norm describes the maximum stretch that can be achieved by multiplication by A.

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

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

$$\|A\vec{x}\| \leq \|A\|_{\text{op}}\|\vec{x}\| \tag{32}$$

for any $\vec{x} \in \mathbb{R}^d$. Then, applying (32) twice,

$$\|AB\vec{x}\| \leq \|A\|_{\text{op}}\|B\vec{x}\| \leq \|A\|_{\text{op}}\|B\|_{\text{op}}\|\vec{x}\|.$$

Hence,

$$\|AB\|_{\text{op}} = \sup_{\|\vec{x}\|=1} \|AB\vec{x}\| \leq \|A\|_{\text{op}}\|B\|_{\text{op}}.$$

The remaining conditions are left for a problem sheet. □

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

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 $\vec{x} \in \mathbb{R}^d$ with $\|\vec{x}\|_\infty = 1$. Then, $|x_i| \leq 1$ for all $i = 1, \ldots, d$ and hence

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

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

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 $\|\vec{x}\|_\infty = 1$ and

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

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

$$\|A\vec{x}\|_\infty \geq f(A),$$

completing the proof. □

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 $\|\vec{x}\|_2$ is much harder to understand. We quote the following theorem:

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

$$\|A\|_2 = \sqrt{\rho(A^\mathsf{T}A)}, \tag{33}$$

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

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

When A is symmetric, we have simply that

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

The proof of (33) 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

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

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

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

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

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

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

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

As A is symmetric, Theorem 5.3 gives

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

5.3 Convergence of iterative methods

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

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

and the Gauss-Seidel method is

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

If D is non-singular, both can be written

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

where

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

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

Proof. Elementary. □

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

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

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

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

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

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

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

Apply the vector norm:

$$\|\vec{x}_{n+1} - \vec{x}\| = \|T\vec{x}_n - T\vec{x}\|.$$

Using (32),

$$\|\vec{x}_{n+1} - \vec{x}\| \le \|T\|_{\mathrm{op}} \, \|\vec{x}_n - \vec{x}\|.$$

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

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

When $\|T\|_{\text{op}}$ 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 $\|T\|_{\text{op}} < 1$.

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

Corollary 5.2. *The sequence $\vec{x}_n$ given by $\vec{x}_{n+1} = T\vec{x}_n + \vec{c}$ converges to the fixed point $\vec{x}$ satisfying $\vec{x} = T\vec{x} + \vec{c}$ for any $\vec{x}_0 \in \mathbb{R}^d$ 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 $\vec{u}_j$, so that $T\vec{u}_j = \lambda_j\vec{u}_j$. Then, $\vec{u}_j$ is a basis for $\mathbb{R}^d$ and

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

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

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

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

The case where the eigenvalues of T are not distinct is omitted. □

Example 5.8. Consider

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

The Jacobi iteration is

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

and

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

Then,

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

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

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 (31). Then, $\vec{x}_{n+1} = T\vec{x}_n + \vec{c}$ for

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

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 $\vec{u}_k = [\sin(k\pi h), \sin(2k\pi h), \ldots, \sin(dk\pi h)]^\mathsf{T}$ for $h = 1/(d+1)$. Then, the jth component of $T\vec{u}_k$ 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 $\rho(T) = \max\{\cos(k\pi h) : k = 1, \ldots, d\} \to 1$ 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 Sheets.

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 $\vec{x}_c$ that we hope approximates the solution $\vec{x}$ of $A\vec{x} = \vec{b}$. We cannot evaluate a norm for $\vec{x}_c - \vec{x}$ as $\vec{x}$ is usually unknown. What we do have is the residual defined by

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

Note that

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

and $\vec{x}_c - \vec{x} = A^{-1}\vec{r}$. Apply (32) to get

$$\|\vec{x}_c - \vec{x}\| \le \|A^{-1}\|_{\text{op}}\|\vec{r}\|.$$

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

$$\|\vec{b}\| = \|A\vec{x}\| \leq \|A\|_{\text{op}} \|\vec{x}\|.$$

Dividing the two expressions,

$$\frac{\|\vec{x}_c - \vec{x}\|}{\|A\|_{\text{op}} \|\vec{x}\|} \leq \frac{\|A^{-1}\|_{\text{op}} \|\vec{r}\|}{\|\vec{b}\|}.$$

Rearranging we get

$$\underbrace{\frac{\|\vec{x}_c - \vec{x}\|}{\|\vec{x}\|}}_{\text{relative error}} \leq \underbrace{\|A^{-1}\|_{\text{op}} \|A\|_{\text{op}}}_{\text{condition number}} \underbrace{\frac{\|\vec{r}\|}{\|\vec{b}\|}}_{\text{relative residual}}.$$

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

$$\text{Cond}(A) := \|A\|_{\text{op}} \|A^{-1}\|_{\text{op}}.$$

Each choice of operator norm $\|A\|_{\text{op}}$ gives a different condition corresponding to the choice of vector norm $\|\vec{x}\|$ 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 $A\vec{x} = \vec{b}$. When $\text{Cond}(A)$ is large, it may be impossible to get accurate results.

Example 5.10. Returning to (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 $\vec{x}$ to changes in $\vec{b}$ found when solving $A\vec{x} = \vec{b}$ in (1).

Example 5.11 (Hilbert matrix)**.** Refer back to Problem Sheet 1 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 $A\vec{x} = \vec{b}$ for a given $\vec{b} \in \mathbb{R}^d$ if d is moderately large (e.g., $d = 10$). In Python, `numpy.linalg` provides the command `cond` for finding the condition number. We apply this to the Hilbert matrix

```
from numpy.linalg import cond
from scipy.linalg import hilbert
d = 4
A = hilbert(d)
cond_A = cond(A,p=1)
print(A, cond_A)
```

The output is

```
[[1.         0.5        0.33333333 0.25      ]
 [0.5        0.33333333 0.25       0.2       ]
 [0.33333333 0.25       0.2        0.16666667]
 [0.25       0.2        0.16666667 0.14285714]]

28374.99999999729
```

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.

5.4.1 Second derivation of $\mathrm{cond}(A)$

Suppose that

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

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

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

Hence,

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

Applying norms, we find that

$$\|\vec{x} - \vec{x}_c\| \le \|A^{-1}\|_{\mathrm{op}}\|\Delta A\|_{\mathrm{op}}\|\vec{x}_c\|.$$

We can rewrite this in terms of the condition number:

$$\frac{\|\vec{x} - \vec{x}_c\|}{\|\vec{x}_c\|} \le \mathrm{Cond}(A)\frac{\|\Delta A\|_{\mathrm{op}}}{\|A\|_{\mathrm{op}}}.$$

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, $\mathrm{Cond}_2(A)$ 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,

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

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

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

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

5.5 Beyond square matrices

5.5.1 Least squares problems

Suppose we have a more general linear system,

$$A\vec{x} = \vec{b}, \tag{34}$$

where $A \in \mathbb{R}^{m\times n}$, $\vec{x} \in \mathbb{R}^n$ and $\vec{b} \in \mathbb{R}^m$. In general, the system (34) will have no solutions if $m > n$ (i.e. it is *overdetermined*) or infinitely many if $m < n$ (i.e. it is *underdetermined*).

Systems of the form (34) frequently occur when we collect m observations (which can be prone to measurement error) and wish to describe them through an n-variable linear model. In statistics, where we typically have $n \ll m$, this is called **linear regression**.

Definition 5.6 (least squares problem)**.** Given $A \in \mathbb{R}^{m\times n}$ and $\vec{b} \in \mathbb{R}^m$, a vector $\vec{x} \in \mathbb{R}^n$ solves the *least squares problem* if it minimises $\|A\vec{x} - \vec{b}\|_2$.

Fortunately, we can solve the least squares problem by considering a specific $n \times n$ linear system!

Theorem 5.5. *The vector $\vec{x} \in \mathbb{R}^n$ is a solution of the least squares problem if and only if*

$$A^{\mathsf{T}}\Big(A\vec{x} - \vec{b}\Big) = 0.$$

Proof. If $\vec{x} \in \mathbb{R}^n$ is a solution of the least squares problem then it minimises

$$f(\vec{x}) := \|A\vec{x} - \vec{b}\|_2^2 = \Big\langle A\vec{x} - \vec{b}, A\vec{x} - \vec{b}\Big\rangle = \vec{x}^{\mathsf{T}}A^{\mathsf{T}}A\,\vec{x} - 2\vec{x}^{\mathsf{T}}A^{\mathsf{T}}\vec{b} + \vec{b}^{\mathsf{T}}\vec{b}\,.$$

Since f is smooth, it follows that $\nabla f(\vec{x}) = 0$. Computing the gradient of f gives

$$\nabla f(\vec{x}) = 2\Big(A^{\mathsf{T}}A\,\vec{x} - A^{\mathsf{T}}\vec{b}\Big),$$

and thus $A^{\mathsf{T}}\Big(A\vec{x} - \vec{b}\Big) = 0$.

Conversely, suppose $A^{\mathsf{T}}\big(A\vec{x}-\vec{b}\big)=0$ and let $\vec{u}\in\mathbb{R}^n$. Letting $\vec{y}:=\vec{u}-\vec{x}$, we have

$$\begin{aligned}\|A\vec{u}-\vec{b}\|_2^2 &= \|A\vec{x}+A\vec{y}-\vec{b}\,\|_2^2\\ &= \langle A\vec{x}-\vec{b}+A\vec{y}, A\vec{x}-\vec{b}+A\vec{y}\rangle\\ &= \langle A\vec{x}-\vec{b}, A\vec{x}-\vec{b}\rangle + 2\vec{y}^{\mathsf{T}}\underbrace{A^{\mathsf{T}}(A\vec{x}-\vec{y})}_{=0}+\langle A\vec{y}, A\vec{y}\rangle\\ &= \|A\vec{x}-\vec{b}\|_2^2+\|A\vec{y}\|_2^2 \geq \|A\vec{x}-\vec{b}\|_2^2\,.\end{aligned}$$

Thus, $\vec{u}$ minimises the left-hand side at $\vec{x}$. □

Therefore, we can solve the least squares problem simply by solving the **normal equations**:

$$\big(A^{\mathsf{T}}A\big)\vec{x}=A^{\mathsf{T}}\,\vec{b}. \tag{35}$$

Definition 5.7. The matrix $A^{\mathsf{T}}A\in\mathbb{R}^{n\times n}$ appear in the normal equations is often called the **Gram matrix**.

Example 5.13. Consider the least-squares problem

$$\min_{\vec{x}\in\mathbb{R}^2}\|A\vec{x}-\vec{b}\|_2\,,\quad \text{where}\quad A=\begin{bmatrix}1&2\\3&4\\5&6\end{bmatrix},\quad \vec{b}=\begin{bmatrix}7\\8\\9\end{bmatrix}.$$

Then we can find $\vec{x}$ by solving

$$\begin{aligned}A^{\mathsf{T}}A\vec{x}=A^{\mathsf{T}}\vec{b} &\iff \begin{bmatrix}1&3&5\\2&4&6\end{bmatrix}\begin{bmatrix}1&2\\3&4\\5&6\end{bmatrix}\vec{x}=\begin{bmatrix}1&3&5\\2&4&6\end{bmatrix}\begin{bmatrix}7\\8\\9\end{bmatrix}\\ &\iff \begin{bmatrix}35&44\\44&56\end{bmatrix}\vec{x}=\begin{bmatrix}76\\100\end{bmatrix}\\ &\iff \vec{x}=\begin{bmatrix}-6\\6.5\end{bmatrix}.\end{aligned}$$

However, this can come with potential disadvantages:

- $A^{\mathsf{T}}A$ may be singular or ill-conditioned, e.g.

$$\begin{bmatrix}a&b\end{bmatrix}^{\mathsf{T}}\begin{bmatrix}a&b\end{bmatrix}=\begin{bmatrix}a^2&ab\\ab&b^2\end{bmatrix},$$

$$\begin{bmatrix}\epsilon&1\\0&1\end{bmatrix}^{\mathsf{T}}\begin{bmatrix}\epsilon&1\\0&1\end{bmatrix}=\begin{bmatrix}\epsilon^2&\epsilon\\\epsilon&2\end{bmatrix}.$$

- $A^\mathsf{T}A$ may encounter rounding errors which A does not, e.g.

$$A = \begin{bmatrix} 10^8 & -10^8 \\ 1 & 1 \end{bmatrix} \implies A^\mathsf{T}A = \begin{bmatrix} 10^{16}+1 & -10^{16}+1 \\ -10^{16}+1 & 10^{16}+1 \end{bmatrix} \approx 10^{16} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

- In practice, a sparse matrix A usually results in sparse $A^\mathsf{T}A$. However, this is **not guaranteed** (e.g. if A contains a dense row).

Therefore, an alternative is to consider a slightly modified least squares problem.

Theorem 5.6. *Let $\Omega \in \mathbb{R}^{m\times m}$ be an arbitrary orthogonal matrix. That is, $\Omega^\mathsf{T}\Omega = I = \Omega\Omega^\mathsf{T}$. Then the vector $\vec{x} \in \mathbb{R}^n$ is a solution of the least squares problem if and only if it minimises*

$$\left\|\Omega A\vec{x} - \Omega\vec{b}\right\|_2.$$

Proof. The result follows as

$$\left\|\Omega A\vec{x} - \Omega\vec{b}\right\|_2^2 = \left(\Omega A\vec{x} - \Omega\vec{b}\right)^\mathsf{T}\left(\Omega A\vec{x} - \Omega\vec{b}\right) = (A\vec{x} - \vec{b})\,\underbrace{\Omega^\mathsf{T}\Omega}_{=I}\,(A\vec{x} - \vec{b}) = \|A\vec{x} - \vec{b}\|_2^2.$$

. □

Therefore, inspired by the previous theorem, we need to find a good choice for Ω.

Theorem 5.7 (QR decomposition)**.** *Let $A \in \mathbb{R}^{m\times n}$ with $m \geq n$. Then, there exist an orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ and an upper triangular matrix $R \in \mathbb{R}^{n\times n}$ such that*

$$A = Q\widetilde{R},$$

where $\widetilde{R} := \begin{bmatrix} R \\ 0 \end{bmatrix} \in \mathbb{R}^{m\times n}$.

Proof. Not covered, though QR decompositions can be computed using the Gram-Schmidt procedure. □

Therefore, if R is invertible (i.e. has non-zero diagonal entries), then we can solve the least squares problem as

1. Factorise $A = Q\widetilde{R}$.
2. Set $\Omega = Q^\mathsf{T}$, and note that the new least squares problem given by Theorem 5.6 becomes

$$\min_{\vec{x}\in\mathbb{R}^n} \left\| \Omega A\vec{x} - \Omega\vec{b} \right\|_2 = \min_{\vec{x}\in\mathbb{R}^n} \left\| \underbrace{Q^\mathsf{T}Q}_{=I} \widetilde{R}\vec{x} - Q^\mathsf{T}\vec{b} \right\|_2 = \min_{\vec{x}\in\mathbb{R}^n} \left\| \widetilde{R}\vec{x} - Q^\mathsf{T}\vec{b} \right\|_2.$$

3. Solve $R\vec{x} = \vec{c}$, where $\vec{c} \in \mathbb{R}^n$ is defined as the first n entries of $Q^\intercal \vec{b}$. Since R is upper triangular, this can be done efficiently (just like for the Gauss-Seidel method).

.

Two important questions beyond the scope of this unit

- How are QR decompositions actually computed?
- What should we do if the matrix R is singular?

Unsurprisingly, addressing either question for large matrices can be challenging!

For further details regarding the QR decomposition, we refer to johnwlambert.github.io/least-squares. Alternatively, you can choose "Numerical Linear Algebra" (MA32065) next year!

5.5.2 The singular value decomposition (SVD)

In addition to encoding linear transformations, matrices can also represent data in a wide variety of scientific applications. For example,

- **Images** – which are simply matrices of pixels. For a grayscale image, each pixel is just a single number, taking integer values between 0 and 255, which denote the 256 shades of gray (with 0 being full black, and 255 being full white). Colour images are slightly more complicated with each pixel consisting of three colours (red, green and blue). Whilst essential to the entertainment industry, images are also used to represent spatial data in STEM fields. However, processing, storing and extracting information from images (particularly high resolution ones) can be challenging. This naturally leads to **image compression**.

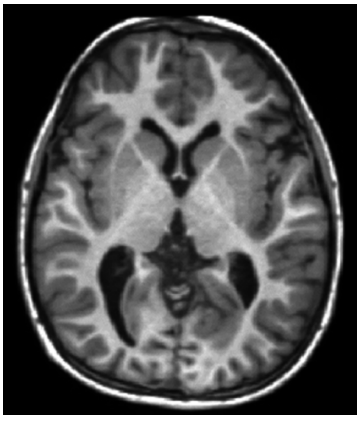

Figure 6: Magnetic Resonance Imaging (MRI) is an important application of image data in healthcare.

- **Datasets of vectors**. In practice, data is usually stored as vectors $\vec{x} \in \mathbb{R}^m$. For example, the following graph illustrates data with $m = 5$ (corresponding to "GDP per capita", "life expectancy", "name of country", "population of country" and "date").

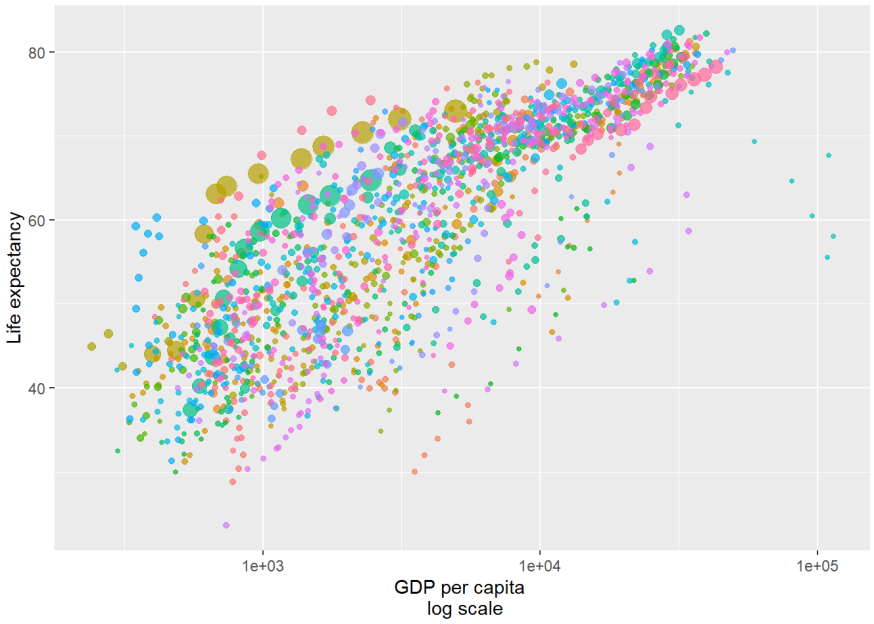

Figure 7: GDP data for different countries (each with its own colour) across time. Each dot represents a country on a specific date with the dot's size corresponding to its population.

Therefore, we can represent a dataset of n vectors $\{\vec{x}_1, \cdots, \vec{x}_n\}$ simply as an $m \times n$ matrix,

$$X := \begin{bmatrix} \vec{x}_1 & \vec{x}_2 & \cdots & \vec{x}_n \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{2,1} & \cdots & x_{n,1} \\ x_{1,2} & x_{2,2} & \cdots & x_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1,m} & x_{2,m} & \cdots & x_{n,m} \end{bmatrix}.$$

However, just as with images, it can be difficult to extract the key information from X (particularly if m and n are large). Thus, we are interested in **dimensionality reduction** and **data compression** more generally.

In this subsubsection, we will introduce the singular value decomposition (SVD) – which is widely considered to be the most important topic in numerical linear algebra. The SVD can be applied to any matrix, square or rectangular, and is the most prominent technique for matrix compression. Most notably, the SVD is used throughout data science for performing Principal Component Analysis (PCA).

Since the SVD has strong connections to symmetric eigenvalue problems, let us first recall the following theorem:

Theorem 5.8 (Symmetric eigenvalue decomposition)**.** *Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Then A admits the following matrix decomposition:*

$$A = V\Lambda V^{\mathsf{T}}, \tag{36}$$

where $V \in \mathbb{R}^{n\times n}$ is orthogonal, $V^{\mathsf{T}}V = I = VV^{\mathsf{T}}$, and $\Lambda = \operatorname{diag}(\lambda_1, \cdots, \lambda_n)$ is a diagonal matrix of eigenvalues $\lambda_i \in \mathbb{R}$.

In (36), λ_i are the eigenvalues and V is the matrix of eigenvectors (that is, its columns are eigenvectors).

It is worth noting that Theorem 5.8 is making two very strong claims. Namely, (a) the eigenvectors can be taken to be orthogonal and (b) the eigenvalues are real.

The decompoition (36) is not unique. We can always flip the sign of eigenvectors and if $\lambda_i = \lambda_j$ for some $i \neq j$, then the eigenvectors will span a subspace (whose dimension is the multiplicity of the eigenvalue). For example, any vector is an eigenvector of the identity matrix.

Before proceeding to our main theorem, we first recall the following key definitions.

Definition 5.8. The **column rank** of a matrix $A = \begin{bmatrix}\vec{a}_1 & \cdots & \vec{a}_n\end{bmatrix} \in \mathbb{R}^{m\times n}$ is the dimension of its column space $\operatorname{span}(\{\vec{a}_1, \cdots, \vec{a}_n\})$. Similarly, we can define the **row rank** of A as the dimension of its row space. It is a theorem that these ranks coincide and can thus unambiguously be called the **rank** of A.

Definition 5.9. A matrix $A \in \mathbb{R}^{m\times n}$ is said to have **full rank** if its rank equals $\min(n, m)$.

We now present one of the most important results in numerical linear algebra – the Singular Value Decomposition. Note: we will only be proving this when A has full rank.

Theorem 5.9 (Singular Value Decomposition (SVD))**.** *Let $A \in \mathbb{R}^{m\times n}$ with $m \geq n$. Then there exists $U \in \mathbb{R}^{m\times m}$, $\Sigma \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{n\times n}$ such that*

$$A = U \begin{bmatrix}\Sigma \\ 0\end{bmatrix} V^{\mathsf{T}}, \tag{37}$$

where U is orthogonal (i.e. $U^{\mathsf{T}}U = UU^{\mathsf{T}} = I_m$), V is orthogonal (i.e. $V^{\mathsf{T}}V = VV^{\mathsf{T}} = I_n$) and $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$.

Remark. The SVD easily applies to $m \times n$ matrices with $m < n$ simply by taking the transpose of (37). In a slight change of notation, it is more common to write the SVD (37) as

$$A = U\Sigma V^{\mathsf{T}}, \tag{38}$$

where $\Sigma \in \mathbb{R}^{m\times n}$ is still referred to as a **diagonal matrix**. However, the notation in (38) is much better suited for our proof.

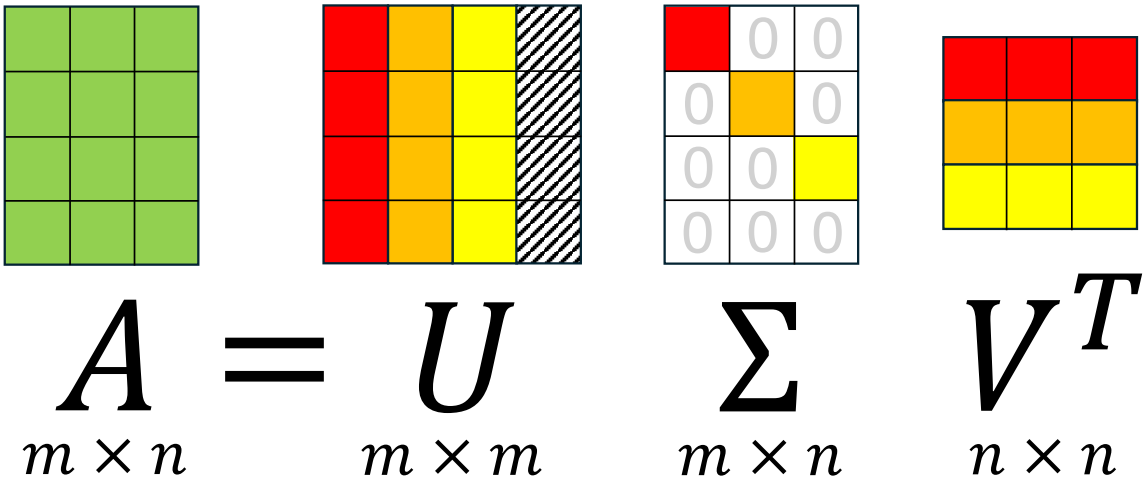

Figure 8: Visualisation of SVD. The hatched region corresponds to $U_\perp$ in our proof.

Proof. (when A has full rank). Without loss of generality, we will assume that $m \geq n$. This because if we had $m < n$, then it would suffice to find an SVD for A^T instead of A.

Since $A^\mathsf{T}A$ is symmetric, it admits an eigenvalue decomposition:

$$A^\mathsf{T}A = V\Lambda V^\mathsf{T},$$

where V is orthogonal.

Since A is assumed to have full rank, the null space of A is $\{0\}$. Therefore, for an eigenvector $\vec{x}$ and eigenvalue λ of $A^\mathsf{T}A$, we have $A\vec{x} \neq 0$ and

$$\|A\vec{x}\|_2^2 = (A\vec{x})^\mathsf{T}(A\vec{x}) = \vec{x}^\mathsf{T}A^\mathsf{T}A\vec{x} = \lambda\|\vec{x}\|_2^2\,,$$

which implies that

$$\lambda = \frac{\|A\vec{x}\|_2^2}{\|\vec{x}\|_2^2} > 0.$$

Hence, the diagonal of Λ must be positive – so we can write it as $\Lambda = \Sigma^2$ where Σ is a positive diagonal matrix. Therefore, Σ^{-1} exists and we can define the matrix $U_\top := AV\Sigma^{-1}$. This has orthogonal columns as

$$U_\top^\mathsf{T}U_\top = \Sigma^{-1}V^\mathsf{T}(A^\mathsf{T}A)V\Sigma^{-1} = \Sigma^{-1}(V^\mathsf{T}V)\Lambda(V^\mathsf{T}V)\Sigma^{-1} = I.$$

Since the n columns of $U_\top$ form an orthogonal basis for a subspace of $\mathbb{R}^m$, we can use the Gram-Schmidt procedure to extend them to an orthogonal basis of $\mathbb{R}^m$. Therefore, we can construct a matrix $U_\perp \in \mathbb{R}^{m\times(m-n)}$ whose columns are these additional basis vectors.

We can now define the orthogonal matrix $U := \begin{bmatrix} U_\top & U_\perp \end{bmatrix}$ and obtain the result as $U_\top\Sigma V^\mathsf{T} = (AV\Sigma^{-1})\Sigma V^\mathsf{T} = A$. □

Remark. The matrix $U_\perp$ is often called the **orthogonal complement** of $U_\top$. However, since it is multiplied by zeroes in the SVD (37), it has very little importance compared to $U_\top$.

Finally, for the terminology "Singular Value Decomposition" to make sense, we will define the "Singular Values" of a matrix.

Definition 5.10 (Singular values and vectors)**.** In Theorem 5.9, $\{\sigma_i\}$ are called the **singular values** of A. The columns of $U = \begin{bmatrix}\vec{u}_1 & \cdots & \vec{u}_m\end{bmatrix}$ are called the **left singular vectors** and the columns of $V = \begin{bmatrix}\vec{v}_1 & \cdots & \vec{v}_n\end{bmatrix}$ are called the **right singular vectors**.

Unsurprisingly, singular values are closely related to eigenvalues.

Theorem 5.10. *Let $\{\sigma_i\}$ denote the singular values of $A \in \mathbb{R}^{m\times n}$. Then*

$$\sigma_i = \sqrt{\lambda_i}$$

where $\{\lambda_i\}$ are the eigenvalues of $A^\mathsf{T}A$ with $|\lambda_1| \geq |\lambda_2| \geq \cdots |\lambda_n| \geq 0$. Furthermore,

- *$\{\lambda_i\}$ are the eigenvalues of AA^T,*
- *$V = \begin{bmatrix}\vec{v}_1 & \cdots & \vec{v}_n\end{bmatrix}$ are the eigenvectors of $A^\mathsf{T}A$,*
- *$U = \begin{bmatrix}\vec{u}_1 & \cdots & \vec{u}_m\end{bmatrix}$ are the eigenvectors of AA^T.*

Proof. From the Singular Value Decomposition (38), we have

$$A = U\Sigma V^\mathsf{T}.$$

Therefore, the matrix $A^\mathsf{T}A$ is

$$A^\mathsf{T}A = (U\Sigma V^\mathsf{T})^\mathsf{T}(U\Sigma V^\mathsf{T}) = V\Sigma \underbrace{U^\mathsf{T}U}_{=I_m}\Sigma V^\mathsf{T} = V\Sigma^2 V^\mathsf{T},$$

which is precisely the eigenvalue decomposition of $A^\mathsf{T}A$. Therefore Σ^2 are the eigenvalues of AA^T and U gives the eigenvectors. An identical argument applies to AA^T and gives the result. □

The SVD tells us that any matrix can be written as orthogonal-diagonal-orthogonal. Roughly speaking, orthogonal matrices can be thought of as rotations or reflection, so the SVD says the action of a matrix can be thought of as a rotation/reflection followed by magnification (or shrinkage), followed by another rotation/reflection.

With this intuition, we would expect that the rank of A corresponds to rank of Σ.

Theorem 5.11. *The rank of A is equal to the number of positive singular values, $\sigma_i > 0$.*

Proof. We first note that, if $A^\mathsf{T}A\vec{x} = 0$ where $\vec{x} \in \mathbb{R}^n$, then

$$\|A\vec{x}\|_2^2 = (A\vec{x})^\mathsf{T}(A\vec{x}) = \vec{x}^\mathsf{T}(A^\mathsf{T}A\vec{x}) = 0.$$

Therefore $A^\mathsf{T}A$ and A have the same null space. So by the rank-nullity theorem, they must have the same rank. From the SVD, we have

$$A^\mathsf{T}A = (V\Sigma^\mathsf{T}U^\mathsf{T})(U\Sigma V^\mathsf{T}) = V(\Sigma^\mathsf{T}\Sigma)V^\mathsf{T}.$$

The result now follows as the rank of the square matrix $A^\mathsf{T}A$ equals the number of non-zero eigenvalues in $\Sigma^\mathsf{T}\Sigma$. □

Having covered quite a bit of theory, let's consider an example.

Example 5.14. We would like to compute the SVD of the matrix $A = \begin{bmatrix} -1 & -2 \\ 2 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$.

Then the Gram Matrix is $A^\mathsf{T}A = \begin{bmatrix} -1 & 2 & 1 & 0 \\ -2 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & -2 \\ 2 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 6 & 4 \\ 4 & 6 \end{bmatrix}$.

We can compute its eigenvalues as 2 and 10 using the characteristic polynomial

$$\det\left(\begin{bmatrix} 6-\lambda & 4 \\ 4 & 6-\lambda \end{bmatrix} \right) = 0 \Leftrightarrow (6-\lambda)^2 = 16.$$

Solving each eigenvalue problem yields the following eigenvector matrix,

$$V = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}.$$

Therefore $A^\mathsf{T}A$ admits the eigenvalue decomposition,

$$A^\mathsf{T}A = V\Sigma^2V^\mathsf{T}, \quad \text{where} \quad \Sigma^2 = \begin{bmatrix} 10 & 0 \\ 0 & 2 \end{bmatrix}.$$

Hence $U_\top$ can be computed as

$$\begin{aligned} U_\top = AV\Sigma^{-1} &= \begin{bmatrix} -1 & -2 \\ 2 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{10}} & 0 \\ 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \\ &= \begin{bmatrix} -1 & -2 \\ 2 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{20}} & -\frac{1}{2} \\ \frac{1}{\sqrt{20}} & \frac{1}{2} \end{bmatrix} \\ &= \begin{bmatrix} -\frac{3}{\sqrt{20}} & -\frac{1}{2} \\ \frac{3}{\sqrt{20}} & -\frac{1}{2} \\ \frac{1}{\sqrt{20}} & -\frac{1}{2} \\ \frac{1}{\sqrt{20}} & \frac{1}{2} \end{bmatrix}. \end{aligned}$$

Therefore, we have the following singular value decomposition for A,

$$\begin{bmatrix} -1 & -2 \\ 2 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \left[\begin{array}{cc|c} -\frac{3}{\sqrt{20}} & -\frac{1}{2} & \\ \frac{3}{\sqrt{20}} & -\frac{1}{2} & \\ \frac{1}{\sqrt{20}} & -\frac{1}{2} & U_\perp \\ \frac{1}{\sqrt{20}} & \frac{1}{2} & \end{array}\right] \begin{bmatrix} \sqrt{10} & 0 \\ 0 & \sqrt{2} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix},$$

where the matrix $U_\perp$ can be computed through the Gram-Schmidt procedure.

5.5.3 Low-rank approximation using the SVD

From the SVD (37) and Theorem 5.11, we obtain the following important alternative formula:

$$A = \sum_{i=1}^{\text{rank}(A)} \sigma_i \vec{u}_i \vec{v}_i^{\mathsf{T}}, \tag{39}$$

where $\{\sigma_i\}$ are the singular values, $U = \begin{bmatrix} \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_n \end{bmatrix}$, $V = \begin{bmatrix} \vec{v}_1 & \vec{v}_2 & \cdots & \vec{v}_n \end{bmatrix}$ and rank(A) is precisely the number of non-zero σ_i.

Letting R denote the rank of A, we can visualise (39) as

$$A = \underbrace{\begin{bmatrix} * \\ * \\ \vdots \\ * \\ * \end{bmatrix} \begin{bmatrix} * & * & \cdots & * & * \end{bmatrix}}_{\sigma_1 \vec{u}_1 \vec{v}_1^{\mathsf{T}}} + \underbrace{\begin{bmatrix} * \\ * \\ \vdots \\ * \\ * \end{bmatrix} \begin{bmatrix} * & * & \cdots & * & * \end{bmatrix}}_{\sigma_2 \vec{u}_2 \vec{v}_2^{\mathsf{T}}} + \cdots + \underbrace{\begin{bmatrix} * \\ * \\ \vdots \\ * \\ * \end{bmatrix} \begin{bmatrix} * & * & \cdots & * & * \end{bmatrix}}_{\sigma_R \vec{u}_R \vec{v}_R^{\mathsf{T}}}.$$

Equation (39) reveals something quite interesting. It is possible to store the $m \times n$ matrix A using only $\text{rank}(A)(m+n+1)$ numbers (since each triple $(\sigma_i, \vec{u}_i, \vec{v}_i)$ requires $m+n+1$ numbers). As previously discussed, large matrices are ubiquitous throughout the sciences. So there is great interest in **low-rank** matrix approximation.

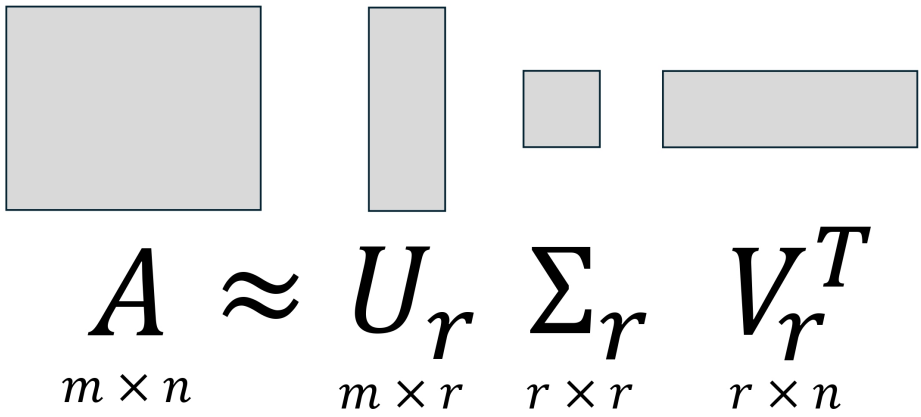

Figure 9: Visualisation of the low-rank approximation obtained by truncating SVD.

Definition 5.11 (Low-rank matrix approximation)**.** Given a matrix $A \in \mathbb{R}^{m\times n}$ with SVD (39), we define a rank-r approximation A_r as

$$A_r := \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^{\mathsf{T}}. \tag{40}$$

Alternatively, we can write A_r as

$$A_r = U_r \Sigma_r V_r^{\mathsf{T}},$$

where $U_r = \begin{bmatrix} \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_r \end{bmatrix} \in \mathbb{R}^{m\times r}$, $V_r = \begin{bmatrix} \vec{v}_1 & \vec{v}_2 & \cdots & \vec{v}_r \end{bmatrix} \in \mathbb{R}^{n\times r}$ and $\Sigma_r = \text{diag}(\sigma_1, \cdots, \sigma_r)$.

Furthermore, due to the remarkable properties of SVD, matrix approximation via (40) is actually optimal in the following sense:

Theorem 5.12 (Optimality of matrix approximation using SVD)**.** *For any $A \in \mathbb{R}^{m\times n}$ with singular values $\sigma_1(A) \geq \sigma_2(A) \geq \cdots \geq \sigma_{\min(m,n)}(A) \geq 0$, and any non-negative integer $r < \min(m,n)$, we have*

$$\|A - A_r\|_2 = \sigma_{r+1}(A) = \min_{\text{rank}(B)\leq r} \|A - B\|_2 , \tag{41}$$

where A_r is the rank-r approximation given by (40)*.*

Proof. Not covered. □

From this theorem, we can make a few observations:

- A good approximation $A \approx A_r$ is achieved if and only if $\sigma_{r+1} \ll \sigma_1$.
- Optimality actually holds under any norm that is invariant under multiplication by orthogonal matrices. We will not prove this optimality, but we can see that the Euclidean norm is invariant as

$$\|V\vec{x}\|_2^2 = (V\vec{x})^\mathsf{T}(V\vec{x}) = \vec{x}^\mathsf{T}\underbrace{V^\mathsf{T}V}_{=I}\vec{x} = \|\vec{x}\|_2^2\,, \tag{42}$$

  for any orthogonal matrix $V \in \mathbb{R}^{n\times n}$ and vector $\vec{x} \in \mathbb{R}^n$. In particular, the above optimality theorem also holds under the Frobenius norm.

Definition 5.12. The **Frobenius norm** of a matrix $A \in \mathbb{R}^{m\times n}$ is

$$\|A\|_F := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} = \sqrt{\mathrm{tr}\big(A^\mathsf{T}A\big)}.$$

- A prominent application of low-rank approximation is PCA (Principal Component Analysis), which is often used in data science.
- By viewing a grayscale image as a matrix $A \in \mathbb{R}^{m\times n}$, we can compute its SVD and visualise the resulting low-rank approximations. Whilst this is not the state of the art for image compression, it clearly demonstrates the effectiveness of SVD.

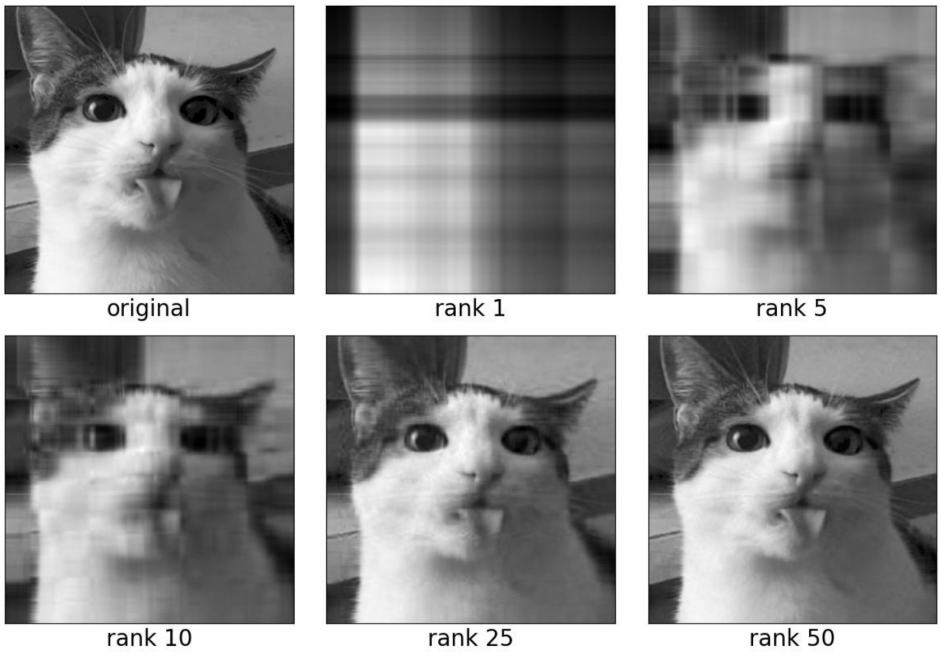

Figure 10: Image compression by low-rank approximation via the truncated SVD.

Although the proof of Theorem 5.12 is not covered, we will prove the following corollary.

Theorem 5.13. *Let $A \in \mathbb{R}^{m\times n}$. Then the first singular value of A is*

$$\sigma_1 = \|A\|_2 \,. \tag{43}$$

Proof. Without loss of generality, suppose $m \geq n$.

Let $\vec{x} \in \mathbb{R}^n$ be such that $\|x\|_2 = 1$ and define $\vec{y} := V^\mathsf{T}\vec{x}$. Then, since the Euclidean norm is invariant to multiplication by orthogonal matrices (see (42)), we have $\|y\|_2 = 1$ and

$$\|A\vec{x}\|_2 = \|U\widetilde{\Sigma}V^\mathsf{T}\vec{x}\|_2 = \|U\widetilde{\Sigma}\vec{y}\|_2 = \|\widetilde{\Sigma}y\|_2 \,,$$

where $\widetilde{\Sigma} := \begin{bmatrix}\Sigma \\ 0\end{bmatrix}$. Since Σ is an $n \times n$ diagonal matrix, this gives

$$\|A\vec{x}\|_2 = \sqrt{\sum_{i=1}^{n} \sigma_i^2 y_i^2} \leq \sqrt{\sum_{i=1}^{n} \sigma_1^2 y_i^2} = \sigma_1 \|\vec{y}\|_2 = \sigma_1 \|\vec{x}\|_2 \,.$$

Therefore, $\|A\|_2 \leq \sigma_1$. On the other hand, taking $\vec{x} = \vec{v}_1$ (the leading right singular vector), we obtain $V^\mathsf{T}\vec{v}_1 = e_1$ and thus $\|A\vec{v}_1\|_2 = \sigma_1$. □

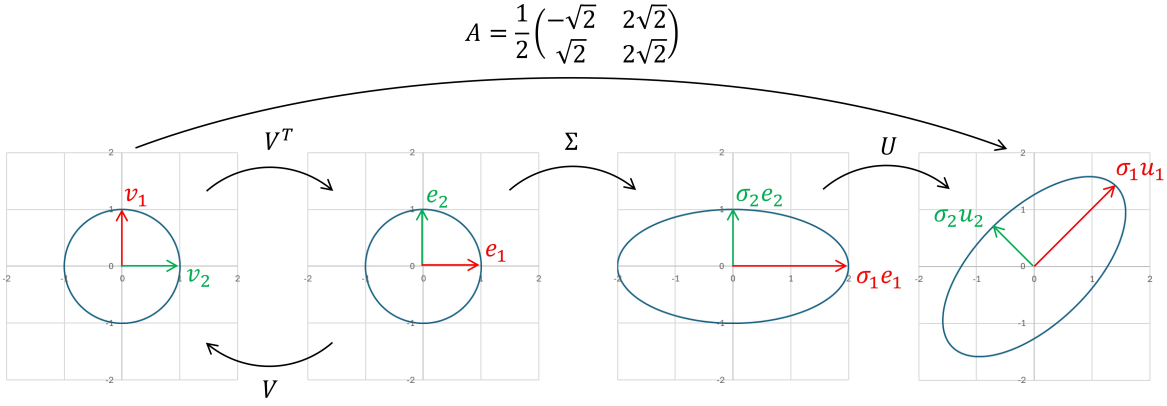

Figure 11: A geometric illustration for the SVD of a square matrix. In particular, we see that the leading singular value is the major semi-axis of the ellipse.

Finally, to conclude this subsection, we will discuss one of the most commonly used applications of SVD – **dimensionality reduction**.

That is, given data $X = \{\vec{x}_1, \cdots, \vec{x}_n\} \subset \mathbb{R}^m$, we aim to compress it into a lower-dimensional space $\mathbb{R}^r$ with $r < m$ (ideally $r \ll m$). Perhaps the most common way to do this is to find a matrix $W \in \mathbb{R}^{r\times m}$ that can reduce any $\vec{x} \in X$ to $\vec{y} := W\vec{x} \in \mathbb{R}^r$. Many computations can then be performed with the reduced vectors $\vec{y}$ directly. However, to recover (or more precisely, *approximate*) the original data $\vec{z} \approx \vec{x}$, we need another matrix $U \in \mathbb{R}^{m\times r}$ such that $\vec{z} = U\vec{y}$.

Definition 5.13. Given a dataset $X = \{\vec{x}_1, \cdots, \vec{x}_n\} \subset \mathbb{R}^m$, the **Principal Component Analysis (PCA)** finds a compression matrix $W_* \in \mathbb{R}^{r\times m}$ and a recovery matrix $U_* \in \mathbb{R}^{m\times r}$ such that the mean squared error betweeen the recovered and original vectors in X is minimal,

$$W_*, U_* = \underset{W\in\mathbb{R}^{r\times m}, U\in\mathbb{R}^{m\times r}}{\arg\min} \left(\frac{1}{n}\sum_{i=1}^{n} \left\|\vec{x}_i - UW\vec{x}_i\right\|_2^2\right). \tag{44}$$

The columns of U_* are called **Principal Components**.

Using the SVD, it is remarkably easy to solve the optimisation problem (44).

Theorem 5.14. *The PCA problem* (44) *can be solved by taking U_* as the truncated matrix $U_r = \begin{bmatrix}\vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_r\end{bmatrix} \in \mathbb{R}^{m\times r}$ from the SVD of X and $W_* = U_*^{\mathsf{T}}$.*

Proof. (non-examinable)

Let $\vec{y} := W\vec{x} \in \mathbb{R}^r$ and $Y = \begin{bmatrix}\vec{y}_1 & \cdots & \vec{y}_n\end{bmatrix} \in \mathbb{R}^{r\times n}$, then

$$\begin{aligned}
\sum_{i=1}^{n} \left\|\vec{x}_i - UW\vec{x}_i\right\|_2^2 &= \sum_{i=1}^{n} \left\|\vec{x}_i - U\vec{y}_i\right\|_2^2 \\
&= \sum_{i=1}^{n} \left(\vec{x}_i - U\vec{y}_i\right)^{\mathsf{T}} \left(\vec{x}_i - U\vec{y}_i\right) \\
&= \operatorname{tr}\Big(\big(X - UY\big)^{\mathsf{T}}\big(X - UY\big)\Big) \\
&= \left\|X - UY\right\|_F^2 .
\end{aligned}$$

As $\operatorname{span}(\{\vec{y}_1, \cdots, \vec{y}_n\})$ is a subspace of $\mathbb{R}^r$, it is as most r-dimensional. This means that $\{U\vec{y} : \vec{y} \in \operatorname{span}(\{\vec{y}_1, \cdots, \vec{y}_n\})\}$ must also be at most r-dimensional. However, since this is the column space of UY, it follows that $\operatorname{rank}(UY) \le r$.

So by the optimality of SVD, Theorem 5.12 (under the Frobenius norm), we see that the mean-squared error (44) is minimised when

$$UY = X_r\,, \tag{45}$$

with X_r denoting the rank-r approximation of X given by

$$X_r = U_r \Sigma_r V_r^{\mathsf{T}},$$

where $U_r = \begin{bmatrix}\vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_r\end{bmatrix} \in \mathbb{R}^{m\times r}$, $V_r = \begin{bmatrix}\vec{v}_1 & \vec{v}_2 & \cdots & \vec{v}_r\end{bmatrix} \in \mathbb{R}^{n\times r}$ and $\Sigma_r = \operatorname{diag}(\sigma_1, \cdots, \sigma_r)$ are obtained from the SVD of X.

We claim that (45) is achieved by setting $U = U_r$ and $W = U_r^\mathsf{T}$. This can be shown by direct calculation,

$$\begin{aligned} UWX &= U_r U_r^\mathsf{T} X \\ &= U_r U_r^\mathsf{T} X_r + U_r U_r^\mathsf{T}(X - X_r) \\ &= U_r \underbrace{U_r^\mathsf{T} U_r}_{=I_m} \Sigma_r V_r^\mathsf{T} + U_r U_r^\mathsf{T}(X - X_r) \\ &= X_r + U_r U_r^\mathsf{T}(X - X_r). \end{aligned}$$

Using the SVD formula (40) for X and X_r, we can express their difference $X - X_r$ as

$$X - X_r = \sum_{i \geq r+1} \sigma_i \vec{u}_i \vec{v}_i^\mathsf{T}.$$

Therefore, for any $j \leq r$, left multiplying by $\vec{u}_j^\mathsf{T}$ gives

$$\vec{u}_j^\mathsf{T}(X - X_r) = \sum_{i \geq r+1} \sigma_i \Big(\underbrace{\vec{u}_j^\mathsf{T} \vec{u}_i}_{=0} \Big) \vec{v}_i^\mathsf{T} = 0,$$

as $\vec{u}_j$ and $\vec{u}_i$ are orthogonal. Hence $U_r^\mathsf{T}(X - X_r) = 0$ and

$$UW = X_r + U_r U_r^\mathsf{T}(X - X_r) = X_r\,,$$

as required. □

Remark. From Theorem 5.14, we see that the process of solving a PCA problem is **exactly the same** as obtaining the matrix U in the SVD. We note that if $X = U\Sigma V^\mathsf{T}$ then

$$XX^\mathsf{T} = \Big(U\widetilde{\Sigma}V^\mathsf{T}\Big)\Big(U\widetilde{\Sigma}V^\mathsf{T}\Big)^\mathsf{T} = U\widetilde{\Sigma}\Big(\underbrace{V^\mathsf{T}V}_{=I_n}\Big)\widetilde{\Sigma}^\mathsf{T}U^\mathsf{T} = U\Sigma^2 U^\mathsf{T},$$

where $\widetilde{\Sigma} := \begin{bmatrix} \Sigma \\ 0 \end{bmatrix}$.

This is exactly the same as the symmetric eigenvalue decomposition given by Theorem 5.8. Therefore, to solve a PCA problem **we can obtain the columns of U_r as the eigenvectors $\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_r$ of XX^T corresponding to its largest r eigenvalues** $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$.

To show that this gives us a practical way of computing PCA by hand, we shall present another example.

Example 5.15. Consider the following vectors in $\mathbb{R}^2$,

$$\vec{x}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \vec{x}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad \vec{x}_3 = \begin{bmatrix} 2 \\ 2 \end{bmatrix},$$

and suppose that we would like to reduce the dimension of the points $\{\vec{x}_i\}$ to one. Just as for SVD, we compute the (unnormalized) covariance matrix XX^{T}:

$$\begin{aligned} A = \sum_{i=1}^{3} \vec{x}_i \vec{x}_i^{\top} &= \left(\begin{bmatrix} 1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \end{bmatrix} \begin{bmatrix} 2 & 2 \end{bmatrix} \right) \\ &= \left(\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} + \begin{bmatrix} 4 & 4 \\ 4 & 4 \end{bmatrix} \right) \\ &= \begin{bmatrix} 6 & 2 \\ 2 & 6 \end{bmatrix}, \end{aligned}$$

Then the eigenvalues of A can be obtained by solving

$$\det\left(\begin{bmatrix} 6-\lambda & 2 \\ 2 & 6-\lambda \end{bmatrix} \right) = (6-\lambda)^2 - 4 = 0 \Leftrightarrow \lambda = 8 \text{ or } 4.$$

Since

$$\begin{bmatrix} 6 & 2 \\ 2 & 6 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 8 \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

it follows that the leading eigenvector and the compression matrix are

$$W = U^{\top} = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & 1 \end{bmatrix}.$$

Therefore, the compressed data points ($\vec{y} = W\vec{x}$) are

$$\vec{y}_1 = 0, \quad \vec{y}_2 = 0, \quad \vec{y}_3 = 2\sqrt{2},$$

and the recovered data ($\vec{z} = U\vec{y}$) is

$$\vec{z}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \vec{z}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \vec{z}_3 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}.$$

As a nice exercise, you could plot the original data points $(\vec{x}_1, \vec{x}_2, \vec{x}_3)$ are consider why the PCA would map them to $(\vec{z}_1, \vec{z}_2, \vec{z}_3)$.

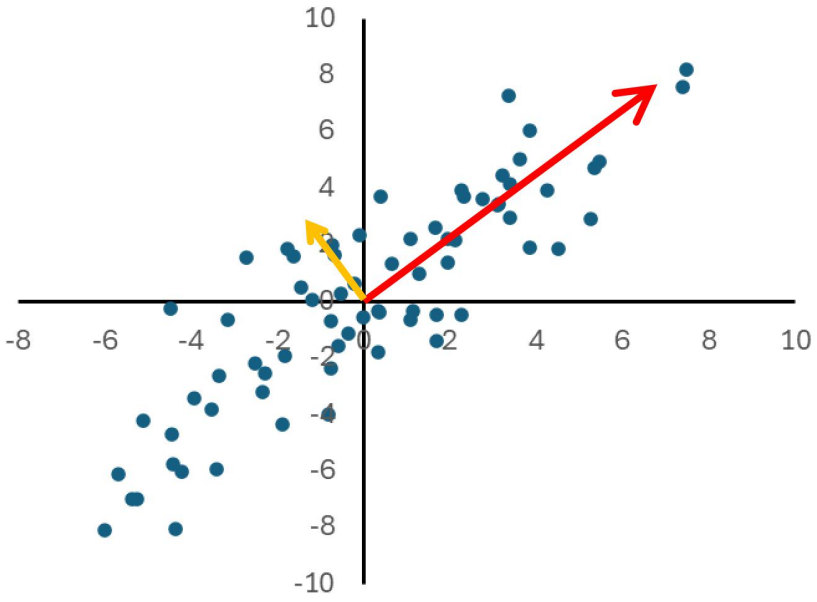

Figure 12: The principal components are the directions which explain the variance of the data.

6 Solution of Initial-Value Problems (IVPs)

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

$$\frac{d\vec{y}}{dt} = f(\vec{y}, t), \quad t \geq 0, \tag{46}$$

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

$$\vec{y}(0) = \vec{y}_0, \tag{47}$$

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

Example 6.1 (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 in Example 6.1.

Example 6.2. A simple pendulum is released from rest at angle α. The angle $\theta = \theta(t)$ of the pendulum satisfies the second-order equation:

$$\frac{d^2\theta}{dt^2} = \frac{g}{a}\sin\theta, \tag{48}$$

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

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

To put these equations in standard form, introduce the new variable ϕ defined by $\phi = d\theta/dt$.

Then (48) becomes

$$\begin{aligned} \frac{d\theta}{dt} &= \phi, \\ \frac{d\phi}{dt} &= \frac{g}{a}\sin\theta. \end{aligned}$$

which has the form (46) with

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

6.1 Euler's method

Given Examples 6.1 and 6.2, suppose we want to find $\vec{y}(t)$ for $t = h$ for some small h. Integrating each side of (46) over $[0, h]$ gives

$$\vec{y}(h) - \vec{y}(0) = \int_0^h \frac{d\vec{y}}{dt}\,dt = \int_0^h f(\vec{y}(t), t)\,dt \tag{49}$$

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 okay if h is small). This yields

$$\vec{y}(h) \approx \vec{y}(0) + hf(\vec{y}(0), 0) = \vec{y}_0 + hf(\vec{y}_0, 0). \tag{50}$$

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

$$\begin{aligned} &\vec{Y}_0 = \vec{y}_0 \\ &\vec{Y}_j = \vec{Y}_{j-1} + hf(\vec{Y}_{j-1}, t_{j-1}), \qquad \text{for all } j \geq 1. \end{aligned} \tag{51}$$

Example 6.3. Solve

$$\frac{dy}{dt} = y^2, \qquad y(0) = 1 \tag{52}$$

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

$$\begin{aligned} Y_0 &= y_0 = 1 \\ Y_1 &= Y_0 + hf(Y_0) = 1 + 0.1(1^2) = 1.1 \\ Y_2 &= Y_1 + hf(Y_1) = 1.1 + 0.1(1.1)^2 = 1.221. \end{aligned}$$

Using the program,we approximate the solution of (52) 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 (46)), 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 $|Y_n - y(t)| \cong \mathcal{O}(h)$ as $h \to 0$. This is what we prove in the next subsection.

6.2 Convergence of one-step methods

We now restrict to the simplified version of (46) with $N = 1$ and $f(y, t) = f(y)$:

$$\frac{dy}{dt} = f(y) \tag{53}$$

subject to the initial condition

$$y(0) = y_0. \tag{54}$$

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

$$\begin{aligned} Y_0 &= y_0 \\ Y_j &= Y_{j-1} + hF_h(Y_{j-1}), \qquad \text{for all } j \geq 1 \end{aligned} \tag{55}$$

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 6.1. If y solves (53),(54), the **local truncation error** for (55) 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 (55)).

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

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

Theorem 6.1. *Suppose F_h is Lipschitz continuous with Lipschitz constant L independent of h. Then the error e_j in* (55) *satisfies*

$$|e_j| \le (1 + hL)|e_{j-1}| + h|\tau_j|, \quad j = 1, 2, 3, \ldots. \tag{56}$$

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

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

Proof. By definition of τ_j,

$$y(t_j) = y(t_{j-1}) + hF_h(y(t_{j-1})) + h\tau_j\,. \tag{57}$$

So with $e_j = y(t_j) - Y_j$, we have by subtracting (55) from (57):

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

By the triangle inequality and the Lipschitz continuity of F_h,

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

We see that (56) holds.

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

$$|e_n| \le h \sum_{j=0}^{n-1} (1 + hL)^j\,|\tau_{n-j}|. \tag{58}$$

Clearly (58) holds for $n = 1$, since the first equation in (55) implies $|e_0| = 0$ and (56) then gives $|e_1| \le h|\tau_1|$. Now if (58) holds for some n, then (56) implies

$$\begin{aligned}
|e_{n+1}| &\le h \sum_{j=0}^{n-1} (1 + hL)^{j+1} \, |\tau_{n-j}| + h|\tau_{n+1}| \\
&= h \sum_{j=1}^{n} (1 + hL)^{j} \, |\tau_{n+1-j}| + h|\tau_{n+1}| \\
&= h \sum_{j=0}^{n} (1 + hL)^{j} \, |\tau_{n+1-j}|.
\end{aligned}$$

Hence, (58) holds for $n + 1$, and for all n by induction.

Finally, from (58) we have, since $\displaystyle\sum_{j=0}^{n-1} q^j = \frac{q^n - 1}{q - 1}$,

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

since $1 + x \le \exp x$, for all $x \ge 0$. And so the result follows for all $nh \le T$.

□

Remark. This theorem shows that the error in the approximation to the solution computed by (55) 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 6.4. Euler's method is (55) with $F_h(Y) := f(Y)$. If we assume that f is Lipschitz continuous, then Theorem 6.1 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

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

Now, since $y(t)$ is the solution of (53), we have, for all $j = 1, \cdots, n$:

$$|\tau_j| \leq \frac{1}{2} \max_{t\in[0,T]} \left| \frac{d^2y}{dt^2}(t) \right| h. \tag{59}$$

Hence if

$$\left| \frac{d^2y}{dt^2}(t) \right| \qquad \text{is bounded for } t \in \mathbb{R}, \tag{60}$$

then Theorem 6.1 implies that

$$|e_n| \leq C(T)h, \tag{61}$$

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 (60) and then concludes (61). To show (60) rigorously, we need to make some assumptions on the given function f. In particular, assume that

$$|f(x)| \leq M, \quad x \in \mathbb{R}, \tag{62}$$

$$\text{and} \quad |f'(x)| \leq L, \quad x \in \mathbb{R}. \tag{63}$$

Then f is Lipschitz continuous with Lipschitz constant L, and by (53) and the chain rule,

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

6.3 Higher-order methods

We saw in (61) 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 (49). For simplicity, restrict to the case the case $N = 1$ and $f(y, t) = f(y)$ again: Then (49) 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 trapezium rule, to obtain

$$y(h) \approx y(0) + \frac{h}{2}\Big(f(y(0)) + f(y(h)) \Big). \tag{64}$$

This motivates the Crank–Nicholson (or Trapezoidal) Method for solving (46):

$$\begin{aligned} &Y_0 = y_0 \\ &Y_j = Y_{j-1} + \frac{h}{2}\Big(f(Y_{j-1}) + f(Y_j) \Big), \qquad \text{for all } j \geq 1. \end{aligned} \tag{65}$$

This method has a big disadvantage, since to find Y_j from Y_{j-1} in (65), 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 (65), 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 (53) and applying Taylor's theorem to both y and dy/dt we get:

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

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 6.2. *Suppose f is Lipschitz continuous with Lipschitz constant L independent of h. If $hL \le 1$, then the error $e_j = y(t_j) - Y_j$ for the Crank–Nicholson method in* (65) *satisfies*

$$|e_j| \le (1 + hL)^2\,|e_{j-1}| + h(1 + hL)\,|\tau_j|, \quad j = 1, 2, 3, \ldots. \tag{66}$$

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

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

Proof. From (65),

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

and by definition of the trunctation 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

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

and rearrange as follows,

$$\begin{aligned}\left(1 - \frac{hL}{2}\right)|e_j| &\le \left(1 + \frac{hL}{2}\right)|e_{j-1}| + h|\tau_j| \\ \Rightarrow \qquad |e_j| &\le \left(1 - \frac{hL}{2}\right)^{-1}\left(\left(1 + \frac{hL}{2}\right)|e_{j-1}| + h|\tau_j|\right).\end{aligned}$$

We note that, for any $0 \le x \le \frac{1}{2}$,

$$(1-x)^{-1} \le 1 + 2x,$$

and thus for $hL \le 1$,

$$\left(1 - \frac{hL}{2}\right)^{-1} \le 1 + hL.$$

Therefore

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

The rest of the proof is Problem E8.3. □

6.3.1 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:

$$\widehat{Y}_j = Y_{j-1} + hf(Y_{j-1}) \qquad \text{“prediction"} \tag{67}$$

$$Y_j = Y_{j-1} + \frac{h}{2}\Big(f(Y_{j-1}) + f(\widehat{Y}_j)\Big) \qquad \text{“correction"} \tag{68}$$

This method fits into the framework of Theorem 6.1 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 (55) with $F_h(Y) := \frac{1}{2}f(Y) + \frac{1}{2}f\Big(Y + hf(Y)\Big)$. The truncation error is

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

and it turns out that $\|\tau_j\| = \mathcal{O}(h^2)$ for sufficiently smooth f (see Problem E8.2).

Moreover if f is Lipschitz then so is F_h, since

$$\begin{aligned}
|F_h(Y) - F_h(Z)| &\le \frac{1}{2}|f(Y) - f(Z)| + \frac{1}{2}|f(Y + hf(Y)) - f(Z + hf(Z))| \\
&\le \frac{L}{2}|Y - Z| + \frac{L}{2}|(Y + hf(Y)) - (Z + hf(Z))| \\
&\le L|Y - Z| + \frac{hL}{2}|f(Y) - f(Z)| \le \left(L + \frac{1}{2}hL^2\right)|Y - Z|.
\end{aligned}$$

So under these conditions, Theorem 6.1 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 6.5. Let

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

This is a fourth order Runge-Kutta method (requires some analysis).