**Numerical Analysis Hilary Term 2021**
**Lecture 1: Lagrange Interpolation**

Numerical analysis is the study of computational algorithms for solving problems in scientific computing. It combines mathematical beauty, rigor and numerous applications; we hope you'll enjoy it! In this course we will cover the basics of three key fields in the subject:

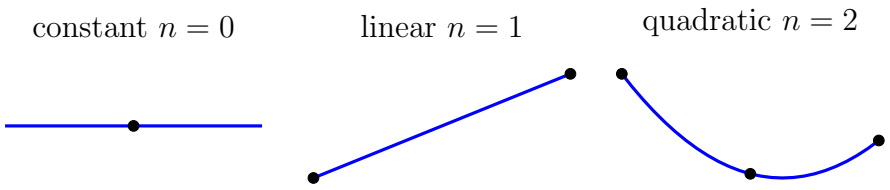
- Approximation Theory (lectures 1, 9–11); recommended reading: L. N. Trefethen, Approximation Theory and Approximation Practice, and E. Süli and D. F. Mayers, An Introduction to Numerical Analysis.
- Numerical Linear Algebra (lectures 2–8); recommended reading: L. N. Trefethen and D. Bau, Numerical Linear Algebra.
- Numerical Solution of Differential Equations (lectures 12–16); recommended reading: E. Süli and D. F. Mayers, An Introduction to Numerical Analysis.

This first lecture comes from Chapter 6 of Süli and Mayers.

**Notation:** $\Pi_n = \{\text{real polynomials of degree} \leq n\}$

**Setup:** Given data $f_i$ at distinct $x_i$, $i = 0, 1, \ldots, n$, with $x_0 < x_1 < \cdots < x_n$, can we find a polynomial $p_n$ such that $p_n(x_i) = f_i$? Such a polynomial is said to **interpolate** the data, and (as we shall see) can approximate $f$ at other values of $x$ if $f$ is smooth enough. This is the most basic question in approximation theory.

**E.g.:**




**Theorem.** $\exists p_n \in \Pi_n$ such that $p_n(x_i) = f_i$ for $i = 0, 1, \ldots, n$.

**Proof.** Consider, for $k = 0, 1, \ldots, n$, the "cardinal polynomial"

$$L_{n,k}(x) = \frac{(x - x_0) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n)}{(x_k - x_0) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_n)} \in \Pi_n. \tag{1}$$

Then $L_{n,k}(x_i) = \delta_{ik}$, that is,

$$L_{n,k}(x_i) = 0 \text{ for } i = 0, \ldots, k-1, k+1, \ldots, n \text{ and } L_{n,k}(x_k) = 1.$$

So now define

$$p_n(x) = \sum_{k=0}^{n} f_k L_{n,k}(x) \in \Pi_n \tag{2}$$

$\Longrightarrow$

$$p_n(x_i) = \sum_{k=0}^{n} f_k L_{n,k}(x_i) = f_i \quad \text{for} \quad i = 0, 1, \ldots, n. \qquad \square$$

The polynomial (2) is the **Lagrange interpolating polynomial**.

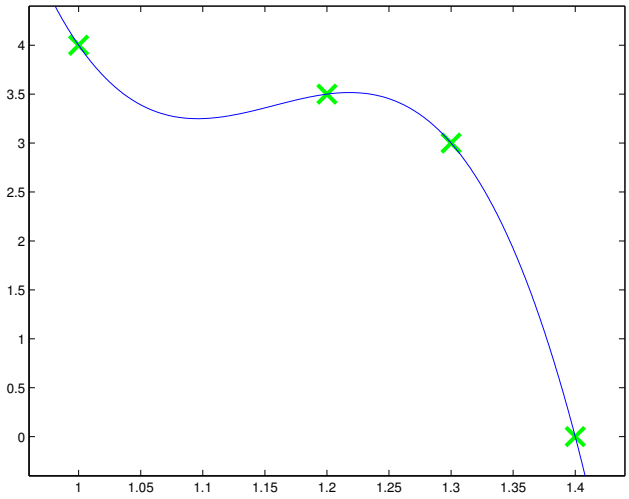
**Theorem.** The interpolating polynomial of degree $\leq n$ is unique.

**Proof.** Consider two interpolating polynomials $p_n, q_n \in \Pi_n$. Their difference $d_n = p_n - q_n \in \Pi_n$ satisfies $d_n(x_k) = 0$ for $k = 0, 1, \ldots, n$. i.e., $d_n$ is a polynomial of degree at most $n$ but has at least $n+1$ distinct roots. Algebra $\Longrightarrow$ $d_n \equiv 0 \Longrightarrow$ $p_n = q_n$. $\square$

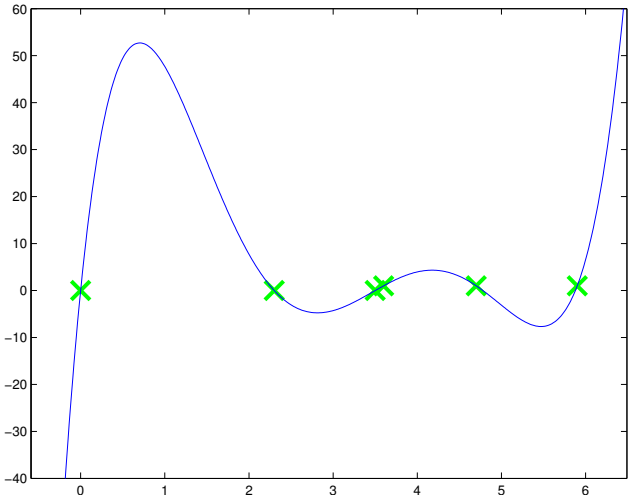
**Matlab:**

```
>> help lagrange
 LAGRANGE  Plots the Lagrange polynomial interpolant for the
 given DATA at the given KNOTS

>> lagrange([1,1.2,1.3,1.4],[4,3.5,3,0]);
```




```
>> lagrange([0,2.3,3.5,3.6,4.7,5.9],[0,0,0,1,1,1]);
```




**Data from an underlying smooth function:** Suppose that $f(x)$ has at least $n+1$ smooth derivatives in the interval $(x_0, x_n)$. Let $f_k = f(x_k)$ for $k = 0, 1, \ldots, n$, and let $p_n$ be the Lagrange interpolating polynomial for the data $(x_k, f_k)$, $k = 0, 1, \ldots, n$.

**Error:** How large can the error $f(x) - p_n(x)$ be on the interval $[x_0, x_n]$?

**Theorem.** For every $x \in [x_0, x_n]$ there exists $\xi = \xi(x) \in (x_0, x_n)$ such that

$$e(x) \stackrel{\text{def}}{=} f(x) - p_n(x) = (x - x_0)(x - x_1) \cdots (x - x_n) \frac{f^{(n+1)}(\xi)}{(n+1)!}, \tag{3}$$

where $f^{(n+1)}$ is the $(n+1)$-st derivative of $f$.

**Proof.** Trivial for $x = x_k$, $k = 0, 1, \ldots, n$ as $e(x) = 0$ by construction. So suppose $x \neq x_k$. Let

$$\phi(t) \stackrel{\text{def}}{=} e(t) - \frac{e(x)}{\pi(x)} \pi(t),$$

where

$$\begin{aligned} \pi(t) \stackrel{\text{def}}{=} & \quad (t - x_0)(t - x_1) \cdots (t - x_n) \\ = & \quad t^{n+1} - \left( \sum_{i=0}^{n} x_i \right) t^n + \cdots (-1)^{n+1} x_0 x_1 \cdots x_n \\ \in & \quad \Pi_{n+1}. \end{aligned}$$

Now note that $\phi$ vanishes at $n+2$ points $x$ and $x_k$, $k = 0, 1, \ldots, n$. $\Longrightarrow$ $\phi'$ vanishes at $n+1$ points $\xi_0, \ldots, \xi_n$ between these points $\Longrightarrow$ $\phi''$ vanishes at $n$ points between these new points, and so on until $\phi^{(n+1)}$ vanishes at an (unknown) point $\xi$ in $(x_0, x_n)$. But

$$\phi^{(n+1)}(t) = e^{(n+1)}(t) - \frac{e(x)}{\pi(x)} \pi^{(n+1)}(t) = f^{(n+1)}(t) - \frac{e(x)}{\pi(x)} (n+1)!$$

since $p_n^{(n+1)}(t) \equiv 0$ and because $\pi(t)$ is a monic polynomial of degree $n+1$. The result then follows immediately from this identity since $\phi^{(n+1)}(\xi) = 0$.

□

**Example:** $f(x) = \log(1+x)$ on $[0, 1]$. Here, $|f^{(n+1)}(\xi)| = n!/(1+\xi)^{n+1} < n!$ on $(0, 1)$. So $|e(x)| < |\pi(x)| n!/(n+1)! \leq 1/(n+1)$ since $|x - x_k| \leq 1$ for each $x$, $x_k$, $k = 0, 1, \ldots, n$, in

$[0,1] \Longrightarrow |\pi(x)| \leq 1$. This is probably pessimistic for many $x$, e.g. for $x = \frac{1}{2}$, $\pi(\frac{1}{2}) \leq 2^{-(n+1)}$ as $|\frac{1}{2} - x_k| \leq \frac{1}{2}$.

This shows the important fact that the error can be large at the end points when samples $\{x_k\}$ are equispaced points, an effect known as the "Runge phenomena" (Carl Runge, 1901), which we return to in lecture 4.

**Generalisation:** Given data $f_i$ and $g_i$ at distinct $x_i$, $i = 0, 1, \ldots, n$, with $x_0 < x_1 < \cdots < x_n$, can we find a polynomial $p$ such that $p(x_i) = f_i$ and $p'(x_i) = g_i$? (i.e., interpolate derivatives in addition to values)

**Theorem.** There is a unique polynomial $p_{2n+1} \in \Pi_{2n+1}$ such that $p_{2n+1}(x_i) = f_i$ and $p'_{2n+1}(x_i) = g_i$ for $i = 0, 1, \ldots, n$.

**Construction:** Given $L_{n,k}(x)$ in (1), let

$$\begin{aligned} H_{n,k}(x) &= [L_{n,k}(x)]^2(1 - 2(x - x_k)L'_{n,k}(x_k)) \\ \text{and } K_{n,k}(x) &= [L_{n,k}(x)]^2(x - x_k). \end{aligned}$$

Then

$$p_{2n+1}(x) = \sum_{k=0}^{n} [f_k H_{n,k}(x) + g_k K_{n,k}(x)] \tag{4}$$

interpolates the data as required. The polynomial (4) is called the **Hermite interpolating polynomial**. Note that $H_{n,k}(x_i) = \delta_{ik}$ and $H'_{n,k}(x_i) = 0$, and $K_{n,k}(x_i) = 0, K'_{n,k}(x_i) = \delta_{ik}$.

**Theorem.** Let $p_{2n+1}$ be the Hermite interpolating polynomial in the case where $f_i = f(x_i)$ and $g_i = f'(x_i)$ and $f$ has at least $2n+2$ smooth derivatives. Then, for every $x \in [x_0, x_n]$,

$$f(x) - p_{2n+1}(x) = [(x - x_0)(x - x_1) \cdots (x - x_n)]^2 \frac{f^{(2n+2)}(\xi)}{(2n+2)!},$$

where $\xi \in (x_0, x_n)$ and $f^{(2n+2)}$ is the $(2n+2)$nd derivative of $f$.

Proof (non-examinable): see Süli and Mayers, Theorem 6.4. □

We note that as $x_k \to 0$ in (3), we essentialy recover Taylor's theorem with $p_n(x)$ equal to the first $n+1$ terms in Taylor's expansion. Taylor's theorem can be regarded as a special case of Lagrange interpolation where we interpolate high-order derivatives at a single point.

**Numerical Analysis Hilary Term 2021**
**Lecture 2: Gaussian Elimination and LU factorisation**

In lecture 1 we treated Lagrange interpolation. A traditional, more straightforward approach (worse for computation!) would be to express the interpolating polynomial as $p_n(x) = \sum_{i=0}^{n} c_i x^i$ and find the coefficients $c_i$ by a linear system of equations:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix}.$$

This is a linear algebra problem, which is the subject we will discuss in the next lectures. We start with solving linear systems.

**Setup:** Given a square $n$ by $n$ matrix $A$ and vector with $n$ components $b$, find $x$ such that

$$Ax = b.$$

Equivalently find $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ for which

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n. \end{aligned} \tag{1}$$

**Lower-triangular matrices:** the matrix $A$ is **lower triangular** if $a_{ij} = 0$ for all $1 \le i < j \le n$. The system (1) is easy to solve if $A$ is lower triangular.

$$\begin{aligned} a_{11}x_1 &= b_1 \implies x_1 = \frac{b_1}{a_{11}} \quad \Downarrow \\ a_{21}x_1 + a_{22}x_2 &= b_2 \implies x_2 = \frac{b_2 - a_{21}x_1}{a_{22}} \quad \Downarrow \\ &\vdots \quad \Downarrow \\ a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{ii}x_i &= b_i \implies x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j}{a_{ii}} \quad \Downarrow \\ &\vdots \quad \Downarrow \end{aligned}$$

This works if, and only if, $a_{ii} \neq 0$ for each $i$. The procedure is known as **forward substitution**.

**Computational work estimate:** one floating-point operation (flop) is one scalar multiply/division/addition/subtraction as in $y = a * x$ where $a$, $x$ and $y$ are computer representations of real scalars.[1]

[1] This is an abstraction: e.g., some hardware can do $y = a * x + b$ in one FMA flop ("Fused Multiply and Add") but then needs several FMA flops for a single division. For a trip down this sort of rabbit hole, look up the "Fast inverse square root" as used in the source code of the video game "Quake III Arena".

Hence the work in forward substitution is 1 flop to compute $x_1$ plus 3 flops to compute $x_2$ plus $\ldots$ plus $2i-1$ flops to compute $x_i$ plus $\ldots$ plus $2n-1$ flops to compute $x_n$, or in total

$$\sum_{i=1}^{n}(2i-1) = 2\left(\sum_{i=1}^{n} i\right) - n = 2\left(\tfrac{1}{2}n(n+1)\right) - n = n^2 + \text{lower order terms}$$

flops. We sometimes write this as $n^2 + O(n)$ flops or more crudely $O(n^2)$ flops.

**Upper-triangular matrices:** the matrix $A$ is **upper triangular** if $a_{ij} = 0$ for all $1 \le j < i \le n$. Once again, the system (1) is easy to solve if $A$ is upper triangular.

$$\begin{array}{rcl} \vdots & & \Uparrow \\ a_{ii}x_i + \cdots + a_{in-1}x_{n-1} + a_{1n}x_n = b_i & \Longrightarrow & x_i = \dfrac{b_i - \displaystyle\sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}} \quad \Uparrow \\ \vdots & & \Uparrow \\ a_{n-1n-1}x_{n-1} + a_{n-1n}x_n = b_{n-1} & \Longrightarrow & x_{n-1} = \dfrac{b_{n-1} - a_{n-1n}x_n}{a_{n-1n-1}} \quad \Uparrow \\ a_{nn}x_n = b_n & \Longrightarrow & x_n = \dfrac{b_n}{a_{nn}}. \quad \Uparrow \end{array}$$

Again, this works if, and only if, $a_{ii} \neq 0$ for each $i$. The procedure is known as **backward** or **back substitution**. This also takes approximately $n^2$ flops.

For computation, we need a reliable, systematic technique for reducing $Ax = b$ to $Ux = c$ with the same solution $x$ but with $U$ (upper) triangular $\Longrightarrow$ Gauss elimination.

**Example**

$$\begin{bmatrix} 3 & -1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 12 \\ 11 \end{bmatrix}.$$

Multiply first equation by $1/3$ and subtract from the second $\Longrightarrow$

$$\begin{bmatrix} 3 & -1 \\ 0 & \frac{7}{3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 12 \\ 7 \end{bmatrix}.$$

**Gauss(ian) Elimination (GE):** this is most easily described in terms of overwriting the matrix $A = \{a_{ij}\}$ and vector $b$. At each stage, it is a systematic way of introducing zeros into the lower triangular part of $A$ by subtracting multiples of previous equations (i.e., rows); such (elementary row) operations do not change the solution.

for columns $j = 1, 2, \ldots, n-1$
  for rows $i = j+1, j+2, \ldots, n$

$$\begin{aligned} \text{row } i &\leftarrow \text{row } i - \frac{a_{ij}}{a_{jj}} * \text{row } j \\ b_i &\leftarrow b_i - \frac{a_{ij}}{a_{jj}} * b_j \end{aligned}$$

  end
end

**Example.**

$$\begin{bmatrix} 3 & -1 & 2 \\ 1 & 2 & 3 \\ 2 & -2 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 12 \\ 11 \\ 2 \end{bmatrix} : \quad \text{represent as} \quad \left[\begin{array}{ccc|c} 3 & -1 & 2 & 12 \\ 1 & 2 & 3 & 11 \\ 2 & -2 & -1 & 2 \end{array}\right]$$

$$\Longrightarrow \quad \begin{array}{l} \text{row } 2 \leftarrow \text{row } 2 - \frac{1}{3}\text{row } 1 \\ \text{row } 3 \leftarrow \text{row } 3 - \frac{2}{3}\text{row } 1 \end{array} \left[\begin{array}{ccc|c} 3 & -1 & 2 & 12 \\ 0 & \frac{7}{3} & \frac{7}{3} & 7 \\ 0 & -\frac{4}{3} & -\frac{7}{3} & -6 \end{array}\right]$$

$$\Longrightarrow \quad \begin{array}{l} \\ \text{row } 3 \leftarrow \text{row } 3 + \frac{4}{7}\text{row } 2 \end{array} \left[\begin{array}{ccc|c} 3 & -1 & 2 & 12 \\ 0 & \frac{7}{3} & \frac{7}{3} & 7 \\ 0 & 0 & -1 & -2 \end{array}\right]$$

Back substitution:

$$\begin{aligned} x_3 &= 2 \\ x_2 &= \frac{7 - \frac{7}{3}(2)}{\frac{7}{3}} = 1 \\ x_1 &= \frac{12 - (-1)(1) - 2(2)}{3} = 3. \end{aligned}$$

**Cost of Gaussian Elimination:** note, row $i \leftarrow$ row $i - \dfrac{a_{ij}}{a_{jj}} *$ row $j$ is

for columns $k = j+1, j+2, \ldots, n$

$$a_{ik} \leftarrow a_{ik} - \frac{a_{ij}}{a_{jj}} a_{jk}$$

end

This is approximately $2(n-j)$ flops as the **multiplier** $a_{ij}/a_{jj}$ is calculated with just one flop; $a_{jj}$ is called the **pivot**. Overall therefore, the cost of GE is approximately

$$\sum_{j=1}^{n-1} 2(n-j)^2 = 2\sum_{l=1}^{n-1} l^2 = 2\frac{n(n-1)(2n-1)}{6} = \frac{2}{3}n^3 + O(n^2)$$

flops. The calculations involving $b$ are

$$\sum_{j=1}^{n-1} 2(n-j) = 2\sum_{l=1}^{n-1} l = 2\frac{n(n-1)}{2} = n^2 + O(n)$$

flops, just as for the triangular substitution.

**LU factorization:**

The basic operation of Gaussian Elimination, row $i \leftarrow$ row $i + \lambda *$ row $j$, can be achieved by pre-multiplication by a special lower-triangular matrix

$$M(i,j,\lambda) = I + \begin{bmatrix} 0 & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 0 \end{bmatrix} \leftarrow i \\ \qquad\qquad \uparrow \\ \qquad\qquad j$$

where $I$ is the identity matrix.

**Example:** $n = 4$,

$$M(3,2,\lambda) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \lambda & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad M(3,2,\lambda) \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} a \\ b \\ \lambda b + c \\ d \end{bmatrix},$$

i.e., $M(3,2,\lambda)A$ performs: row 3 of $A \leftarrow$ row 3 of $A + \lambda *$ row 2 of $A$ and similarly $M(i,j,\lambda)A$ performs: row $i$ of $A \leftarrow$ row $i$ of $A + \lambda *$ row $j$ of $A$.

So GE for e.g., $n = 3$ is

$$\underset{l_{32} = \frac{a_{32}}{a_{22}}}{M(3,2,-l_{32})} \cdot \underset{l_{31} = \frac{a_{31}}{a_{11}}}{M(3,1,-l_{31})} \cdot \underset{l_{21} = \frac{a_{21}}{a_{11}}}{M(2,1,-l_{21})} \cdot A = U = (\text{upper triangular}).$$

The $l_{ij}$ are called the **multipliers**.

**Be careful:** each multiplier $l_{ij}$ uses the data $a_{ij}$ and $a_{ii}$ that *results from the transformations already applied*, not data from the original matrix. So $l_{32}$ uses $a_{32}$ and $a_{22}$ that result from the previous transformations $M(2,1,-l_{21})$ and $M(3,1,-l_{31})$.

**Lemma.** If $i \neq j$, $(M(i,j,\lambda))^{-1} = M(i,j,-\lambda)$.

**Proof.** Exercise.

**Outcome:** for $n = 3$, $A = M(2,1,l_{21}) \cdot M(3,1,l_{31}) \cdot M(3,2,l_{32}) \cdot U$, where

$$M(2,1,l_{21}) \cdot M(3,1,l_{31}) \cdot M(3,2,l_{32}) = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} = L = (\text{lower triangular}).$$

This is true for general $n$:

**Theorem.** For any dimension $n$, GE can be expressed as $A = LU$, where $U = (\text{upper triangular})$ is upper triangular resulting from GE, and $L = (\text{lower triangular})$ is unit lower triangular (lower

triangular with ones on the diagonal) with $l_{ij}$ = multiplier used to create the zero in the $(i,j)$th position.

Most implementations of GE therefore, rather than doing GE as above,

$$\begin{array}{rll} \text{factorize} & A = LU & (\approx \tfrac{1}{3}n^3 \text{ adds} + \approx \tfrac{1}{3}n^3 \text{ mults}) \\ \text{and then solve} & Ax = b & \\ \text{by solving} & Ly = b & \text{(forward substitution)} \\ \text{and then} & Ux = y & \text{(back substitution)} \end{array}$$

Note: this is much more efficient if we have many different right-hand sides $b$ but the same $A$.

**Pivoting:** GE or LU can fail if the pivot $a_{ii} = 0$. For example, if

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

GE fails at the first step. However, we are free to reorder the equations (i.e., the rows) into any order we like. For example, the equations

$$\begin{array}{l} 0 \cdot x_1 + 1 \cdot x_2 = 1 \\ 1 \cdot x_1 + 0 \cdot x_2 = 2 \end{array} \quad \text{and} \quad \begin{array}{l} 1 \cdot x_1 + 0 \cdot x_2 = 2 \\ 0 \cdot x_1 + 1 \cdot x_2 = 1 \end{array}$$

are the same, but their matrices

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

have had their rows reordered: GE fails for the first but succeeds for the second $\Longrightarrow$ better to interchange the rows and then apply GE.

**Partial pivoting:** when creating the zeros in the $j$th column, find

$$|a_{kj}| = \max(|a_{jj}|, |a_{j+1j}|, \ldots, |a_{nj}|),$$

then swap (interchange) rows $j$ and $k$.

For example,

$$\begin{bmatrix} a_{11} & \cdot & a_{1j-1} & a_{1j} & \cdot & \cdot & \cdot & a_{1n} \\ 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & a_{j-1j-1} & a_{j-1j} & \cdot & \cdot & \cdot & a_{j-1n} \\ 0 & \cdot & 0 & a_{jj} & \cdot & \cdot & \cdot & a_{jn} \\ 0 & \cdot & 0 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & a_{kj} & \cdot & \cdot & \cdot & a_{kn} \\ 0 & \cdot & 0 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & a_{nj} & \cdot & \cdot & \cdot & a_{nn} \end{bmatrix} \rightarrow \begin{bmatrix} a_{11} & \cdot & a_{1j-1} & a_{1j} & \cdot & \cdot & \cdot & a_{1n} \\ 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & a_{j-1j-1} & a_{j-1j} & \cdot & \cdot & \cdot & a_{j-1n} \\ 0 & \cdot & 0 & a_{kj} & \cdot & \cdot & \cdot & a_{kn} \\ 0 & \cdot & 0 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & a_{jj} & \cdot & \cdot & \cdot & a_{jn} \\ 0 & \cdot & 0 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & a_{nj} & \cdot & \cdot & \cdot & a_{nn} \end{bmatrix}$$

**Property:** GE with partial pivoting cannot fail if $A$ is nonsingular.

**Proof.** If $A$ is the first matrix above at the $j$th stage,

$$\det[A] = a_{11} \cdots a_{j-1j-1} \cdot \det \begin{bmatrix} a_{jj} & \cdot & \cdot & \cdot & a_{jn} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{kj} & \cdot & \cdot & \cdot & a_{kn} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{nj} & \cdot & \cdot & \cdot & a_{nn} \end{bmatrix}.$$

Hence $\det[A] = 0$ if $a_{jj} = \cdots = a_{kj} = \cdots = a_{nj} = 0$. Thus if the pivot $a_{k,j}$ is zero, $A$ is singular. So if $A$ is nonsingular, all of the pivots are nonzero. (Note: actually $a_{nn}$ can be zero and an LU factorization still exist.)

The effect of pivoting is just a permutation (reordering) of the rows, and hence can be represented by a permutation matrix $P$.

**Permutation matrix:** $P$ has the same rows as the identity matrix, but in the pivoted order. So

$$PA = LU$$

represents the factorization—equivalent to GE with partial pivoting. E.g.,

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} A$$

has the 2nd row of $A$ first, the 3rd row of $A$ second and the 1st row of $A$ last.

**Matlab example:**

```
>> A = rand(5,5)
A =
      0.69483      0.38156      0.44559       0.6797      0.95974
       0.3171      0.76552      0.64631       0.6551      0.34039
      0.95022       0.7952      0.70936      0.16261      0.58527
     0.034446      0.18687      0.75469        0.119      0.22381
      0.43874      0.48976      0.27603      0.49836      0.75127
>> exactx = ones(5,1);  b = A*exactx;
>> [LL, UU] = lu(A)  % note "psychologically lower triangular" LL
LL =
      0.73123     -0.39971      0.15111            1            0
      0.33371            1            0            0            0
            1            0            0            0            0
     0.036251        0.316            1            0            0
      0.46173      0.24512     -0.25337      0.31574            1
UU =
      0.95022       0.7952      0.70936      0.16261      0.58527
            0      0.50015      0.40959      0.60083      0.14508
            0            0      0.59954    -0.076759      0.15675
            0            0            0      0.81255      0.56608
            0            0            0            0      0.30645
```

```
>> [L, U, P] = lu(A)
L =
            1            0            0            0            0
      0.33371            1            0            0            0
     0.036251        0.316            1            0            0
      0.73123     -0.39971      0.15111            1            0
      0.46173      0.24512     -0.25337      0.31574            1
U =
      0.95022       0.7952      0.70936      0.16261      0.58527
            0      0.50015      0.40959      0.60083      0.14508
            0            0      0.59954    -0.076759      0.15675
            0            0            0      0.81255      0.56608
            0            0            0            0      0.30645
P =
     0     0     1     0     0
     0     1     0     0     0
     0     0     0     1     0
     1     0     0     0     0
     0     0     0     0     1

>> max(max(P'*L - LL)))  % we see LL is P'*L
ans =
     0

>> y = L \ (P*b);  % now to solve Ax = b...
>> x = U \ y
x =
            1
            1
            1
            1
            1

>> norm(x - exactx, 2)  % within roundoff error of exact soln
ans =
   3.5786e-15
```

**Numerical Analysis Hilary Term 2021**
**Lecture 3: QR Factorization**

---

**Definition:** a square real matrix $Q$ is **orthogonal** if $Q^{\mathrm{T}} = Q^{-1}$. This is true if, and only if, $Q^{\mathrm{T}}Q = I = QQ^{\mathrm{T}}$.

**Example:** the permutation matrices $P$ in LU factorization with partial pivoting are orthogonal.

**Proposition.** The product of orthogonal matrices is an orthogonal matrix.

**Proof.** If $S$ and $T$ are orthogonal, $(ST)^{\mathrm{T}} = T^{\mathrm{T}}S^{\mathrm{T}}$ so

$$(ST)^{\mathrm{T}}(ST) = T^{\mathrm{T}}S^{\mathrm{T}}ST = T^{\mathrm{T}}(S^{\mathrm{T}}S)T = T^{\mathrm{T}}T = I.$$

**Definition:** The **scalar (dot)(inner) product** of two vectors

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \text{and} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

in $\mathbb{R}^n$ is

$$x^{\mathrm{T}}y = y^{\mathrm{T}}x = \sum_{i=1}^{n} x_i y_i \in \mathbb{R}$$

**Definition:** Two vectors $x,\ y \in \mathbb{R}^n$ are **orthogonal** if $x^{\mathrm{T}}y = 0$. A set of vectors $\{u_1, u_2, \ldots, u_r\}$ is an **orthogonal set** if $u_i^{\mathrm{T}}u_j = 0$ for all $i, j \in \{1, 2, \ldots, r\}$ such that $i \neq j$.

**Lemma.** The columns of an orthogonal matrix $Q$ form an orthogonal set, which is moreover an orthonormal basis for $\mathbb{R}^n$.

**Proof.** Suppose that $Q = [q_1 \ \ q_2 \ \ \cdots \ \ q_n]$, i.e., $q_j$ is the $j$th column of $Q$. Then

$$Q^{\mathrm{T}}Q = I = \begin{bmatrix} q_1^{\mathrm{T}} \\ q_2^{\mathrm{T}} \\ \vdots \\ q_n^{\mathrm{T}} \end{bmatrix} [q_1 \ \ q_2 \ \ \cdots \ \ q_n] = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

Comparing the $(i, j)$th entries yields

$$q_i^{\mathrm{T}}q_j = \begin{cases} 0 & i \neq j \\ 1 & i = j. \end{cases}$$

Note that the columns of an orthogonal matrix are of length 1 as $q_i^{\mathrm{T}}q_i = 1$, so they form an orthonormal.

To see that it forms a basis, let $x \in \mathbb{R}^n$ be any vector. One has $x = QQ^T x = Qc$ where $c = Q^T x$, so $x = \sum_{i=1}^n c_i q_i$.

**Lemma.** If $u \in \mathbb{R}^n$, $P$ is $n$-by-$n$ orthogonal and $v = Pu$, then $u^{\mathrm{T}}u = v^{\mathrm{T}}v$.

**Proof.** $v^{\mathrm{T}}v = (Pu)^{\mathrm{T}}(Pu) = (u^{\mathrm{T}}P^{\mathrm{T}})(Pu) = u^{\mathrm{T}}(P^{\mathrm{T}}P)u = u^{\mathrm{T}}u$.

**Definition:** The **outer product** of two vectors $x$ and $y \in \mathbb{R}^n$ is

$$xy^{\mathrm{T}} = \begin{bmatrix} x_1y_1 & x_1y_2 & \cdots & x_1y_n \\ x_2y_1 & x_2y_2 & \cdots & x_2y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_ny_1 & x_ny_2 & \cdots & x_ny_n \end{bmatrix},$$

an $n$-by-$n$ matrix (notation: $xy^{\mathrm{T}} \in \mathbb{R}^{n\times n}$). More usefully, if $z \in \mathbb{R}^n$, then

$$(xy^{\mathrm{T}})z = xy^{\mathrm{T}}z = x(y^{\mathrm{T}}z) = \left(\sum_{i=1}^n y_iz_i\right)x.$$

**Definition:** For $w \in \mathbb{R}^n$, $w \neq 0$, the **Householder** reflector $H(w) \in \mathbb{R}^{n\times n}$ is the matrix

$$H(w) = I - \frac{2}{w^{\mathrm{T}}w}ww^{\mathrm{T}}.$$

**Proposition.** $H(w)$ is a symmetric orthogonal matrix.

**Proof.**

Symmetry is straightforward to verify. For orthogonality,

$$\begin{aligned} H(w)H(w)^{\mathrm{T}} &= \left(I - \frac{2}{w^{\mathrm{T}}w}ww^{\mathrm{T}}\right)\left(I - \frac{2}{w^{\mathrm{T}}w}ww^{\mathrm{T}}\right) \\ &= I - \frac{4}{w^{\mathrm{T}}w}ww^{\mathrm{T}} + \frac{4}{(w^{\mathrm{T}}w)^2}w(w^{\mathrm{T}}w)w^{\mathrm{T}} \\ &= I. \end{aligned}$$

□

**Lemma.** Given $u \in \mathbb{R}^n$, there exists a $w \in \mathbb{R}^n$ such that

$$H(w)u = \begin{bmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{bmatrix} \equiv v,$$

say, where $\alpha = \pm\sqrt{u^{\mathrm{T}}u}$.

**Remark**: Since $H(w)$ is an orthogonal matrix for any $w \in \mathbb{R}$, $w \neq 0$, it is necessary for the validity of the equality $H(w)u = v$ that $v^{\mathrm{T}}v = u^{\mathrm{T}}u$, i.e., $\alpha^2 = u^{\mathrm{T}}u$; hence our choice of $\alpha = \pm\sqrt{u^{\mathrm{T}}u}$.

**Proof.** Take $w = \gamma(u - v)$, where $\gamma \neq 0$. Recall that $u^{\mathrm{T}}u = v^{\mathrm{T}}v$. Thus,

$$\begin{aligned} w^{\mathrm{T}}w &= \gamma^2(u-v)^{\mathrm{T}}(u-v) = \gamma^2(u^{\mathrm{T}}u - 2u^{\mathrm{T}}v + v^{\mathrm{T}}v) \\ &= \gamma^2(u^{\mathrm{T}}u - 2u^{\mathrm{T}}v + u^{\mathrm{T}}u) = 2\gamma u^{\mathrm{T}}(\gamma(u-v)) \\ &= 2\gamma w^{\mathrm{T}}u. \end{aligned}$$

So

$$H(w)u = \left(I - \frac{2}{w^{\mathrm{T}}w}ww^{\mathrm{T}}\right)u = u - \frac{2w^{\mathrm{T}}u}{w^{\mathrm{T}}w}w = u - \frac{1}{\gamma}w = u - (u - v) = v.$$

□

Now if $u$ is the first column of the $n$-by-$n$ matrix $A$,

$$H(w)A = \left[\begin{array}{c|ccc} \alpha & \times & \cdots & \times \\ \hline 0 & & & \\ \vdots & & B & \\ 0 & & & \end{array}\right], \quad \text{where } \times = \text{general entry.}$$

Similarly for $B$, we can find $\hat{w} \in \mathbb{R}^{n-1}$ such that

$$H(\hat{w})B = \left[\begin{array}{c|ccc} \beta & \times & \cdots & \times \\ \hline 0 & & & \\ \vdots & & C & \\ 0 & & & \end{array}\right]$$

and then

$$\left[\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & H(\hat{w}) & \\ 0 & & & \end{array}\right] H(w)A = \left[\begin{array}{cc|ccc} \alpha & \times & \times & \cdots & \times \\ 0 & \beta & \times & \cdots & \times \\ 0 & 0 & & & \\ 0 & 0 & & C & \\ \vdots & \vdots & & & \\ 0 & 0 & & & \end{array}\right].$$

Note

$$\left[\begin{array}{cc} 1 & 0 \\ 0 & H(\hat{w}) \end{array}\right] = H(w_2), \quad \text{where } w_2 = \left[\begin{array}{c} 0 \\ \hat{w} \end{array}\right].$$

Thus if we continue in this manner for the $n-1$ steps, we obtain

$$\underbrace{H(w_{n-1})\cdots H(w_3)H(w_2)H(w)}_{Q^{\mathrm{T}}}A = \left[\begin{array}{cccc} \alpha & \times & \cdots & \times \\ 0 & \beta & \cdots & \times \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \gamma \end{array}\right] = \left(\begin{array}{c} \\ \urcorner \end{array}\right).$$

The matrix $Q^{\mathrm{T}}$ is orthogonal as it is the product of orthogonal (Householder) matrices, so we have constructively proved that

**Theorem.** Given any square matrix $A$, there exists an orthogonal matrix $Q$ and an upper triangular matrix $R$ such that

$$A = QR$$

**Notes:** 1. This could also be established using the Gram–Schmidt Process.
2. If $u$ is already of the form $(\alpha,\ 0,\ \cdots,\ 0)^{\mathrm{T}}$, we just take $H = I$.

3. It is not necessary that $A$ is square: if $A \in \mathbb{R}^{m\times n}$, then we need the product of (a) $m-1$ Householder matrices if $m \leq n \implies$

$$(\square) = A = QR = (\square)(\lceil\rceil)$$

or (b) $n$ Householder matrices if $m > n \implies$

$$\left(\square\right) = A = QR = \left(\square\right)\left(\lceil\right). \tag{1}$$

This $m > n$ case is particular important, and we note that one can also write

$$\left(\square\right) = A = QR = \left(\square\right)(\lceil).$$

This is called the *thin* QR factorization, wherein $Q \in \mathbb{R}^{m\times n}$ has orthonormal columns and has the same size as $A$; by contrast, in (1) $Q$ is square orthogonal, and (1) is called the *full* QR.

**Numerical Analysis Hilary Term 2021**
**Lecture 4: Least-squares problem**

So far the linear systems we treated had the same number of equations as unknowns (variables), so the problem was $Ax = b$ for a square matrix $A$. Very often in practice, we have more equations that we would like to satisfy than variables to fit them. It is then usually impossible to obtain $Ax = b$; a common approach is then to try minimise the difference between $Ax$ and $b$. If we choose to minimise the Euclidean length of the vector, this leads to a *least-squares problem*:

$$\min_x \|Ax - b\|, \qquad A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m, m \geq n. \tag{1}$$

Here $\|y\| := \sqrt{y_1^2 + y_2^2 + \cdots + y_m^2} = \sqrt{y^T y}$.

Least-squares problems (also known as *overdetermined* systems) are ubiquitous in applied mathematics and data science; linear regression is a basic example.

**Solution of least-squares by the QR factorisation:**

Let $A = [Q\ Q_\perp]\left[\begin{smallmatrix} R \\ 0 \end{smallmatrix}\right] = Q_F\left[\begin{smallmatrix} R \\ 0 \end{smallmatrix}\right]$ be a 'full' QR factorization, computed e.g. via the Householder QR factorization. We assume $R$ is nonsingular (i.e., $A$ has full column rank); this is a generic condition. Noting that $\|Q_F^T y\| = \sqrt{y^T Q_F Q_F^T y} = \sqrt{y^T y} = \|y\|$ for any vector $y$, we have

$$\|Ax - b\| = \|Q_F^T(Ax - b)\| = \left\| \begin{bmatrix} R \\ 0 \end{bmatrix} x - \begin{bmatrix} Q^T b \\ Q_\perp^T b \end{bmatrix} \right\|.$$

The bottom part is $-Q_\perp^T b$, no matter what $x$ is. The top part can be made 0 by taking $x = R^{-1}Q^T b$—this is therefore the solution.

The argument also suggests an algorithm: compute the "thin" QR factorization $A = QR$, then solve $Rx = Q^T b$ for $x$, which is obtained by backward substitution as $R$ is triangular. Note that while we used the full QR for the derivation, we only need the thin QR for the solution of (1).

Later we will see that a general linear least-squares problem has solution characterised by the orthogonality condition, which in our context reduces to $A^T(Ax - b) = 0$, so $x = (A^T A)^{-1} A^T b$; one can verify this is the same as $R^{-1}Q^T b$ obtained above.

**Illustration of least-squares for polynomial approximation:** We treated Lagrange interpolation in Lecture 1. While Lagrange polynomials give a clean expression for the interpolating polynomial, the interpolating polynomial is not always a good approximation to the original underlying function $f$. For example, suppose $f(x) = 1/(25x^2 + 1)$ (this is a famous function called the *Runge function*), and take a degree-$n$ polynomial interpolant $p_n$ at $n + 1$ equispaced points in $[-1, 1]$. The interpolating polynomials for varying $n$ are shown in Figure 1.

As we increase $n$, we hope that $p_n \to f$—but this is far from the truth! $p_n$ is diverging as $n$ grows near the endpoints $\pm 1$, and the divergence is actually exponential (very bad); note the vertical scales of the final plots! This is called Runge's phenomenon.

How can we avoid the divergence, and get $p_n \to f$ as we hope? One approach is to *oversample*: take (many) more points than the degree $n$. With $m(> n + 1)$ data

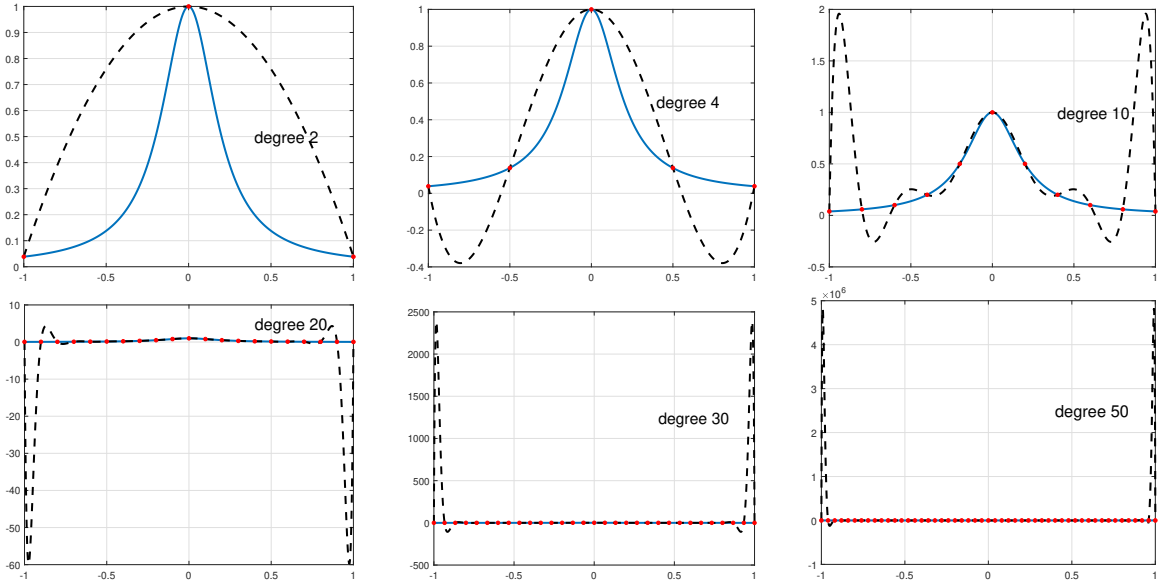



Figure 1: Polynomial interpolants (dashed black curves) of $f(x) = 1/(25x^2 + 1)$ (blue). The red dots are the interpolation points.

points $x_1, \ldots, x_m$, this will lead to the least-squares problem $\min_c \|Ac - b\|$, wherein $c = [c_0, c_1, \ldots, c_n]^T$ represents the coefficients of the polynomial $p_n(x) = \sum_{j=0}^{n} c_j x^j$, $A \in \mathbb{R}^{m\times(n+1)}$ with $A_{ij} = (x_i)^{j-1}$ and $b = [f(x_1), \ldots, f(x_m)]^T$.

We illustrate this in Figure 2 with the example above, but now fixing $n = 20$ and varying the number of data points $m$. This time, for large enough $m$ the polynomial $p_n$ is close to $f$ across the whole interval $[-1, 1]$.

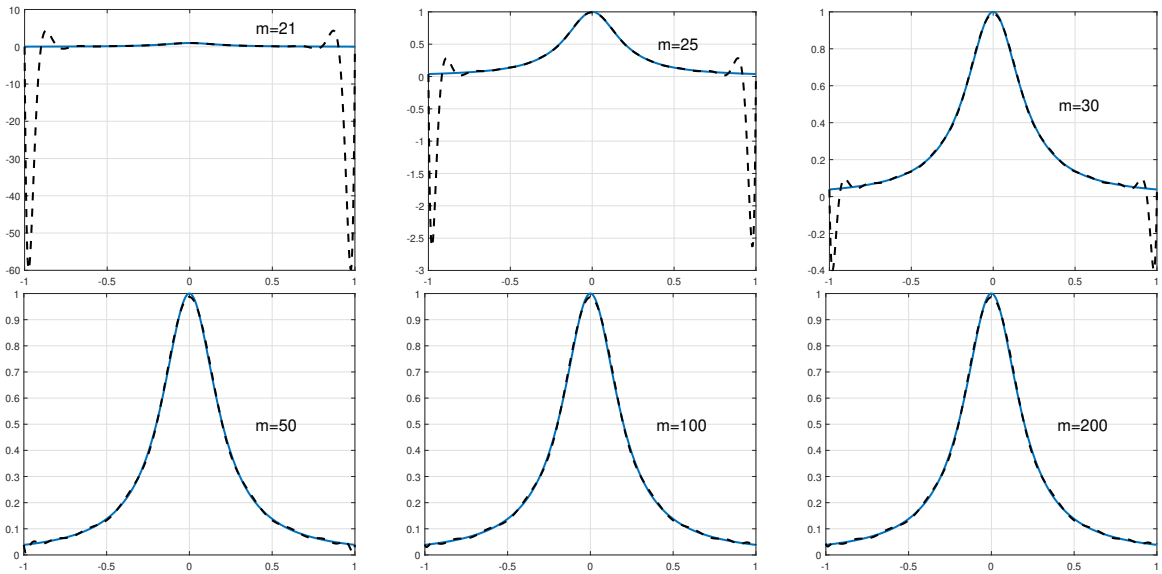



Figure 2: Least-squares polynomial fits of degree 20 (black dashed curves) of $f(x) = 1/(25x^2 + 1)$ (blue).

**Extensions and related facts (Non-examinable):**

- Instead of $p_n(x) = \sum_{j=0}^{n} c_j x^j$, it is actually much better to use a different polynomial basis involving *orthogonal polynomials* $\{\phi_i\}_{i=0}^{n}$ such as the Chebyshev polynomials, a topic discussed later. Then we would express $p_n(x) = \sum_{j=0}^{n} c_j \phi_j(x)$ and $A_{ij} = (\phi_{j-1}(x_i))$, and the least-squares problem will be beter-conditioned (easier to solve accurately). However, Runge's phenomenon still persists unless $m \gg n$.

- Note that we do not have $p_n \to f$ in Figure 2 as $m \to \infty$ because the polynomial degree $n = 20$ is fixed; to get $p_n \to f$ one needs to increase $n$ together with $m$. It can be shown that if one takes $m = n^2$, we do have $p_n \to f$ for any analytic function $f$ (the convergence is exponential in $n$).

- Another—more elegant—solution to overcome the instability in Figure 1 is to change the interpolation points. If one chooses them to be the so-called Chebyshev points $x_j = \cos(j\pi/n)$ for $j = 0, 1, \ldots, n$, the interpolating polynomial can be shown to be an excellent approximation to $f$, in fact nearly the best-possible polynomial approximation for any continuous $f$. This is a fundamental fact in approximation theory; for a rigorous and extended discussion (including an explanation of Runge's phenomenon), check out the Part C course Approximation of Functions.

**Underdetermined case (Non-examinable):** One might wonder, what if we have *fewer* equations than variables? That is, if we have $Ax = b$ with $A \in \mathbb{R}^{m\times n}$, $m < n$. This *underdetermined* system of equations has infinitely many solutions (if there is one). The natural question becomes, which one should we look for? One possibility is to find the minimum-norm solution minimize $\|x\|$ subject to $Ax = b$; the solution can be computed again via the QR factorization (of $A^T$). This problem has connections to the hot topic of *deep learning*. Another fascinating approach that has had enormous impact is to minimise the 1-norm $\|x\|_1$ subject to $Ax = b$, where $\|x\|_1 = \sum_{i=1}^{n} |x_i|$. This is the basis of the exciting field of *compressed sensing*.

**Numerical Analysis Hilary Term 2021**
**Lecture 5: Singular Value Decomposition**

We now introduce the Singular Value Decomposition (SVD), an extremely important matrix decomposition applicable to any matrix, including nonsymmetric and rectangular ones.

**Theorem.** (SVD) Every matrix $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ can be written as

$$A = U\Sigma V^T, \tag{1}$$

where $U \in \mathbb{R}^{m\times n}$ and $V \in \mathbb{R}^{n\times n}$ are matrices with orthonormal columns, i.e., $U^TU = I_n$ and $V^TV = I_n = VV^T$ ($V$ is square orthogonal; note that $UU^T \neq I_m$), and

$$\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix} \ (= \mathrm{diag}(\sigma_1, \ldots, \sigma_n))$$

is a diagonal matrix with nonnegative diagonal entries. In short, the SVD is a decomposition of $A$ into a product of 'orthonormal-diagonal-orthogonal' matrices; when $A$ is square $m = n$, 'orthogonal-diagonal-orthogonal'.

One can think of orthogonal matrices as a length-preserving rotation, so the SVD indicates that applying a matrix performs a rotation, followed by shrinkage or amplification of the elements, followed by another (different) rotation.

$\sigma_i$ are called the *singular values* and usually arranged in decreasing order $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. The columns of $U, V$ are called the (left and right) *singular vectors* of $A$. The *rank* of a matrix $A$ is the number of its positive singular values (this is equivalent e.g. to the number of linearly independent columns or rows).

**Proof.** Let's prove the existence of the SVD (1) by the following steps.

1. The matrix $A^TA \in \mathbb{R}^{n\times n}$ is symmetric. This is straightforward to verify, either by direct calculations or from the general identity $(XY)^T = Y^TX^T$.

2. The eigenvalues of $A^TA$ are all real and nonnegative (such matrices are called symmetric positive definite). To see this, suppose $A^TAx = \lambda x$, $x \neq 0$. Then $x^TA^TAx = \lambda x^Tx$, so $\lambda = \frac{x^TA^TAx}{x^Tx} = \frac{y^Ty}{x^Tx} \geq 0$, where $y = Ax$.

3. Let $A^TA = VD^2V^T$ be the symmetric eigenvalue decomposition, with $V \in \mathbb{R}^{n\times n}$ orthogonal and $D$ diagonal. Then let $B = AV$. Now $B^TB = D^2$ is a diagonal matrix, implying that the columns of $B$ are pairwise orthogonal.

4. Let's write $B^TB = D^2 = \mathrm{diag}(\lambda_1, \ldots, \lambda_r, 0, \ldots, 0)$, where $\lambda_r > 0$.

   (a) It is possible that $r = n$, and this is an important case (happens iff $\mathrm{rank}(A) = n$) where there is no 0 diagonal entry in $D^2$. We then have $D^{-1} = \mathrm{diag}(1/\sqrt{\lambda_1}, \ldots, 1/\sqrt{\lambda_r})$. Take $U := BD^{-1} = AVD^{-1}$, which has orthonormal columns $U^TU = I_n$. We are then done, as taking $\Sigma = D$, $A = U\Sigma V^T$.

(b) When $r < n$ (the rank-deficient case), $B$ has columns that are 0. Let $D_r = \text{diag}(\lambda_1, \ldots, \lambda_r)$. We still have $B \begin{bmatrix} D_r^{-1} & \\ & I_{n-r} \end{bmatrix} = [U_1,\ 0]$, and so

$$A = [U_1,\ 0] \begin{bmatrix} D_r & \\ & I_{n-r} \end{bmatrix} V^T = [U_1 U_2] \begin{bmatrix} D_r & \\ & 0 \end{bmatrix} V^T$$

for any $U_2$; we take it to be orthonormal $U_2^T U_2 = I_{n-r}$ and $U_2^T U_1 = 0$ ($U_2$ is any orthonormal matrix in the orthogonal complement of $U_1$; its existence can be verified e.g. using Householder reflectors). Taking $U = [U_1, U_2]$ completes the proof, again with $\Sigma = D$.

□

Some comments:

- Analogous to the full QR factorisation, there is a 'full SVD' $A = \tilde{U}\tilde{\Sigma}\tilde{V}^T$, where $\tilde{U} = [U\ U_\perp] \in \mathbb{R}^{m \times m}$ is orthogonal and $\tilde{\Sigma} \in \mathbb{R}^{m \times n} = \begin{bmatrix} \Sigma \\ 0_{(m-n)\times n} \end{bmatrix}$ and $\tilde{V} = V$. This can be obtained by starting from (1) and finding an orthogonal complement $U_\perp$ of $U$.
- Fat matrices: the assumption $m \geq n$ is just for convenience; if $m < n$, one still has $A = U\Sigma V^T$ where $\Sigma \in \mathbb{R}^{m \times m}$ is diagonal, $U \in \mathbb{R}^{m \times m}$ is orthogonal, and $V \in \mathbb{R}^{n \times m}$ has orthonormal columns. Below we continue with the assumption $m \geq n$.
- The SVD extends directly to matrices with nonreal entries: $A = U\Sigma V^*$, where $U, V$ are unitary matrices and $^*$ denotes the conjugate transpose.

**Matrix spectral norm** Let us briefly introduce the *spectral norm*[1] for matrices $A \in \mathbb{R}^{m \times n}$: $\|A\|_2 = \sigma_1(A)$, i.e., the largest singular value. It is a nonnegative scalar that measures 'how large' the matrix is. It has the equivalent characterisation $\|A\|_2 = \max_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2}$, where the norms in the right-hand side are the standard Euclidean norm (length) for vectors $\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$.

**Low-rank approximation** The SVD is useful for theoretical purposes, as it identifies e.g. the range (column space), null space, rank, and many more. In applications, the primary reason SVD is so important is that it gives the optimal low-rank approximation.

Let $A = U\Sigma V^T$ be the SVD and write $U = [u_1, \ldots, u_n], V = [v_1, \ldots, v_n]$, and define the "tall-skinny matrices" $U_k = [u_1, \ldots, u_k]$, $V_k = [v_1, \ldots, v_k]$, and $\Sigma_k = \text{diag}(\sigma_1, \ldots, \sigma_k)$. Let $k$ be any integer $k \leq n$. Then set

$$A_k = U_k \Sigma_k V_k^T = \sum_{i=1}^{k} \sigma_i u_i v_i^T.$$

Note that $\text{rank}(A_r) = r$. Also note that $A = \sum_{i=1}^{n} \sigma_i u_i v_i^T$, which is another way of expressing the SVD. $A_r$ is called the *truncated SVD* of $A$, as $A_r$ is obtained by truncating the trailing components of the SVD of $A$.

[1] Also known as the 2-norm or the operator norm. We return to the topic of norms later in the course.

We are now ready to state the result.

**Theorem.** Let $r \leq n$ be an integer. For any $B \in \mathbb{C}^{m\times n}$ with $\text{rank}(B) \leq r$,

$$\|A - A_r\|_2 = \sigma_{r+1} \leq \|A - B\|_2. \tag{2}$$

In other words, $A_r$ *is the best rank-r approximant to* $A$ in the spectral norm.

**Proof.** The first equality $\|A - A_r\|_2 = \sigma_{r+1}$ can be seen by noting that $A - A_r = \sum_{i=r+1}^{n} \sigma_i u_i v_i^T$ with singular values $\sigma_{r+1}, \ldots, \sigma_n$, along with $r$ 0's. For the inequality:

1. Since $\text{rank}(B) \leq r$, we can write $B = B_1 B_2^T$ where $B_1, B_2$ have $r$ columns. Therefore, there exists an orthonormal null space $W \in \mathbb{C}^{n\times(n-r)}$ s.t. $BW = 0$.

2. Then $\|A - B\|_2 \geq \|(A - B)W\|_2 = \|AW\|_2 = \|U\Sigma(V^T W)\|_2$. Now since $W$ is $(n - r)$-dimensional, there is an interesection between $W$ and $[v_1, \ldots, v_{r+1}]$, the $(r + 1)$-dimensional subspace spanned by the leading $r + 1$ left singular vectors ($[W, v_1, \ldots, v_{r+1}][x_1, x_2]^T = 0$ has a solution; then $Wx_1$ is such a vector).

3. Scale $x_1$ to have unit norm, and by orthogonal invariance $\|U\Sigma V^T W x_1\|_2 = \|\Sigma V^T W x_1\|_2 = \|\Sigma_{r+1} y_1\|_2$, where $\|y_1\|_2 = 1$ (b.c. $Wx_1$ lies in $\text{span}[v_1, \ldots, v_{r+1}]$) and $\Sigma_{r+1}$ is the leading $r + 1$ part of $\Sigma$.

4. Then $\|U\Sigma_{r+1} y_1\|_2 \geq \sigma_{r+1}$ can be verified by direct calculations.

□

In fact, more generally it is known that

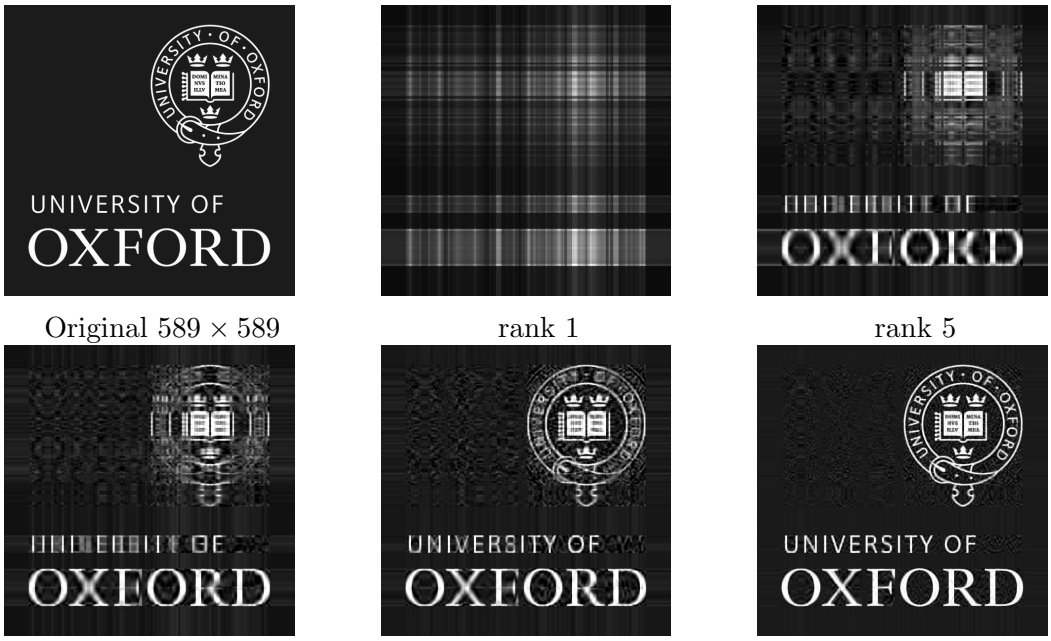
$$\|A - A_r\| \leq \|A - B\| \tag{3}$$

for any so-called unitarily invariant norm $\|\cdot\|$ (non-examinable).

In many applications $\sigma_{r+1} \ll \sigma_1$ for some $r \ll n$, in which case $A \approx U_r \Sigma_r V_r^T$. Now, storing $U_r, \Sigma_r, V_r$ requires $\approx (m + n + 1)r$ memory, as opposed to $mn$ for the full $A$, so when $r \ll \min(m, n)$, this can be used for data compression; this fact is used everywhere e.g. in data science!

**Illustration of low-rank approximation:** A traditional example to illustrate low-rank approximation via the truncated SVD is image compression. A grayscale image can be represented by a matrix $A$, with each entry representing the intensity of a pixel. One can then approximate $A$ by a truncated SVD, and use that to get a compressed image that hopefully looks similar to the original image to human eyes. Images tend to have structure that lends $A$ to be approximately low-rank.

Below we take an image of the Oxford logo, represent it as a matrix $A \in \mathbb{R}^{589\times 589}$ and compute its SVD (just `[U,S,V] = svd(A)` in MATLAB). Using the truncated SVD we then compute a rank-$r$ approximation for different values of $r$. With a rank-1 matrix the rows (and columns) are all parallel so the image is uninformative; but as $r$ increases the image becomes clear, and with rank 50 the image is almost indistinguishable from the original, while still giving some data compression. For larger images, such savings can be




Original $589 \times 589$ rank 1 rank 5

rank 10 rank 20 rank 50

Figure 1: The Oxford logo and its low-rank approximations via the truncated SVD.

significant. (This is however not how images are usually compressed in practice; e.g. the algorithm behind the jpg format is completely different).

The SVD $A = U\Sigma V^T$ and symmetric eigenvalue decomposition $A = V\Lambda V^T$ have many properties and results in common (e.g. Courant-Fisher min-max theorem; nonexaminable), stemming from the fact that they are both decompositions of the form "orthogonal-diagonal-orthogonal". In fact the SVD proof given above suggests an algorithm for computing the SVD via a symmetric eigenvalue decomposition of $A^T A$ (this is not exactly how the SVD is compute in practice, but this is outside the scope); we now turn to eigenvalue problems $Ax = \lambda x$ and describe an algorithm for solving them.

**Numerical Analysis Hilary Term 2021**
**Lecture 6: Matrix Eigenvalues**

We now turn to eigenvalue problems $Ax = \lambda x$, where $A \in \mathbb{R}^{n\times n}$ or $A \in \mathbb{C}^{n\times n}$, $\lambda \in \mathbb{C}$, and $x(\neq 0) \in \mathbb{C}^n$. Recall that there are $n$ eigenvalues in $\mathbb{C}$ (nonreal $\lambda$ possible even if $A$ is real). There are usually, but not always, $n$ linearly independent eigenvectors (e.g. Jordan block $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ has only one eigenvector $[1,0]^T$).

**Background:** An important result from analysis (not proved or examinable!), which will be useful.

**Theorem.** (Ostrowski) The eigenvalues of a matrix are continuously dependent on the entries. That is, suppose that $\{\lambda_i, i = 1,\dots,n\}$ and $\{\mu_i, i = 1,\dots,n\}$ are the eigenvalues of $A \in \mathbb{R}^{n\times n}$ and $A + B \in \mathbb{R}^{n\times n}$ respectively. Given any $\varepsilon > 0$, there is a $\delta > 0$ such that $|\lambda_i - \mu_i| < \varepsilon$ whenever $\max_{i,j} |b_{ij}| < \delta$, where $B = \{b_{ij}\}_{1\le i,j\le n}$.

Noteworthy properties related to eigenvalues:

- $A$ has $n$ eigenvalues (counting multiplicities), equal to the roots of the **characteristic polynomial** $p_A(\lambda) = \det(\lambda I - A)$.
- If $Ax_i = \lambda_i x_i$ for $i = 1,\dots,n$ and $x_i$ are linearly independent so that $[x_1, x_2, \dots, x_n] =: X$ is nonsingular, then $A$ has the **eigenvalue decomposition** $A = X\Lambda X^{-1}$. This usually, but not always, exist. The most general form is the Jordan canonical form (which we don't treat much in this course).
- Any square matrix has a **Schur decomposition** $A = QTQ^*$ where $Q$ is unitary $QQ^* = Q^*Q = I_n$, and $T$ triangular. The superscript $*$ denotes the (complex) conjugate transpose, $(Q^*)_{ij} = \overline{Q_{ji}}$.
- For a **normal matrix** s.t. $A^*A = AA^*$, the Schur decomposition shows $T$ is diagonal (proof: examine diagonal elements of $A^*A$ and $AA^*$), i.e., $A$ can be diagonalized by a unitary similarity transformation: $A = Q\Lambda Q^*$, where $\Lambda = \mathrm{diag}(\lambda_1,\dots,\lambda_n)$. Most of the structured matrices we treat are normal, including symmetric ($\lambda \in \mathbb{R}$), orthogonal ($|\lambda| = 1$), and skew-symmetric ($\lambda \in i\mathbb{R}$).

**Aim:** estimate the eigenvalues of a matrix.

**Theorem. Gerschgorin's theorem**: Suppose that $A = \{a_{ij}\}_{1\le i,j\le n} \in \mathbb{R}^{n\times n}$, and $\lambda$ is an eigenvalue of $A$. Then, $\lambda$ lies in the union of the **Gerschgorin discs**

$$D_i = \left\{ z \in \mathbb{C} \;\middle|\; |a_{ii} - z| \le \sum_{\substack{j\neq i \\ j=1}}^{n} |a_{ij}| \right\}, \quad i = 1,\dots,n.$$

**Proof.** If $\lambda$ is an eigenvalue of $A \in \mathbb{R}^{n\times n}$, then there exists an eigenvector $x \in \mathbb{R}^n$ with $Ax = \lambda x$, $x \neq 0$, i.e.,

$$\sum_{j=1}^{n} a_{ij}x_j = \lambda x_i, \quad i = 1,\dots,n.$$

Suppose that $|x_k| \geq |x_\ell|$, $\ell = 1, \ldots, n$, i.e.,

$$\text{“}x_k \text{ is the largest entry”}. \tag{1}$$

Then the $k$th row of $Ax = \lambda x$ gives $\sum_{j=1}^{n} a_{kj}x_j = \lambda x_k$, or

$$(a_{kk} - \lambda)x_k = -\sum_{\substack{j=1 \\ j\neq k}}^{n} a_{kj}x_j.$$

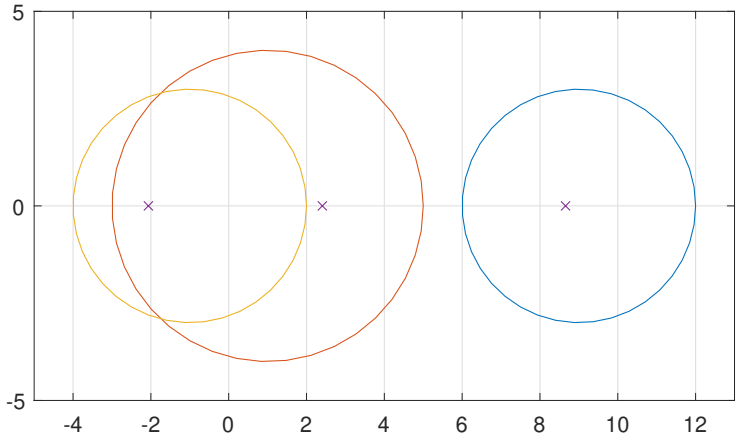
Dividing by $x_k$, (which, we know, is $\neq 0$) and taking absolute values,

$$|a_{kk} - \lambda| = \left| \sum_{\substack{j=1 \\ j\neq k}}^{n} a_{kj}\frac{x_j}{x_k} \right| \leq \sum_{\substack{j=1 \\ j\neq k}}^{n} |a_{kj}| \left|\frac{x_j}{x_k}\right| \leq \sum_{\substack{j=1 \\ j\neq k}}^{n} |a_{kj}|$$

by (1). □

**Example.**

$$A = \begin{bmatrix} 9 & 1 & 2 \\ -3 & 1 & 1 \\ 1 & 2 & -1 \end{bmatrix}$$




With Matlab calculate `>> eig(A) = 8.6573, -2.0639, 2.4066`

**Theorem. Gerschgorin's 2nd theorem:** If any union of $\ell$ (say) discs is disjoint from the other discs, then it contains $\ell$ eigenvalues.

**Proof.** Consider $B(\theta) = \theta A + (1-\theta)D$, where $D = \text{diag}(A)$, the diagonal matrix whose diagonal entries are those from $A$. As $\theta$ varies from 0 to 1, $B(\theta)$ has entries that vary continuously from $B(0) = D$ to $B(1) = A$. Hence the eigenvalues $\lambda(\theta)$ vary continuously by Ostrowski's theorem. The Gerschgorin discs of $B(0) = D$ are points (the diagonal entries), which are clearly the eigenvalues of $D$. As $\theta$ increases the Gerschgorin discs of $B(\theta)$ increase in radius about these same points as centres. Thus if $A = B(1)$ has a disjoint set of $\ell$ Gerschgorin discs by continuity of the eigenvalues it must contain exactly $\ell$ eigenvalues (as they can't jump!). □

**Numerical Analysis Hilary Term 2021**
**Lectures 7–8: Computing eigenvalues: The Symmetric QR Algorithm**

**Direct vs. Iterative Methods:** methods such as LU or QR factorisations and solving $Ax = b$ using them are *direct*: they compute a certain number of operations and then finish with "the answer". Another class of methods are **iterative**:

- construct a sequence;
- truncate that sequence "after convergence";
- typically concerned with fast convergence rate (rather than operation count).

Note that unlike LU, QR or linear systems $Ax = b$, algorithms for eigenvalues are necessarily iterative: By Galois theory, no finite algorithm can compute eigenvalues of $n \times n (\geq 5)$ matrices exactly in a finite number of operations. We still have an incredibly reliable algorithm to compute them, essentially to full accuracy (for symmetric matrices; for nonsymmetric matrices, in a "backward stable" manner; this is outside the scope).

**Notation:** for $x \in \mathbb{R}^n$, $\|x\| = \sqrt{x^{\mathrm{T}}x}$ is the (Euclidean) length of $x$.

**Notation:** in iterative methods, $x_k$ usually means the vector $x$ at the $k$th iteration (rather than $k$th entry of vector $x$). Some sources use $x^k$ or $x^{(k)}$ instead.

**Power Iteration:** a simple method for calculating a single (largest) eigenvalue of a square matrix $A$ (and its associated eigenvector). For arbitrary $y \in \mathbb{R}^n$, set $x_0 = y/\|y\|$ to calculate an initial vector, and then for $k = 0, 1, \ldots$

Compute $y_k = Ax_k$
and set $x_{k+1} = y_k/\|y_k\|$.

This is the **Power Method** or **Power Iteration**, and computes unit vectors in the direction of $x_0, Ax_0, A^2x_0, A^3x_0, \ldots, A^kx_0$.

Suppose that $A$ is diagonalizable so that there is a basis of eigenvectors of $A$:

$$\{v_1, v_2, \ldots, v_n\}$$

with $Av_i = \lambda_i v_i$ and $\|v_i\| = 1$, $i = 1, 2, \ldots, n$, and assume that

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|.$$

Then we can write

$$x_0 = \sum_{i=1}^{n} \alpha_i v_i$$

for some $\alpha_i \in \mathbb{R}$, $i = 1, 2, \ldots, n$, so

$$A^k x_0 = A^k \sum_{i=1}^{n} \alpha_i v_i = \sum_{i=1}^{n} \alpha_i A^k v_i.$$

However, since $Av_i = \lambda_i v_i \implies A^2 v_i = A(Av_i) = \lambda_i A v_i = \lambda_i^2 v_i$, inductively $A^k v_i = \lambda_i^k v_i$. So

$$A^k x_0 = \sum_{i=1}^{n} \alpha_i \lambda_i^k v_i = \lambda_1^k \left[ \alpha_1 v_1 + \sum_{i=2}^{n} \alpha_i \left( \frac{\lambda_i}{\lambda_1} \right)^k v_i \right].$$

Since $(\lambda_i/\lambda_1)^k \to 0$ as $k \to \infty$, $A^k x_0$ tends to look like $\lambda_1^k \alpha_1 v_1$ as $k$ gets large. The result is that by normalizing to be a unit vector

$$\frac{A^k x_0}{\|A^k x_0\|} \to \pm v_1 \quad \text{and} \quad \frac{\|A^k x_0\|}{\|A^{k-1} x_0\|} \approx \left| \frac{\lambda_1^k \alpha_1}{\lambda_1^{k-1} \alpha_1} \right| = |\lambda_1|$$

as $k \to \infty$, and the sign of $\lambda_1$ is identified by looking at, e.g., $(A^k x_0)_1/(A^{k-1} x_0)_1$.

Essentially the same argument works when we normalize at each step: the Power Iteration may be seen to compute $y_k = \beta_k A^k x_0$ for some $\beta_k$. Then, from the above,

$$x_{k+1} = \frac{y_k}{\|y_k\|} = \frac{\beta_k}{|\beta_k|} \cdot \frac{A^k x_0}{\|A^k x_0\|} \to \pm v_1.$$

Similarly, $y_{k-1} = \beta_{k-1} A^{k-1} x_0$ for some $\beta_{k-1}$. Thus

$$x_k = \frac{\beta_{k-1}}{|\beta_{k-1}|} \cdot \frac{A^{k-1} x_0}{\|A^{k-1} x_0\|} \quad \text{and hence} \quad y_k = A x_k = \frac{\beta_{k-1}}{|\beta_{k-1}|} \cdot \frac{A^k x_0}{\|A^{k-1} x_0\|}.$$

Therefore, as above,

$$\|y_k\| = \frac{\|A^k x_0\|}{\|A^{k-1} x_0\|} \approx |\lambda_1|,$$

and the sign of $\lambda_1$ may be identified by looking at, e.g., $(x_{k+1})_1/(x_k)_1$.

Hence the largest eigenvalue (and its eigenvector) can be found.

Note: it is unlikely but possible for a chosen vector $x_0$ that $\alpha_1 = 0$, but rounding errors in the computation generally introduce a small component in $v_1$, so that in practice this is not a concern!

This simplified method for eigenvalue computation is the basis for effective methods, but the current state of the art is the **QR Algorithm** which was invented by John Francis in London in 1959/60. As we shall see, the mechanics of QR algorithm is very much related to the power method.

**The QR algorithm:** We now describe the QR algorithm, a magical algorithm that can solve eigenvalue problems $Ax = \lambda x$.

For simplicity we consider the algorithm only in the case when $A$ is symmetric, but it is applicable also to nonsymmetric matrices with minor modifications.

**Recall:** a symmetric matrix $A$ is similar to $B$ if there is a nonsingular matrix $P$ for which $A = P^{-1}BP$. Similar matrices have the same eigenvalues, since if $A = P^{-1}BP$,

$$0 = \det(A - \lambda I) = \det(P^{-1}(B - \lambda I)P) = \det(P^{-1}) \det(P) \det(B - \lambda I),$$

so $\det(A - \lambda I) = 0$ if, and only if, $\det(B - \lambda I) = 0$.

The basic **QR algorithm** is:

`Set` $A_1 = A$.
`for` $k = 1, 2, \ldots$
  `form the QR factorization` $A_k = Q_k R_k$
  `and set` $A_{k+1} = R_k Q_k$

```
end
```

**Proposition.** The symmetric matrices $A_1, A_2, \ldots, A_k, \ldots$ are all similar and thus have the same eigenvalues.

**Proof.** Since

$$A_{k+1} = R_k Q_k = (Q_k^{\mathrm{T}} Q_k) R_k Q_k = Q_k^{\mathrm{T}} (Q_k R_k) Q_k = Q_k^{\mathrm{T}} A_k Q_k = Q_k^{-1} A_k Q_k,$$

$A_{k+1}$ is symmetric if $A_k$ is, and is similar to $A_k$. □

At least when $A$ has eigenvalues of distinct modulus $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$, this basic QR algorithm can be shown to work ($A_k$ converges to a diagonal matrix as $k \to \infty$, the diagonal entries of which are the eigenvalues). To see this, we make the following observations.

**Lemma.**

$$A_{k+1} = (Q^{(k)})^T A Q^{(k)}. \tag{1}$$

(Note 18/2/2021: corrected from $A_k = (Q^{(k)})^T A Q^{(k)}$) and

$$A^k = (Q_1 \cdots Q_k)(R_k \cdots R_1) = Q^{(k)} R^{(k)} \tag{2}$$

is the QR factorization of $A^k$.

**Proof.** (1) follows from a repeated application of the above proposition.

We use induction for (2): $k = 1$ trivial. Suppose $A^{k-1} = Q^{(k-1)} R^{(k-1)}$. Then $A_k = R_{k-1} Q_{k-1} = (Q^{(k-1)})^T A Q^{(k-1)}$, and

$$(Q^{(k-1)})^T A Q^{(k-1)} = Q_k R_k.$$

Then $A Q^{(k-1)} = Q^{(k-1)} Q_k R_k$, and so

$$A^k = A Q^{(k-1)} R^{(k-1)} = Q^{(k-1)} Q_k R_k R^{(k-1)} = Q^{(k)} R^{(k)},$$

giving (2). □

Let us now connect the above lemma with the power method.

**Lemma.** With $Q^{(k)}$ as in (2), let $q_1$ be its first column, and let $e_1 = [1, 0, \ldots, 0]^T$. Then $q_1$ is equal to either $\frac{A^k e_1}{\|A^k e_1\|_2}$ or $-\frac{A^k e_1}{\|A^k e_1\|_2}$.

**Proof.** Right-multiplying $e_1$ to (2) yields $A^k e_1 = Q^{(k)} R^{(k)} e_1$. Since $R^{(k)}$ is upper triangular $R^{(k)} e_1 = [R^{(k)}_{1,1}, 0, \ldots, 0]^T$, and so $Q^{(k)} R^{(k)} e_1$ is parallel to $q_1$, which has unit norm. □

The results show in particular that the first column $q_1$ of $Q^{(k)}$ is the result of power method applied $k$ times to the initial vector $e_1 = [1, 0, \ldots, 0]^T$. It then follows that $q_1$ converges to the dominant eigenvector. The second vector then starts converging to the 2nd dominant eigenvector, and so on. Once the columns of $Q^{(k)}$ converge to eigenvectors (note that they are orthogonal by design), (1) shows that $A_k$ converge to a diagonal matrix of eigenvalues.

However, a really practical, fast algorithm is based on some refinements.

**Reduction to tridiagonal form:** the idea is to apply explicit similarity transformations $QAQ^{-1} = QAQ^{\mathrm{T}}$, with $Q$ orthogonal, so that $QAQ^{\mathrm{T}}$ is tridiagonal.

Note: direct reduction to triangular form would reveal the eigenvalues, but is not possible. If

$$H(w)A = \begin{bmatrix} \times & \times & \cdots & \times \\ 0 & \times & \cdots & \times \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \times & \cdots & \times \end{bmatrix}$$

then $H(w)AH(w)^{\mathrm{T}}$ is generally full, i.e., all zeros created by pre-multiplication are destroyed by the post-multiplication. However, if

$$A = \begin{bmatrix} \gamma & u^{\mathrm{T}} \\ u & C \end{bmatrix}$$

(as $A = A^{\mathrm{T}}$) and

$$w = \begin{bmatrix} 0 \\ \hat{w} \end{bmatrix} \quad \text{where} \quad H(\hat{w})u = \begin{bmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

it follows that

$$H(w)A = \begin{bmatrix} \gamma & & u^{\mathrm{T}} & \\ \alpha & \times & \vdots & \times \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \times & \vdots & \times \end{bmatrix},$$

i.e., the $u^{\mathrm{T}}$ part of the first row of $A$ is unchanged. However, then

$$H(w)AH(w)^{-1} = H(w)AH(w)^{\mathrm{T}} = H(w)AH(w) = \left[\begin{array}{c|ccc} \gamma & \alpha & 0 & \cdots & 0 \\ \hline \alpha & & & & \\ 0 & & & & \\ \vdots & & B & & \\ 0 & & & & \end{array}\right],$$

where $B = H(\hat{w})CH^{\mathrm{T}}(\hat{w})$, as $u^{\mathrm{T}}H(\hat{w})^{\mathrm{T}} = (\alpha, \quad 0, \quad \cdots, \quad 0)$; note that $H(w)AH(w)^{\mathrm{T}}$ is symmetric as $A$ is.

Now we inductively apply this to the smaller matrix $B$, as described for the QR factorization but using post- as well as pre-multiplications. The result of $n-2$ such Householder similarity transformations is the matrix

$$H(w_{n-2}) \cdots H(w_2)H(w)AH(w)H(w_2) \cdots H(w_{n-2}),$$

which is tridiagonal.

The QR factorization of a tridiagonal matrix can now easily be achieved with $n-1$ *Givens* rotations $J(i,j)$; these are orthogonal matrices that are $I$ except for the four elements: the $(i,i),(i,j),(j,i),(j,j)$ entries with values $c, s, -s, c$ respectively, where $c^2 + s^2 = 1$

(cosine and sine); one can choose $c$ s.t. $\begin{bmatrix} c & s \\ -s & c \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sqrt{a^2+b^2} \\ 0 \end{bmatrix}$. (The operations below can be done with Householder matrices too, but Givens rotations are more straightforward).

Now if $A$ is tridiagonal

$$\underbrace{J(n-1,n)\cdots J(2,3)J(1,2)}_{Q^{\mathrm{T}}}A = R, \quad \text{upper triangular.}$$

Precisely, $R$ has a diagonal and 2 super-diagonals,

$$R = \begin{bmatrix} \times & \times & \times & 0 & 0 & 0 & \cdots & 0 \\ 0 & \times & \times & \times & 0 & 0 & \cdots & 0 \\ 0 & 0 & \times & \times & \times & 0 & \cdots & 0 \\ \vdots & \vdots & & & & & & \vdots \\ 0 & 0 & 0 & 0 & \times & \times & \times & 0 \\ 0 & 0 & 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \times \end{bmatrix}$$

(exercise: check!). In the QR algorithm, the next matrix in the sequence is $RQ$.

**Lemma.** In the QR algorithm applied to a symmetric tridiagonal matrix, the symmetry and tridiagonal form are preserved when Givens rotations are used.

**Proof.** We have already shown that if $A_k = QR$ is symmetric, then so is $A_{k+1} = RQ$. If $A_k = QR = J(1,2)^{\mathrm{T}}J(2,3)^{\mathrm{T}}\cdots J(n-1,n)^{\mathrm{T}}R$ is tridiagonal, then $A_{k+1} = RQ = RJ(1,2)^{\mathrm{T}}J(2,3)^{\mathrm{T}}\cdots J(n-1,n)^{\mathrm{T}}$. Recall that post-multiplication of a matrix by $J(i,i+1)^{\mathrm{T}}$ replaces columns $i$ and $i+1$ by linear combinations of the pair of columns, while leaving columns $j = 1, 2, \ldots, i-1, i+2, \ldots, n$ alone. Thus, since $R$ is upper triangular, the only subdiagonal entry in $RJ(1,2)^{\mathrm{T}}$ is in position $(2,1)$. Similarly, the only subdiagonal entries in $RJ(1,2)^{\mathrm{T}}J(2,3)^{\mathrm{T}} = (RJ(1,2)^{\mathrm{T}})J(2,3)^{\mathrm{T}}$ are in positions $(2,1)$ and $(3,2)$. Inductively, the only subdiagonal entries in

$$\begin{aligned} &RJ(1,2)^{\mathrm{T}}J(2,3)^{\mathrm{T}}\cdots J(i-2,i-1)^{\mathrm{T}}J(i-1,i)^{\mathrm{T}} \\ &\quad = (RJ(1,2)^{\mathrm{T}}J(2,3)^{\mathrm{T}}\cdots J(i-2,i-1)^{\mathrm{T}})J(i-1,i)^{\mathrm{T}} \end{aligned}$$

are in positions $(j, j-1)$, $j = 2, \ldots i$. So, the lower triangular part of $A_{k+1}$ only has nonzeros on its first subdiagonal. However, then since $A_{k+1}$ is symmetric, it must be tridiagonal. □

**Using shifts.** One further and final step in making an efficient algorithm is the use of **shifts**:

`for` $k = 1, 2, \ldots$
  `form the QR factorization of` $A_k - \mu_k I = Q_k R_k$
  `and set` $A_{k+1} = R_k Q_k + \mu_k I$

`end`

For any chosen sequence of values of $\mu_k \in \mathbb{R}$, $\{A_k\}_{k=1}^{\infty}$ are symmetric and tridiagonal if $A_1$ has these properties, and similar to $A_1$.

The simplest shift to use is $a_{n,n}$, which leads rapidly in almost all cases to

$$A_k = \left[\begin{array}{c|c} T_k & 0 \\ \hline 0^{\mathrm{T}} & \lambda \end{array}\right],$$

where $T_k$ is $n-1$ by $n-1$ and tridiagonal, and $\lambda$ is an eigenvalue of $A_1$. Inductively, once this form has been found, the QR algorithm with shift $a_{n-1,n-1}$ can be concentrated only on the $n-1$ by $n-1$ leading submatrix $T_k$. This process is called **deflation**.

Why does introducing shifts help? To understand this we establish a connection between QR and the power method applied to the *inverse* (known as the *inverse power method*).

**Lemma.** With $Q^{(k)}$ as in (2), denote by $q_n$ its last column, and let $e_n = [0, 0, \ldots, 1]^T$. Then $q_n$ is equal to either $\frac{A^{-k}e_n}{\|A^{-k}e_1\|_2}$ or $-\frac{A^{-k}e_n}{\|A^{-k}e_1\|_2}$.

**Proof.** Recall (2), and take the inverse:

$$A^{-k} = (R^{(k)})^{-1}(Q^{(k)})^T,$$

and take the transpose:

$$(A^{-k})^T (= A^{-k}) = Q^{(k)}(R^{(k)})^{-T}.$$

Now multiplying $e_n$ gives

$$A^{-k}e_n = Q^{(k)}(R^{(k)})^{-T}e_n.$$

Since $(R^{(k)})^{-T}$ is lower triangular, it follows that $Q^{(k)}(R^{(k)})^{-T}e_n$ is parallel to $q_n$. □

This shows that the **final** column of $Q^{(k)}$ is the result of power method applied to $e_n$ now with the **inverse** $A^{-1}$. Thus the last column of $Q^{(k)}$ is converging to the eigenvector for the smallest eigenvalue $\lambda_n$, with convergence factor $|\frac{\lambda_n}{\lambda_{n-1}}|$; $Q^{(k)}$ is converging not only from the first, but (more significantly) from the last column(s).

Now we see how the introduction of shift has a drastic effect on the convergence: it changes the factor to $|\frac{\lambda_{\sigma(n)}-\mu}{\lambda_{\sigma(n-1)}-\mu}|$, where $\sigma$ is a permutation such that $|\lambda_{\sigma(1)} - \mu| \geq |\lambda_{\sigma(2)} - \mu| \geq \cdots \geq |\lambda_{\sigma(n)} - \mu|$. If $\mu$ is close to an eigenvalue, this implies (potentially extremely) fast convergence; in fact by choosing the shift $\mu_k = a_{n,n}$, it can be shown that (proof omitted and non-examinable) $a_{m,m-1}$ converges *cubically*: $|a_{m,m-1,k+1}| = O(|a_{m,m-1,k}|^3)$.

**The overall algorithm** for calculating the eigenvalues of an $n$ by $n$ symmetric matrix:

`reduce` $A$ `to tridiagonal form by orthogonal`
`(Householder) similarity transformations.`

`for` $m = n, n-1, \ldots 2$
  `while` $a_{m-1,m} >$ `tol`
    $[Q, R] =$ `qr`$(A - a_{m,m}I)$
    $A = RQ + a_{m,m}I$
  `end while`
  `record eigenvalue` $\lambda_m = a_{m,m}$

$A \leftarrow$ `leading` $m-1$ `by` $m-1$ `submatrix of` $A$
`end`
`record eigenvalue` $\lambda_1 = a_{1,1}$

**Computing roots of polynomials via eigenvalues** Let us describe a nice application of computing eigenvalues (by the QR algorithm). Let $p(x) = \sum_{i=0}^{n} c_i x^i$ be a degree-$n$ polynomial so that $c_n \neq 0$, and suppose we want to find its roots, i.e., values of $\lambda$ for which $p(\lambda) = 0$; there are $n$ of them in $\mathbb{C}$. For example, $p(x)$ might be an approximant to data, obtained by Lagrange interpolation from the first lecture. Why roots? For example, you might be interested in the minimum of $p$; this can be obtained by differentiating and setting to zero $p'(x) = 0$, which is again a polynomial rootfinding problem (for $p'$).

How do we solve $p(x) = 0$? Recall that eigenvalues of $A$ are the roots of its characteristic polynomial. Here we take the opposite direction—construct a matrix whose characteristic polynomial is $p$.

Consider the following matrix, which is called the **companion matrix** (the blank elements are all 0) for the polynomial $p(x) = \sum_{i=0}^{n} c_i x^i$:

$$C = \begin{bmatrix} -\frac{c_{n-1}}{c_n} & -\frac{c_{n-2}}{c_n} & \cdots & -\frac{c_1}{c_n} & -\frac{c_0}{c_n} \\ 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & 0 \end{bmatrix}. \tag{3}$$

Then direct calculation shows that if $p(\lambda) = 0$ then $Cx = \lambda x$ with $x = [\lambda^{n-1}, \lambda^{n-2}, \ldots, \lambda, 1]^T$. Indeed one can show that the characteristic polynomial is $\det(\lambda I - C) = p(\lambda)/c_n$ (nonexaminable), so this implication is necessary and sufficient, so the eigenvalues of $C$ are precisely the roots of $p$, counting multiplicities.

Thus to compute roots of polynomials, one can compute eigenvalues of the companion matrix via the QR algorithm—this turns out to be a very powerful idea!

**Numerical Analysis Hilary Term 2021**
**Lecture 9: Best Approximation in Inner-Product Spaces**

**Best approximation of functions:** given a function $f$ on $[a, b]$, find the "closest" polynomial/piecewise polynomial (see later sections)/ trigonometric polynomial (truncated Fourier series).

**Norms:** are used to measure the size of/distance between elements of a vector space. Given a vector space $V$ over the field $\mathbb{R}$ of real numbers, the mapping $\|\cdot\| : V \to \mathbb{R}$ is a **norm** on $V$ if it satisfies the following axioms:
**(i)** $\|f\| \geq 0$ for all $f \in V$, with $\|f\| = 0$ if, and only if, $f = 0 \in V$;
**(ii)** $\|\lambda f\| = |\lambda|\|f\|$ for all $\lambda \in \mathbb{R}$ and all $f \in V$; and
**(iii)** $\|f + g\| \leq \|f\| + \|g\|$ for all $f, g \in V$ (the **triangle inequality**).

**Examples:** **1.** For vectors $x \in \mathbb{R}^n$, with $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$,

$$\|x\| \equiv \|x\|_2 = (x_1^2 + x_2^2 + \cdots + x_n^2)^{\frac{1}{2}} = \sqrt{x^{\mathrm{T}}x}$$

is the $\ell^2$- or vector two-norm.
**2.** For continuous functions on $[a, b]$,

$$\|f\| \equiv \|f\|_\infty = \max_{x\in[a,b]} |f(x)|$$

is the $\mathrm{L}^\infty$- or $\infty$-norm.
**3.** For integrable functions on $(a, b)$,

$$\|f\| \equiv \|f\|_1 = \int_a^b |f(x)| \, \mathrm{d}x$$

is the $\mathrm{L}^1$- or one-norm.
**4.** For functions in

$$V = \mathrm{L}^2_w(a, b) \equiv \{f : [a, b] \to \mathbb{R} \mid \int_a^b w(x)[f(x)]^2 \, \mathrm{d}x < \infty\}$$

for some given **weight** function $w(x) > 0$ (this certainly includes continuous functions on $[a, b]$, and piecewise continuous functions on $[a, b]$ with a finite number of jump-discontinuities),

$$\|f\| \equiv \|f\|_2 = \left( \int_a^b w(x)[f(x)]^2 \, \mathrm{d}x \right)^{\frac{1}{2}}$$

is the $\mathrm{L}^2$- or two-norm—the space $\mathrm{L}^2(a, b)$ is a common abbreviation for $\mathrm{L}^2_w(a, b)$ for the case $w(x) \equiv 1$.
**Note:** $\|f\|_2 = 0 \implies f = 0$ almost everywhere on $[a, b]$. We say that a certain property $\mathsf{P}$ holds *almost everywhere* (a.e.) on $[a, b]$ if property $\mathsf{P}$ holds at each point of $[a, b]$ except perhaps on a subset $S \subset [a, b]$ of zero measure. We say that a set $S \subset \mathbb{R}$ has *zero measure* (or that it is of *measure zero*) if for any $\varepsilon > 0$ there exists a sequence $\{(\alpha_i, \beta_i)\}_{i=1}^\infty$ of subintervals of $\mathbb{R}$ such that

$S \subset \cup_{i=1}^{\infty}(\alpha_i, \beta_i)$ and $\sum_{i=1}^{\infty}(\beta_i - \alpha_i) < \varepsilon$. Trivially, the empty set $\emptyset(\subset \mathbb{R})$ has zero measure. Any finite subset of $\mathbb{R}$ has zero measure. Any countable subset of $\mathbb{R}$, such as the set of all natural numbers $\mathbb{N}$, the set of all integers $\mathbb{Z}$, or the set of all rational numbers $\mathbb{Q}$, is of measure zero.

**Least-squares polynomial approximation:** aim to find the best polynomial approximation to $f \in \mathrm{L}^2_w(a,b)$, i.e., find $p_n \in \Pi_n$ for which

$$\|f - p_n\|_2 \leq \|f - q\|_2 \qquad \forall q \in \Pi_n.$$

Seeking $p_n$ in the form $p_n(x) = \sum_{k=0}^{n} \alpha_k x^k$ then results in the minimization problem

$$\min_{(\alpha_0,\ldots,\alpha_n)} \int_a^b w(x) \left[ f(x) - \sum_{k=0}^{n} \alpha_k x^k \right]^2 \mathrm{d}x.$$

The unique minimizer can be found from the (linear) system

$$\frac{\partial}{\partial \alpha_j} \int_a^b w(x) \left[ f(x) - \sum_{k=0}^{n} \alpha_k x^k \right]^2 \mathrm{d}x = 0 \ \text{ for each } \ j = 0, 1, \ldots, n,$$

but there is important additional structure here.

**Inner-product spaces:** a real **inner-product space** is a vector space $V$ over $\mathbb{R}$ with a mapping $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ (the **inner product**) for which
**(i)** $\langle v, v \rangle \geq 0$ for all $v \in V$ and $\langle v, v \rangle = 0$ if, and only if $v = 0$;
**(ii)** $\langle u, v \rangle = \langle v, u \rangle$ for all $u, v \in V$; and
**(iii)** $\langle \alpha u + \beta v, z \rangle = \alpha \langle u, z \rangle + \beta \langle v, z \rangle$ for all $u, v, z \in V$ and all $\alpha, \beta \in \mathbb{R}$.

**Examples:** **1.** $V = \mathbb{R}^n$,

$$\langle x, y \rangle = x^{\mathrm{T}} y = \sum_{i=1}^{n} x_i y_i,$$

where $x = (x_1, \ldots, x_n)^{\mathrm{T}}$ and $y = (y_1, \ldots, y_n)^{\mathrm{T}}$.
**2.** $V = \mathrm{L}^2_w(a,b) = \{ f : (a,b) \to \mathbb{R} \mid \int_a^b w(x)[f(x)]^2 \, \mathrm{d}x < \infty \}$,

$$\langle f, g \rangle = \int_a^b w(x) f(x) g(x) \, \mathrm{d}x,$$

where $f, g \in \mathrm{L}^2_w(a,b)$ and $w$ is a weight-function, defined, positive and integrable on $(a,b)$.

**Notes:** **1.** Suppose that $V$ is an inner product space, with inner product $\langle \cdot, \cdot \rangle$. Then $\langle v, v \rangle^{\frac{1}{2}}$ defines a norm on $V$ (see the final paragraph on the last page for a proof). In Example 2 above, the norm defined by the inner product is the (weighted) $\mathrm{L}^2$-norm.
**2.** Suppose that $V$ is an inner product space, with inner product $\langle \cdot, \cdot \rangle$, and let $\| \cdot \|$ denote the norm defined by the inner product via $\|v\| = \langle v, v \rangle^{\frac{1}{2}}$, for $v \in V$. The angle $\theta$ between $u, v \in V$ is

$$\theta = \cos^{-1} \left( \frac{\langle u, v \rangle}{\|u\| \|v\|} \right).$$

Thus $u$ and $v$ are orthogonal in $V \iff \langle u, v\rangle = 0$.

E.g., $x^2$ and $\frac{3}{4} - x$ are orthogonal in $\mathrm{L}^2(0,1)$ with inner product $\langle f, g\rangle = \displaystyle\int_0^1 f(x)g(x)\,\mathrm{d}x$ as

$$\int_0^1 x^2 \left(\tfrac{3}{4} - x\right)\,\mathrm{d}x = \tfrac{1}{4} - \tfrac{1}{4} = 0.$$

**3. Pythagoras Theorem:** Suppose that $V$ is an inner-product space with inner product $\langle\cdot,\cdot\rangle$ and norm $\|\cdot\|$ defined by this inner product. For any $u, v \in V$ such that $\langle u, v\rangle = 0$ we have

$$\|u \pm v\|^2 = \|u\|^2 + \|v\|^2.$$

**Proof.**

$$\begin{aligned}
\|u \pm v\|^2 &= \langle u \pm v, u \pm v\rangle = \langle u, u \pm v\rangle \pm \langle v, u \pm v\rangle && \text{[axiom (iii)]}\\
&= \langle u, u \pm v\rangle \pm \langle u \pm v, v\rangle && \text{[axiom (ii)]}\\
&= \langle u, u\rangle \pm \langle u, v\rangle \pm \langle u, v\rangle + \langle v, v\rangle\\
&= \langle u, u\rangle + \langle v, v\rangle && \text{[orthogonality]}\\
&= \|u\|^2 + \|v\|^2.
\end{aligned}$$

**4.** The **Cauchy–Schwarz inequality**: Suppose that $V$ is an inner-product space with inner product $\langle\cdot,\cdot\rangle$ and norm $\|\cdot\|$ defined by this inner product. For any $u, v \in V$,

$$|\langle u, v\rangle| \leq \|u\|\|v\|.$$

**Proof.** For every $\lambda \in \mathbb{R}$,

$$0 \leq \langle u - \lambda v, u - \lambda v\rangle = \|u\|^2 - 2\lambda\langle u, v\rangle + \lambda^2\|v\|^2 = \phi(\lambda),$$

which is a quadratic in $\lambda$. The minimizer of $\phi$ is at $\lambda_* = \langle u, v\rangle/\|v\|^2$, and thus since $\phi(\lambda_*) \geq 0$, $\|u\|^2 - \langle u, v\rangle^2/\|v\|^2 \geq 0$, which gives the required inequality. □

**5.** The **triangle inequality**: Suppose that $V$ is an inner-product space with inner product $\langle\cdot,\cdot\rangle$ and norm $\|\cdot\|$ defined by this inner product. For any $u, v \in V$,

$$\|u + v\| \leq \|u\| + \|v\|.$$

**Proof.** Note that

$$\|u + v\|^2 = \langle u + v, u + v\rangle = \|u\|^2 + 2\langle u, v\rangle + \|v\|^2.$$

Hence, by the Cauchy–Schwarz inequality,

$$\|u + v\|^2 \leq \|u\|^2 + 2\|u\|\|v\| + \|v\|^2 = \left(\|u\| + \|v\|\right)^2.$$

Taking square-roots yields

$$\|u + v\| \leq \|u\| + \|v\|.$$

□

**Note:** The function $\|\cdot\| : V \to \mathbb{R}$ defined by $\|v\| := \langle v, v\rangle^{\frac{1}{2}}$ on the inner-product space $V$, with inner product $\langle\cdot,\cdot\rangle$, trivially satisfies the first two axioms of norm on $V$; this is a

consequence of $\langle \cdot, \cdot \rangle$ being an inner product on $V$. Result 5 above implies that $\| \cdot \|$ also satisfies the third axiom of norm, the triangle inequality.

**Least-Squares Approximation**

For the problem of least-squares approximation, $\langle f, g \rangle = \int_a^b w(x) f(x) g(x) \, \mathrm{d}x$ and $\|f\|_2^2 = \langle f, f \rangle$ where $w(x) > 0$ on $(a, b)$.

**Theorem.** If $f \in \mathrm{L}_w^2(a, b)$ and $p_n \in \Pi_n$ is such that

$$\langle f - p_n, r \rangle = 0 \qquad \forall r \in \Pi_n, \tag{1}$$

then

$$\|f - p_n\|_2 \leq \|f - r\|_2 \qquad \forall r \in \Pi_n,$$

i.e., $p_n$ is a best (weighted) least-squares approximation to $f$ on $[a, b]$.

**Proof.**

$$\begin{aligned} \|f - p_n\|_2^2 = \; & \langle f - p_n, f - p_n \rangle \\ = \; & \langle f - p_n, f - r \rangle + \langle f - p_n, r - p_n \rangle \qquad \forall r \in \Pi_n \\ & \text{Since } r - p_n \in \Pi_n \text{ the assumption (1) implies that} \\ = \; & \langle f - p_n, f - r \rangle \\ \leq \; & \|f - p_n\|_2 \|f - r\|_2 \text{ by the Cauchy–Schwarz inequality.} \end{aligned}$$

Dividing both sides by $\|f - p_n\|_2$ gives the required result. □

**Remark:** the converse is true too (see problem sheet 3).

This gives a direct way to calculate a best approximation: we want to find $p_n(x) = \sum_{k=0}^{n} \alpha_k x^k$ such that

$$\int_a^b w(x) \left( f - \sum_{k=0}^{n} \alpha_k x^k \right) x^i \, \mathrm{d}x = 0 \text{ for } i = 0, 1, \ldots, n. \tag{2}$$

[Note that (2) holds if, and only if,

$$\int_a^b w(x) \left( f - \sum_{k=0}^{n} \alpha_k x^k \right) \left( \sum_{i=0}^{n} \beta_i x^i \right) \mathrm{d}x = 0 \qquad \forall q = \sum_{i=0}^{n} \beta_i x^i \in \Pi_n.]$$

However, (2) implies that

$$\sum_{k=0}^{n} \left( \int_a^b w(x) x^{k+i} \, \mathrm{d}x \right) \alpha_k = \int_a^b w(x) f(x) x^i \, \mathrm{d}x \text{ for } i = 0, 1, \ldots, n$$

which is the component-wise statement of a matrix equation

$$A\alpha = \varphi, \tag{3}$$

to determine the coefficients $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_n)^{\mathrm{T}}$, where $A = \{a_{i,k}, i, k = 0, 1, \ldots, n\}$, $\varphi = (f_0, f_1, \ldots, f_n)^{\mathrm{T}}$,

$$a_{i,k} = \int_a^b w(x) x^{k+i} \, \mathrm{d}x \text{ and } f_i = \int_a^b w(x) f(x) x^i \, \mathrm{d}x.$$

The system (3) are called the **normal equations**.

**Example:** the best least-squares approximation to $\mathrm{e}^x$ on $[0,1]$ from $\Pi_1$ in $\langle f,g\rangle = \int_a^b f(x)g(x)\,\mathrm{d}x$. We want

$$\int_0^1 [\mathrm{e}^x - (\alpha_0 1 + \alpha_1 x)]1\,\mathrm{d}x = 0 \;\text{ and }\; \int_0^1 [\mathrm{e}^x - (\alpha_0 1 + \alpha_1 x)]x\,\mathrm{d}x = 0.$$

$\iff$

$$\begin{aligned} \alpha_0 \int_0^1 \mathrm{d}x + \alpha_1 \int_0^1 x\,\mathrm{d}x &= \int_0^1 \mathrm{e}^x\,\mathrm{d}x \\ \alpha_0 \int_0^1 x\,\mathrm{d}x + \alpha_1 \int_0^1 x^2\,\mathrm{d}x &= \int_0^1 \mathrm{e}^x x\,\mathrm{d}x \end{aligned}$$

i.e.,

$$\begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \begin{bmatrix} \mathrm{e}-1 \\ 1 \end{bmatrix}$$

$\implies \alpha_0 = 4\mathrm{e} - 10$ and $\alpha_1 = 18 - 6\mathrm{e}$, so $p_1(x) := (18-6\mathrm{e})x + (4\mathrm{e}-10)$ is the best approximation.

Proof that the coefficient matrix $A$ is nonsingular will now establish existence and uniqueness of (weighted) $\|\cdot\|_2$ best-approximation.

**Theorem.** The coefficient matrix $A$ is nonsingular.

**Proof.** Suppose not $\implies \exists \alpha \neq 0$ with $A\alpha = 0 \implies \alpha^{\mathrm{T}} A \alpha = 0$

$$\iff \sum_{i=0}^n \alpha_i (A\alpha)_i = 0 \iff \sum_{i=0}^n \alpha_i \sum_{k=0}^n a_{ik}\alpha_k = 0,$$

and using the definition $a_{ik} = \int_a^b w(x)x^k x^i\,\mathrm{d}x$ ,

$$\iff \sum_{i=0}^n \alpha_i \sum_{k=0}^n \left( \int_a^b w(x) x^k x^i\,\mathrm{d}x \right) \alpha_k = 0.$$

Rearranging gives

$$\int_a^b w(x) \left( \sum_{i=0}^n \alpha_i x^i \right) \left( \sum_{k=0}^n \alpha_k x^k \right) \mathrm{d}x = 0 \;\text{ or }\; \int_a^b w(x) \left( \sum_{i=0}^n \alpha_i x^i \right)^2 \mathrm{d}x = 0$$

which implies that $\sum_{i=0}^n \alpha_i x^i = 0$ and thus $\alpha_i = 0$ for $i = 0, 1, \ldots, n$. This contradicts the initial supposition, and thus $A$ is nonsingular. $\square$

**Remark:**

- Note in the simplest least-squares approximation problem $\min_x \|Ax - b\|_2$ that we dealt with in lecture 4, the theorem gives the solution $A^T(Ax - b) = 0$, that is, $x = (A^T A)^{-1} A^T b$. This coincides with the QR-based solution derived in lecture 4.
- The above theorem does not imply that the normal equations are usable in practice: the method would need to be stable with respect to small perturbations. In fact, difficulties arise from the "ill-conditioning" of the matrix $A$ as $n$ increases. The next lecture looks at a fix.

**Numerical Analysis Hilary Term 2021**
**Lecture 10: Orthogonal Polynomials**

**Gram–Schmidt orthogonalization procedure:** the solution of the normal equations $A\alpha = \varphi$ for best least-squares polynomial approximation would be easy if $A$ were diagonal. Instead of $\{1, x, x^2, \ldots, x^n\}$ as a basis for $\Pi_n$, suppose we have a basis $\{\phi_0, \phi_1, \ldots, \phi_n\}$. Then $p_n(x) = \sum_{k=0}^{n} \beta_k \phi_k(x)$, and the normal equations become

$$\int_a^b w(x) \left( f(x) - \sum_{k=0}^{n} \beta_k \phi_k(x) \right) \phi_i(x)\, \mathrm{d}x = 0 \quad \text{for} \quad i = 0, 1, \ldots, n,$$

or equivalently

$$\sum_{k=0}^{n} \left( \int_a^b w(x)\phi_k(x)\phi_i(x)\, \mathrm{d}x \right) \beta_k = \int_a^b w(x) f(x) \phi_i(x)\, \mathrm{d}x, \quad i = 0, \ldots, n, \quad \text{i.e.,}$$

$$A\beta = \varphi, \tag{1}$$

where $\beta = (\beta_0, \beta_1, \ldots, \beta_n)^{\mathrm{T}}$, $\varphi = (f_1, f_2, \ldots, f_n)^{\mathrm{T}}$ and now

$$a_{i,k} = \int_a^b w(x)\phi_k(x)\phi_i(x)\, \mathrm{d}x \quad \text{and} \quad f_i = \int_a^b w(x) f(x) \phi_i(x)\, \mathrm{d}x.$$

So $A$ is diagonal if

$$\langle \phi_i, \phi_k \rangle = \int_a^b w(x)\phi_i(x)\phi_k(x)\, \mathrm{d}x \quad \begin{cases} = 0 & i \neq k \quad \text{and} \\ \neq 0 & i = k. \end{cases}$$

We can create such a set of **orthogonal polynomials**

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

with $\phi_i \in \Pi_i$ for each $i$, by the Gram–Schmidt procedure, which is based on the following lemma.

**Lemma.** Suppose that $\phi_0, \ldots, \phi_k$, with $\phi_i \in \Pi_i$ for each $i$, are orthogonal with respect to the inner product $\langle f, g \rangle = \int_a^b w(x) f(x) g(x)\, \mathrm{d}x$. Then,

$$\phi_{k+1}(x) = x^{k+1} - \sum_{i=0}^{k} \lambda_i \phi_i(x)$$

satisfies

$$\langle \phi_{k+1}, \phi_j \rangle = \int_a^b w(x)\phi_{k+1}(x)\phi_j(x)\, \mathrm{d}x = 0, \quad j = 0, 1, \ldots, k, \quad \text{with}$$

$$\lambda_j = \frac{\langle x^{k+1}, \phi_j \rangle}{\langle \phi_j, \phi_j \rangle}, \quad j = 0, 1, \ldots, k.$$

**Proof.** For any $j$, $0 \le j \le k$,

$$\begin{aligned}\langle \phi_{k+1}, \phi_j \rangle &= \langle x^{k+1}, \phi_j \rangle - \sum_{i=0}^{k} \lambda_i \langle \phi_i, \phi_j \rangle \\ &= \langle x^{k+1}, \phi_j \rangle - \lambda_j \langle \phi_j, \phi_j \rangle \\ &\qquad \text{by the orthogonality of } \phi_i \text{ and } \phi_j,\ i \neq j, \\ &= 0 \qquad \text{by definition of } \lambda_j. \end{aligned}$$

□

**Notes:** 1. The G–S procedure does this successively for $k = 0, 1, \ldots, n$.
2. $\phi_k$ is always of exact degree $k$, so $\{\phi_0, \ldots, \phi_\ell\}$ is a basis for $\Pi_\ell$ $\forall \ell \ge 0$.
3. $\phi_k$ can be normalised to satisfy $\langle \phi_k, \phi_k \rangle = 1$ or to be monic, or ...

**Examples:** 1. The inner product $\langle f, g \rangle = \displaystyle\int_{-1}^{1} f(x)g(x)\,\mathrm{d}x$

gives orthogonal polynomials called the **Legendre polynomials**,

$$\phi_0(x) \equiv 1, \quad \phi_1(x) = x, \quad \phi_2(x) = x^2 - \tfrac{1}{3}, \quad \phi_3(x) = x^3 - \tfrac{3}{5}x, \ldots$$

2. The inner product $\langle f, g \rangle = \displaystyle\int_{-1}^{1} \frac{f(x)g(x)}{\sqrt{1-x^2}}\,\mathrm{d}x$

gives orthogonal polynomials called the **Chebyshev polynomials**,

$$\phi_0(x) \equiv 1, \quad \phi_1(x) = x, \quad \phi_2(x) = 2x^2 - 1, \quad \phi_3(x) = 4x^3 - 3x, \ldots$$

3. The inner product $\langle f, g \rangle = \displaystyle\int_{0}^{\infty} \mathrm{e}^{-x} f(x)g(x)\,\mathrm{d}x$

gives orthogonal polynomials called the **Laguerre polynomials**,

$$\phi_0(x) \equiv 1, \quad \phi_1(x) = 1 - x, \quad \phi_2(x) = 2 - 4x + x^2,$$
$$\phi_3(x) = 6 - 18x + 9x^2 - x^3, \ldots$$

**Lemma.** Suppose that $\{\phi_0, \phi_1, \ldots, \phi_k, \ldots\}$ are orthogonal polynomials for a given inner product $\langle \cdot, \cdot \rangle$. Then, $\langle \phi_k, q \rangle = 0$ whenever $q \in \Pi_{k-1}$.

**Proof.** This follows since if $q \in \Pi_{k-1}$, then $q(x) = \displaystyle\sum_{i=0}^{k-1} \sigma_i \phi_i(x)$ for some $\sigma_i \in \mathbb{R}$, $i = 0, 1, \ldots, k-1$, so

$$\langle \phi_k, q \rangle = \sum_{i=0}^{k-1} \sigma_i \langle \phi_k, \phi_i \rangle = 0.$$

□

**Remark:** note from the above argument that if $q(x) = \displaystyle\sum_{i=0}^{k} \sigma_i \phi_i(x)$ is of exact degree $k$ (so $\sigma_k \neq 0$), then $\langle \phi_k, q \rangle = \sigma_k \langle \phi_k, \phi_k \rangle \neq 0$.

**Theorem.** Suppose that $\{\phi_0, \phi_1, \ldots, \phi_n, \ldots\}$ is a set of orthogonal polynomials. Then, there exist sequences of real numbers $(\alpha_k)_{k=1}^{\infty}$, $(\beta_k)_{k=1}^{\infty}$, $(\gamma_k)_{k=1}^{\infty}$ such that a three-term recurrence relation holds of the form

$$\phi_{k+1}(x) = \alpha_k (x - \beta_k)\phi_k(x) - \gamma_k \phi_{k-1}(x), \qquad k = 1, 2, \ldots.$$

**Proof.** The polynomial $x\phi_k \in \Pi_{k+1}$, so there exist real numbers

$$\sigma_{k,0}, \sigma_{k,1}, \ldots, \sigma_{k,k+1}$$

such that

$$x\phi_k(x) = \sum_{i=0}^{k+1} \sigma_{k,i}\phi_i(x)$$

as $\{\phi_0, \phi_1, \ldots, \phi_{k+1}\}$ is a basis for $\Pi_{k+1}$. Now take the inner product on both sides with $\phi_j$ where $j \leq k-2$. On the left-hand side, note $x\phi_j \in \Pi_{k-1}$ and thus

$$\langle x\phi_k, \phi_j\rangle = \int_a^b w(x)x\phi_k(x)\phi_j(x)\,\mathrm{d}x = \int_a^b w(x)\phi_k(x)x\phi_j(x)\,\mathrm{d}x = \langle \phi_k, x\phi_j\rangle = 0,$$

by the above lemma for $j \leq k-2$. On the right-hand side

$$\left\langle \sum_{i=0}^{k+1} \sigma_{k,i}\phi_i, \phi_j \right\rangle = \sum_{i=0}^{k+1} \sigma_{k,i}\langle \phi_i, \phi_j\rangle = \sigma_{k,j}\langle \phi_j, \phi_j\rangle$$

by the linearity of $\langle\cdot,\cdot\rangle$ and orthogonality of $\phi_i$ and $\phi_j$ for $i \neq j$. Hence $\sigma_{k,j} = 0$ for $j \leq k-2$, and so

$$x\phi_k(x) = \sigma_{k,k+1}\phi_{k+1}(x) + \sigma_{k,k}\phi_k(x) + \sigma_{k,k-1}\phi_{k-1}(x).$$

Almost there: taking the inner product with $\phi_{k+1}$ reveals that

$$\langle x\phi_k, \phi_{k+1}\rangle = \sigma_{k,k+1}\langle \phi_{k+1}, \phi_{k+1}\rangle,$$

so $\sigma_{k,k+1} \neq 0$ by the above remark as $x\phi_k$ is of exact degree $k+1$ (e.g., from above Gram–Schmidt notes). Thus,

$$\phi_{k+1}(x) = \frac{1}{\sigma_{k,k+1}}(x - \sigma_{k,k})\phi_k(x) - \frac{\sigma_{k,k-1}}{\sigma_{k,k+1}}\phi_{k-1}(x),$$

which is of the given form, with

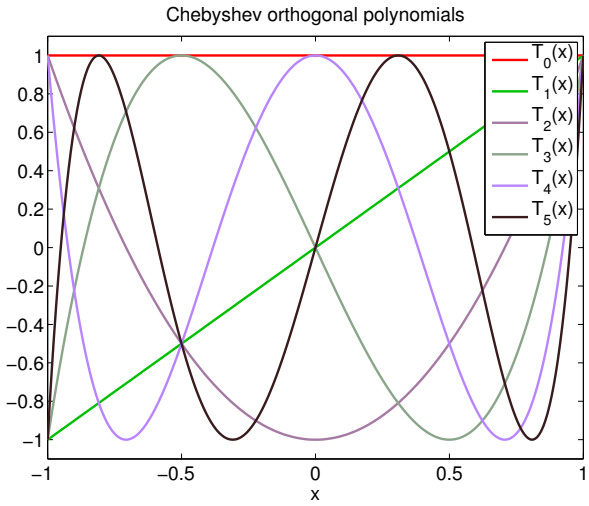
$$\alpha_k = \frac{1}{\sigma_{k,k+1}}, \qquad \beta_k = \sigma_{k,k}, \qquad \gamma_k = \frac{\sigma_{k,k-1}}{\sigma_{k,k+1}}, \qquad k = 1, 2, \ldots.$$

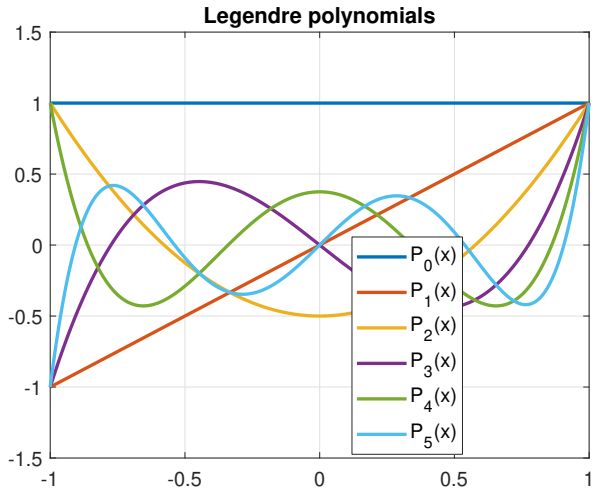
That completes the proof. □

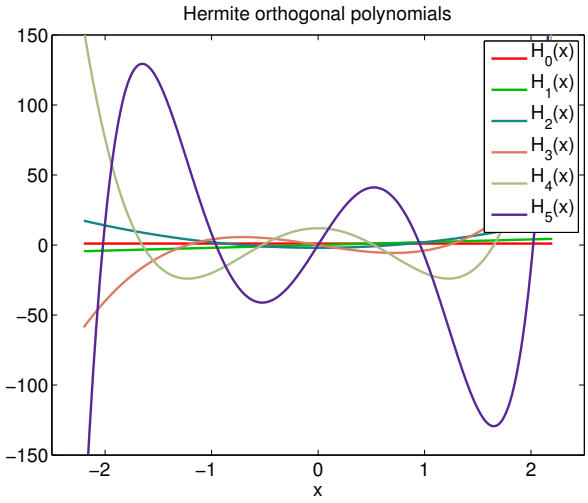
**Example.** The inner product $\langle f, g\rangle = \displaystyle\int_{-\infty}^{\infty} \mathrm{e}^{-x^2} f(x)g(x)\,\mathrm{d}x$

gives orthogonal polynomials called the **Hermite polynomials**,

$$\phi_0(x) \equiv 1, \quad \phi_1(x) = 2x, \quad \phi_{k+1}(x) = 2x\phi_k(x) - 2k\phi_{k-1}(x) \ \text{ for } \ k \geq 1.$$


Chebyshev orthogonal polynomials
T0(x)
T1(x)
T2(x)
T3(x)
T4(x)
T5(x)
x


Legendre polynomials
P0(x)
P1(x)
P2(x)
P3(x)
P4(x)
P5(x)


Hermite orthogonal polynomials
H0(x)
H1(x)
H2(x)
H3(x)
H4(x)
H5(x)
x

**Numerical Analysis Hilary Term 2021**
**Lecture 11: Gauss quadrature**

**Terminology:** Quadrature ≡ numerical integration

**Goal:** given a (continuous) function $f : [a, b] \to \mathbb{R}$, find its integral $I = \int_a^b f(x)dx$, as accurately as possible.

**Idea: Approximate and Integrate**. Find a polynomial $p_n$ from data $\{(x_k, f(x_k))\}_{k=0}^n$ by Lagrange interpolation (lecture 1), and integrate $\int_{x_0}^{x_n} p_n(x)\,\mathrm{d}x =: I_n$. Ideally, $I_n = I$ or at least $I_n \approx I$. Is this true?

If we choose $x_k$ to be equispaced points in $[a, b]$, the resulting $I_n$ is known as the Newton-Cotes quadrature. This method is actually quite unstable and inaccurate, and a much more accurate and elegant quadrature rule exists: Gauss quadrature. In this lecture we cover this beautiful result involving orthogonal polynomials.

**Preparations:** Suppose that $w$ is a weight function, defined, positive and integrable on the open interval $(a, b)$ of $\mathbb{R}$.

**Lemma.** Let $\{\phi_0, \phi_1, \ldots, \phi_n, \ldots\}$ be orthogonal polynomials for the inner product $\langle f, g\rangle = \int_a^b w(x)f(x)g(x)\,\mathrm{d}x$. Then, for each $k = 0, 1, \ldots$, $\phi_k$ has $k$ distinct roots in the interval $(a, b)$.

**Proof.** Since $\phi_0(x) \equiv$ const. $\neq 0$, the result is trivially true for $k = 0$. Suppose that $k \geq 1$: $\langle \phi_k, \phi_0\rangle = \int_a^b w(x)\phi_k(x)\phi_0(x)\,\mathrm{d}x = 0$ with $\phi_0$ constant implies that $\int_a^b w(x)\phi_k(x)\,\mathrm{d}x = 0$ with $w(x) > 0$, $x \in (a, b)$. Thus $\phi_k(x)$ must change sign in $(a, b)$, i.e., $\phi_k$ has at least one root in $(a, b)$.

Suppose that there are $\ell$ points $a < r_1 < r_2 < \cdots < r_\ell < b$ where $\phi_k$ changes sign for some $1 \leq \ell \leq k$. Then

$$q(x) = \prod_{j=1}^{\ell}(x - r_j) \times \text{ the sign of } \phi_k \text{ on } (r_\ell, b)$$

has the same sign as $\phi_k$ on $(a, b)$. Hence

$$\langle \phi_k, q\rangle = \int_a^b w(x)\phi_k(x)q(x)\,\mathrm{d}x > 0,$$

and thus it follows from the previous lemma (cf. Lecture 12) that $q$, (which is of degree $\ell$) must be of degree $\geq k$, i.e., $\ell \geq k$. However, $\phi_k$ is of exact degree $k$, and therefore the number of its distinct roots, $\ell$, must be $\leq k$. Hence $\ell = k$, and $\phi_k$ has $k$ distinct roots in $(a, b)$. □

**Application to quadrature.** The above lemma leads to very efficient quadrature rules since it answers the question: how should we choose the quadrature points $x_0, x_1, \ldots, x_n$ in the quadrature rule

$$\int_a^b w(x)f(x)\,\mathrm{d}x \approx \sum_{j=0}^{n} w_j f(x_j) \tag{1}$$

so that the rule is exact for polynomials of degree as high as possible? (The case $w(x) \equiv 1$ is the most common.)

**Recall:** the Lagrange interpolating polynomial

$$p_n = \sum_{j=0}^{n} f(x_j) L_{n,j} \in \Pi_n$$

is unique, so $f \in \Pi_n \implies p_n \equiv f$ whatever interpolation points are used, and moreover

$$\int_a^b w(x) f(x)\,\mathrm{d}x = \int_a^b w(x) p_n(x)\,\mathrm{d}x = \sum_{j=0}^{n} w_j f(x_j),$$

exactly, where

$$w_j = \int_a^b w(x) L_{n,j}(x)\,\mathrm{d}x. \tag{2}$$

**Theorem.** Suppose that $x_0 < x_1 < \cdots < x_n$ are the roots of the $n+1$-st degree orthogonal polynomial $\phi_{n+1}$ with respect to the inner product

$$\langle g, h \rangle = \int_a^b w(x) g(x) h(x)\,\mathrm{d}x.$$

Then, the quadrature formula (1) with weights (2) is exact whenever $f \in \Pi_{2n+1}$.

**Proof.** Let $p \in \Pi_{2n+1}$. Then by the Division Algorithm $p(x) = q(x)\phi_{n+1}(x) + r(x)$ with $q, r \in \Pi_n$. So

$$\int_a^b w(x) p(x)\,\mathrm{d}x = \int_a^b w(x) q(x) \phi_{n+1}(x)\,\mathrm{d}x + \int_a^b w(x) r(x)\,\mathrm{d}x = \sum_{j=0}^{n} w_j r(x_j) \tag{3}$$

since the integral involving $q \in \Pi_n$ is zero by the lemma above and the other is integrated exactly since $r \in \Pi_n$. Finally $p(x_j) = q(x_j)\phi_{n+1}(x_j) + r(x_j) = r(x_j)$ for $j = 0, 1, \ldots, n$ as the $x_j$ are the roots of $\phi_{n+1}$. So (3) gives

$$\int_a^b w(x) p(x)\,\mathrm{d}x = \sum_{j=0}^{n} w_j p(x_j),$$

where $w_j$ is given by (2) whenever $p \in \Pi_{2n+1}$. □

These quadrature rules are called **Gauss quadratures**.

- $w(x) \equiv 1$, $(a, b) = (-1, 1)$: Gauss–Legendre quadrature.
- $w(x) = (1 - x^2)^{-1/2}$ and $(a, b) = (-1, 1)$: Gauss–Chebyshev quadrature.
- $w(x) = \mathrm{e}^{-x}$ and $(a, b) = (0, \infty)$: Gauss–Laguerre quadrature.
- $w(x) = \mathrm{e}^{-x^2}$ and $(a, b) = (-\infty, \infty)$: Gauss–Hermite quadrature.

They give better accuracy than Newton–Cotes quadrature for the same number of function evaluations.
Note when using quadrature on unbounded intervals, the integral should be of the form $\int_0^\infty \mathrm{e}^{-x} f(x)\,\mathrm{d}x$ and only $f$ is sampled at the nodes.
Note that by the linear change of variable $t = (2x - a - b)/(b - a)$, which maps $[a, b] \to [-1, 1]$, we can evaluate for example

$$\int_a^b f(x)\,\mathrm{d}x = \int_{-1}^1 f\left(\frac{(b-a)t + b + a}{2}\right)\frac{b-a}{2}\,\mathrm{d}t \simeq \frac{b-a}{2}\sum_{j=0}^n w_j f\left(\frac{b-a}{2}t_j + \frac{b+a}{2}\right),$$

where $\simeq$ denotes "quadrature" and the $t_j$, $j = 0, 1, \ldots, n$, are the roots of the $n+1$-st degree Legendre polynomial.

**Example.** 2-point Gauss–Legendre quadrature: $\phi_2(t) = t^2 - \frac{1}{3} \implies t_0 = -\frac{1}{\sqrt{3}}$, $t_1 = \frac{1}{\sqrt{3}}$ and

$$w_0 = \int_{-1}^1 \frac{t - \frac{1}{\sqrt{3}}}{-\frac{1}{\sqrt{3}} - \frac{1}{\sqrt{3}}}\,\mathrm{d}t = -\int_{-1}^1 \left(\tfrac{\sqrt{3}}{2}t - \tfrac{1}{2}\right)\,\mathrm{d}t = 1,$$

with $w_1 = 1$, similarly. So e.g., changing variables $x = (t+3)/2$,

$$\int_1^2 \frac{1}{x}\,\mathrm{d}x = \frac{1}{2}\int_{-1}^1 \frac{2}{t+3}\,\mathrm{d}t \simeq \frac{1}{3 + \frac{1}{\sqrt{3}}} + \frac{1}{3 - \frac{1}{\sqrt{3}}} = 0.6923077\ldots\ .$$

Note that the trapezium rule (also two evaluations of the integrand) gives

$$\int_1^2 \frac{1}{x}\,\mathrm{d}x \simeq \frac{1}{2}\left[\frac{1}{2} + 1\right] = 0.75,$$

whereas $\displaystyle\int_1^2 \frac{1}{x}\,\mathrm{d}x = \ln 2 = 0.6931472\ldots$ .

**Theorem.** Error in Gauss quadrature: suppose that $f^{(2n+2)}$ is continuous on $(a, b)$. Then

$$\int_a^b w(x) f(x)\,\mathrm{d}x = \sum_{j=0}^n w_j f(x_j) + \frac{f^{(2n+2)}(\eta)}{(2n+2)!}\int_a^b w(x)\prod_{j=0}^n (x - x_j)^2\,\mathrm{d}x,$$

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

**Proof.** The proof is based on the Hermite interpolating polynomial $H_{2n+1}$ to $f$ on $x_0, x_1, \ldots, x_n$. [Recall that $H_{2n+1}(x_j) = f(x_j)$ and $H'_{2n+1}(x_j) = f'(x_j)$ for $j = 0, 1, \ldots, n$.] The error in Hermite interpolation is

$$f(x) - H_{2n+1}(x) = \frac{1}{(2n+2)!} f^{(2n+2)}(\eta(x))\prod_{j=0}^n (x - x_j)^2$$

for some $\eta = \eta(x) \in (a, b)$. Now $H_{2n+1} \in \Pi_{2n+1}$, so

$$\int_a^b w(x) H_{2n+1}(x)\,\mathrm{d}x = \sum_{j=0}^n w_j H_{2n+1}(x_j) = \sum_{j=0}^n w_j f(x_j),$$

the first identity because Gauss quadrature is exact for polynomials of this degree and the second by interpolation. Thus

$$\int_a^b w(x)f(x)\,\mathrm{d}x - \sum_{j=0}^{n} w_j f(x_j) = \int_a^b w(x)[f(x) - H_{2n+1}(x)]\,\mathrm{d}x$$
$$= \frac{1}{(2n+2)!}\int_a^b f^{(2n+2)}(\eta(x))w(x)\prod_{j=0}^{n}(x-x_j)^2\,\mathrm{d}x,$$

and hence the required result follows from the integral mean value theorem as $w(x)\prod_{j=0}^{n}(x-x_j)^2 \geq 0$. □

**Remark:** the "direct" approach of finding Gauss quadrature formulae sometimes works for small $n$, but more sophisticated algorithms are used for large $n$.[1]

**Example.** To find the two-point Gauss–Legendre rule $w_0 f(x_0) + w_1 f(x_1)$ on $(-1, 1)$ with weight function $w(x) \equiv 1$, we need to be able to integrate any cubic polynomial exactly, so

$$2 = \int_{-1}^{1} 1\,\mathrm{d}x = w_0 + w_1 \tag{4}$$
$$0 = \int_{-1}^{1} x\,\mathrm{d}x = w_0 x_0 + w_1 x_1 \tag{5}$$
$$\tfrac{2}{3} = \int_{-1}^{1} x^2\,\mathrm{d}x = w_0 x_0^2 + w_1 x_1^2 \tag{6}$$
$$0 = \int_{-1}^{1} x^3\,\mathrm{d}x = w_0 x_0^3 + w_1 x_1^3. \tag{7}$$

These are four nonlinear equations in four unknowns $w_0$, $w_1$, $x_0$ and $x_1$. Equations (5) and (7) give

$$\begin{bmatrix} x_0 & x_1 \\ x_0^3 & x_1^3 \end{bmatrix}\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

which implies that

$$x_0 x_1^3 - x_1 x_0^3 = 0$$

for $w_0$, $w_1 \neq 0$, i.e.,

$$x_0 x_1 (x_1 - x_0)(x_1 + x_0) = 0.$$

If $x_0 = 0$, this implies $w_1 = 0$ or $x_1 = 0$ by (5), either of which contradicts (6). Thus $x_0 \neq 0$, and similarly $x_1 \neq 0$. If $x_1 = x_0$, (5) implies $w_1 = -w_0$, which contradicts (4). So $x_1 = -x_0$, and hence (5) implies $w_1 = w_0$. But then (4) implies that $w_0 = w_1 = 1$ and (6) gives

$$x_0 = -\tfrac{1}{\sqrt{3}} \qquad \text{and} \qquad x_1 = \tfrac{1}{\sqrt{3}},$$

[1] See e.g., the research paper by Hale and Townsend, "Fast and accurate computation of Guass–Legendre and Gauss–Jacobi quadrature nodes and weights" SIAM J. Sci. Comput. 2013.

which are the roots of the Legendre polynomial $x^2 - \frac{1}{3}$.

**Convergence:** Gauss quadrature converges astonishingly fast. It can be shown that if $f$ is analytic on $[a, b]$, the convergence is geometric (exponential) in the number of samples. This is in contrast to other (more straightforward) quadrature rules:

- Newton-Cotes: Find interpolant in $n$ equispaced points, and integrate interpolant. Convergence: (often) Divergent!
- (Composite) trapezium rule: Find piecewise-linear interpolant in $n$ equispaced points, and integrate interpolant. Convergence: $O(1/n^2)$ (assumes $f''$ exists)
- (Composite) Simpson's rule: Find piecewise-quadratic interpolant in $n$ equispaced points (each subinterval containing three points), and integrate interpolant. Convergence: $O(1/n^4)$ (assumes $f''''$ exists)

The figure below illustrates the performance on integrating the Runge function.

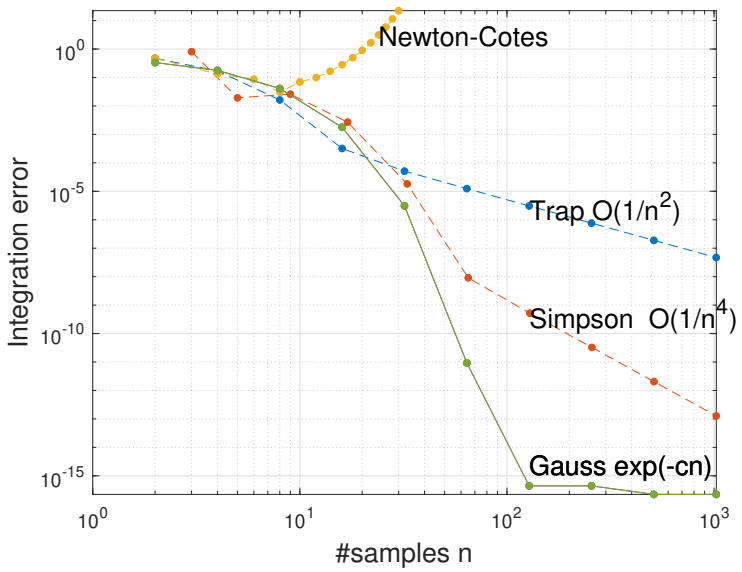
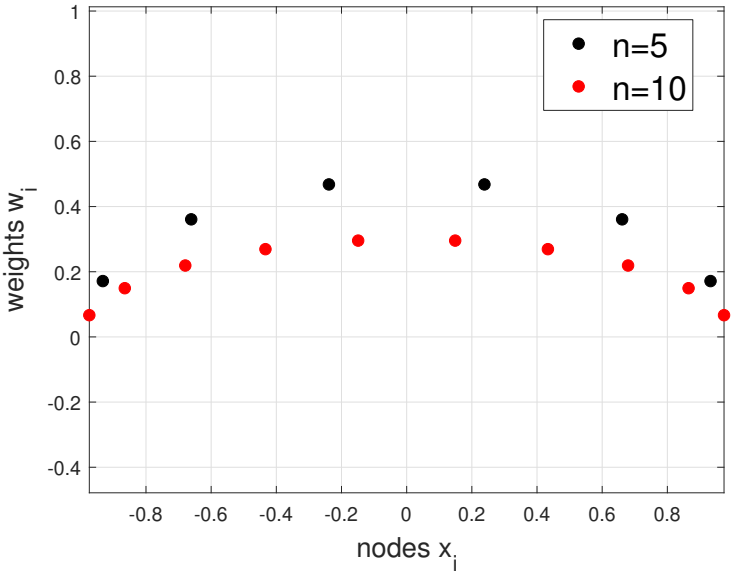



Figure 1: Convergence of quadrature rules for $\int_{-1}^{1} \frac{1}{25x^2+1} dx$ (Runge function)

**Nodes and weights for Gauss(-Legendre) quadrature** The figure below shows the nodes (interpolation points) and the corrsponding weights with the standard Gauss-Legendre quadrature rule, i.e., when $w(x) = 1$ and $[a, b] = [-1, 1]$. In Chebfun these are computed conveniently by `[x,w] = legpts(n+1)`




Note that the nodes/interpolation points cluster near endpoints (and sparser in the middle); this is a general phenomenon, and very analogous to the Chebyshev interpolation points mentioned in the least-squares lecture (Gauss and Chebyshev points have asymptotically the same distribution of points). Note also that the weights are all positive and shrink as $n$ grows; they have to because they sum to 2 (why?).

**Numerical Analysis Hilary Term 2021**
**Lecture 12–13: Introduction to numerical methods for initial-value problems**

**Initial value problems:** Initial value problems arise everywhere in mathematics where we wish to model the evolution in time of a given system.

**Definition 1.** *Let $I = [x_0, X] \subset \mathbb{R}$ be a (time) interval and $D \subset \mathbb{R}^d$ be an open subset, where $d \in \mathbb{N}^+$ denotes the space dimension.*

- *A* first-order ordinary differential equation *(ODE) is an equation of the form*

$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}) \,,$$

  *where the righthand side is a function $\mathbf{f} : I \times D \to \mathbb{R}^d$.*

- *An* initial value problem *(IVP) is an ODE with an initial condition, that is,*

$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}) \,, \qquad \mathbf{y}(x_0) = \mathbf{y}_0 \,.$$

Picard's Theorem gives sufficient conditions to ensure that the IVP admits a unique solution[1].

**Theorem.** Suppose that $\mathbf{f}$ is continuous in a neighborhood $U \subset \mathbb{R}^{1+d}$ of $(x_0, \mathbf{y}_0)$ that contains the (closed) cylinder

$$R = \{(x, \mathbf{y}) : x_0 \le x \le X_M, \|\mathbf{y} - \mathbf{y}_0\| \le Y_M\} \,,$$

where $X_M > x_0$ and $Y_M > 0$ are constants. Suppose also that there exists a positive constant $L$ such that

$$\|\mathbf{f}(x, \mathbf{y}) - \mathbf{f}(x, \mathbf{z})\| \le L\|\mathbf{y} - \mathbf{z}\|$$

holds whenever $(x, \mathbf{y})$ and $(x, \mathbf{z})$ lie in $R$. Finally, letting

$$M := \max\{\|\mathbf{f}(x, \mathbf{y})\| : (x, \mathbf{y}) \in R\} \,,$$

suppose that $M(X_M - x_0) \le Y_M$. Then, there exists a unique continuously differentiable function

$$[x_0, X_M] \ni x \mapsto \mathbf{y}(x) \in \mathbb{R}^d$$

that is the solution to our IVP.

Note that Picard's theorem guarantees the existence of a solution only up to a finite time $X_M$; consider : $y' = y^2$, $y(0) = 1$, which has solution $y(x) = (1-x)^{-1}$ and blows up at $x = 1$. Furthermore, if an IVP satisfies the assumptions of Picard's theorem, its solution is stable on the bounded interval $[x_0, X]$. This means that if $\mathbf{y} : [x_0, X] \to D$ solves the IVP

$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}) \,, \qquad \mathbf{y}(x_0) = \mathbf{y}_0 \,,$$

[1]For more details about Picard's theorem, we refer to chapter 11 of Prof. Trefethen's book *Exploring ODEs*, which is freely available at http://people.maths.ox.ac.uk/trefethen/Exploring.pdf

and $\tilde{\mathbf{y}} : [x_0, X] \to D$ solves the same ODE with a perturbed initial condition $\tilde{\mathbf{y}}_0$, that is,

$$\tilde{\mathbf{y}}'(x) = \mathbf{f}(x, \tilde{\mathbf{y}}) \,, \qquad \tilde{\mathbf{y}}(x_0) = \tilde{\mathbf{y}}_0 \,,$$

then

$$\|\mathbf{y}(x) - \tilde{\mathbf{y}}(x)\| \leq e^{L(X-x_0)} \|\mathbf{y}_0 - \tilde{\mathbf{y}}_0\| \quad \forall\, x \in [x_0, X] \,.$$

This implies that a small error in the initial condition does not compromise dramatically the solution of the IVP. However, note that the constant $e^{L(X-x_0)}$ in the above bound grows exponentially as the final time $X$ increases.

You have seen in A1 Differential Equations 1 that any higher-order IVP can be reformulated as a larger first-order IVP. We therefore mainly consider numerical methods for first-order problems. It is also possible to reformulate nonautonomous problems as larger autonomous ones (wherein $\mathbf{y}' = \mathbf{f}(\mathbf{y})$), so sometimes we will restrict ourselves to autonomous ones when it is convenient to do so.

An inconvenient truth is that most IVPs (and most differential equations) cannot be solve analytically (i.e., exactly, to obtain closed-form solutions). It therefore becomes necessary to find approximate solutions with a numerical algorithm. Fortunately, a number of reliable and efficient methods are available for such solution. The remainder of this course is devoted to these methods and their analysis.

**One-step methods** Assume that the IVP

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}) \,, \qquad \mathbf{y}(x_0) = \mathbf{y}_0 \,,$$

admits a unique stable solution $\mathbf{y} : [x_0, X] \to D$ that is defined on the bounded interval $[x_0, X]$. How can we compute a numerical approximation of $\mathbf{y}$ that can be made arbitrarily accurate? A simple idea is to first divide the interval $[x_0, X]$ into $N \in \mathbb{N}^+$ subintervals defined by the equidistant points $x_n = x_0 + nh$, $n = 0, \ldots, N$, where the *step size* $h$ is $h = (X - x_0)/N$. To each time step $x_n$, we associate an approximation $\mathbf{y}_n$ of $\mathbf{y}(x_n)$. To define how to compute these approximations, we take inspiration from the integral equation arising in Picard's theorem:

$$\mathbf{y}(x_{n+1}) = \mathbf{y}(x_n) + \int_{x_n}^{x_{n+1}} \mathbf{f}(x, \mathbf{y}(x)) \, \mathrm{d}x \,,$$

which is obtained by integrating the IVP, and where the integration is to be understood componentwise.

This equality suggests that, if we have already computed an approximation $\mathbf{y}_n$ of $\mathbf{y}(x_n)$, we could compute $\mathbf{y}_{n+1}$ by adding to $\mathbf{y}_n$ an approximation of the integral appearing on the righthand side. There is therefore a deep connection between quadrature and the solution of IVPs. Indeed if $\mathbf{f}(x, \mathbf{y}) = \mathbf{f}(x)$, i.e., $\mathbf{f}$ does not depend on $\mathbf{y}$, then computing $\mathbf{y}$ is a standard quadrature problem, and can be solved by e.g. Gauss quadrature. Starting with $n = 0$, we could iterate such a strategy to compute the entire sequence $\{\mathbf{y}_n\}_{n=0}^N$. In what follows, we construct three different schemes based on three different (and still very similar) approximations of the integral and investigate the impact that this choice has on the properties of the resulting numerical method.

To construct an approximation of the integral, we recall that by the mean value theorem there is a $\xi \in [x_n, x_{n+1}]$ such that

$$\int_{x_n}^{x_{n+1}} \mathbf{f}(x, \mathbf{y}(x))\,\mathrm{d}x = h\mathbf{f}(\xi, \mathbf{y}(\xi))\,.$$

Therefore, we can construct an approximation by replacing $\xi$ with a value $s$ we like. The resulting numerical approximation rule is called a *rectangle rule*. A consequence is that, for any $s \in [x_n, x_{n+1}]$, the approximation error is at most

$$\int_{x_n}^{x_{n+1}} \mathbf{f}(x, \mathbf{y}(x))\,\mathrm{d}x - h\mathbf{f}(s, \mathbf{y}(s)) = h(\mathbf{f}(\xi, \mathbf{y}(\xi)) - \mathbf{f}(s, \mathbf{y}(s))) \leq h \max_{r\in[x_n,x_{n+1}]} |\mathbf{f}(r, \mathbf{y}(r)) - \mathbf{f}(s, \mathbf{y}(s))|.$$

For instance, we can choose $s = x_n$, so that

$$\int_{x_n}^{x_{n+1}} \mathbf{f}(x, \mathbf{y}(x))\,\mathrm{d}x \approx h\mathbf{f}(x_n, \mathbf{y}(x_n))\,.$$

Inserting this gives

$$\mathbf{y}(x_{n+1}) \approx \mathbf{y}(x_n) + h\mathbf{f}(x_n, \mathbf{y}(x_n))\,,$$

which motivates the definition of the *explicit Euler method*[2]

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(x_n, \mathbf{y}_n)\,.$$

Two other interesting choices are $\xi = x_{n+1}$ and $\xi = (x_n + x_{n+1})/2$, which give rise to the *implicit Euler method*

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(x_{n+1}, \mathbf{y}_{n+1})$$

and the *implicit midpoint rule*

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(x_n + h/2, (\mathbf{y}_n + \mathbf{y}_{n+1})/2)\,,$$

respectively.

Note the occurrence of $\mathbf{y}_{n+1}$ on the right-hand side of these last two methods. These numerical methods are called *implicit*, because computing $\mathbf{y}_{n+1}$ requires solving a (generally nonlinear) system, which makes them more computationally expensive than explicit Euler. The arising equations are typically solved with Newton's method, which you met in M4 Constructive Mathematics. Explicit methods are faster per timestep, but as we will see often suffer from severe timestep restrictions to retain stability, and implicit methods are usually faster for such problems.

We test these methods on two different examples. First, we consider the linear test case

$$y' = \lambda y\,, \quad y_0 = 1\,, \quad x \in [0, 1]\,. \tag{1}$$

For $\lambda = 3$ and $N = 10$, we observe that all three methods compute a qualitatively correct solution, although the one computed with the implicit midpoint rule is way more

[2]The explicit and the implicit Euler methods have been known since 1768!

accurate. Doubling the value of $N$, we see that the accuracy of the Euler methods improves, although they are never as precise as the implicit midpoint rule.

Next, we investigate what happens for negative values of $\lambda$. This case is interesting because the exact solution converges to 0 exponentially fast. We fix $N = 10$ and investigate different values of $\lambda$. For $\lambda \in [-1, -10]$, we see that all methods provide a qualitatively correct solution. For $\lambda < -10$, we see that the explicit Euler solution start oscillating, becoming equioscillatory for $\lambda = -20$, and diverging for $\lambda < -20$.

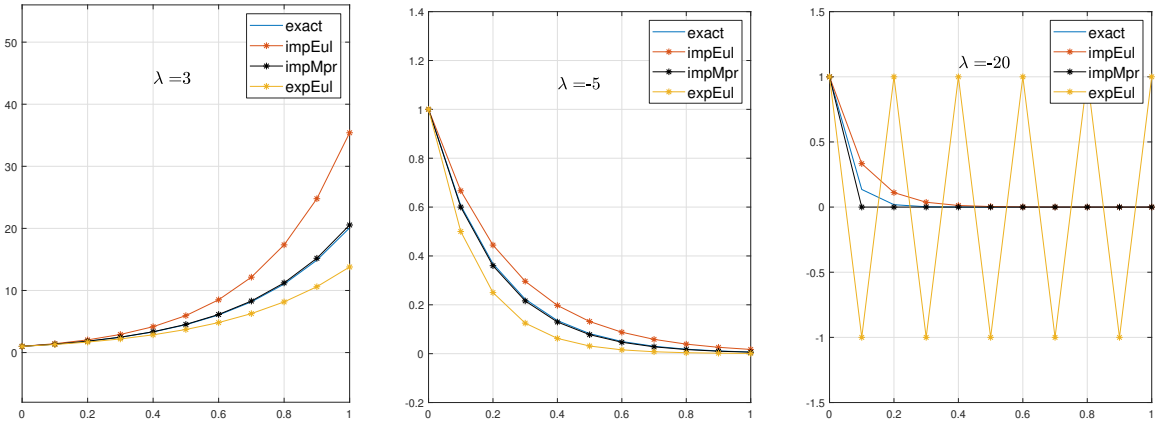



Figure 1: Solving (1) using explicit Euler, midpoint and implicit Euler methods.

For $\lambda < -20$, the solution computed with the implicit midpoint rule also starts to oscillate, although the level of these oscillations cannot be compared with the ones of the explicit Euler method, and the method does not diverge (not even for $\lambda = -9000$). On the other hand, it is surprising to see that the implicit Euler method provides excellent solutions for any negative number of $\lambda$. This example shows that the stability of a numerical method can vary drastically.

The second test case we consider is the following IVP:

$$\mathbf{y}' = \begin{pmatrix} y_2 \\ -y_1 \end{pmatrix}, \quad \mathbf{y}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad x \in [0, 2\pi], \tag{2}$$

whose analytical (exact) solution is $\mathbf{y}(x) = \binom{\cos(x)}{\sin(x)}$. This case is interesting because the quantity $Q(\mathbf{y}) := \|\mathbf{y}(x)\|$ is constant in time. We fix $N = 40$ and plot the orbit (that is, the curve $t \mapsto \mathbf{y}(x)$) of the numerical solutions computed with the three methods above and the evolution of their quantity $Q$.

The numerical solutions are illustrated in Figure 2. We see that the implicit midpoint rule is the only method that preserves $Q$, and that it does it up to machine precision! The laws of physics are typically formulated by considering the conservation laws of energy, mass, momentum, etc., and numerical methods that exactly preserve key structural properties of the underyling models are now of prime importance. This line of thinking has led to a beautiful confluence of numerical analysis with geometry and topology.

### Consistency of a one-step method

**Definition 2.** *A* one-step method *is a function* $\mathbf{\Psi}$ *that takes the triplet* $(s, \mathbf{y}, h) \subset \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}$ *and a function* $\mathbf{f}$*, and computes an approximation* $\mathbf{\Psi}(s, \mathbf{y}, h, \mathbf{f})$ *of* $\mathbf{y}(s + h)$*, which*

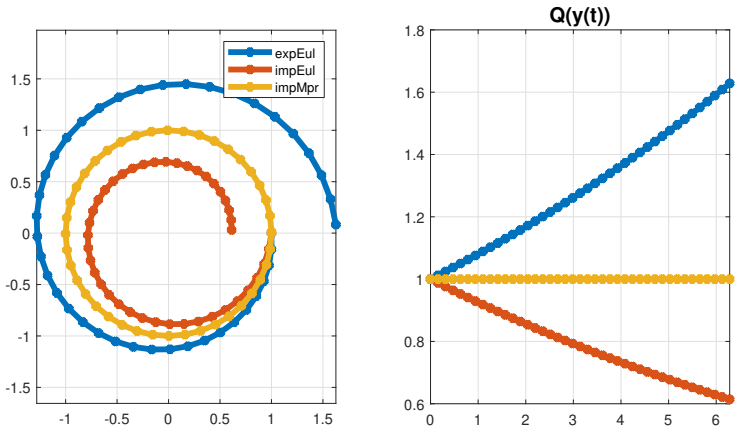



Figure 2: Numerical solutions for (2).

*is the solution at $s+h$ of the IVP*

$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}) , \quad \mathbf{y}(s) = \mathbf{y} .$$

*Here, we tacitly assume that $\mathbf{y}(s+h)$ exists. Additionally, the timestep $h$ may need to be sufficiently small for $\boldsymbol{\Psi}$ to be well defined.*

**Definition 3.** *A one-step method $\boldsymbol{\Psi}$ is said to be* consistent *if*

$$\boldsymbol{\Psi}(s, \mathbf{y}, 0, \mathbf{f}) = \mathbf{y}$$

*and*

$$\frac{\mathrm{d}}{\mathrm{d}h}\boldsymbol{\Psi}(s, \mathbf{y}, h, \mathbf{f})|_{h=0} = \mathbf{f}(s, \mathbf{y}) .$$

**Definition 4.** *The* consistency error $\boldsymbol{\tau}$ *is defined as*

$$\boldsymbol{\tau}(s, \mathbf{y}, h, \mathbf{f}) := \frac{\mathbf{y}(s+h) - \mathbf{y}}{h} - \frac{\boldsymbol{\Psi}(s, \mathbf{y}, h, \mathbf{f}) - \mathbf{y}}{h} = \frac{\mathbf{y}(s+h) - \boldsymbol{\Psi}(s, \mathbf{y}, h, \mathbf{f})}{h} ,$$

*where $\mathbf{y}(s+h)$ is the solution at $s+h$ of the IVP.*

The following lemma gives additional insight about the definition of consistent one-step method. The gist of it is that a one-step method is consistent if the consistency error can be made arbitrarily small by reducing $h$.

**Lemma.** Assume that $h \mapsto \boldsymbol{\Psi}(s, \mathbf{y}, h, \mathbf{f})$ is continuously differentiable in a neighborhood of 0. Then, $\boldsymbol{\Psi}$ is consistent if and only if, for any fixed $\mathbf{f}$,

$$\|\boldsymbol{\tau}(\tilde{s}, \tilde{\mathbf{y}}, h, \mathbf{f})\| \to 0 \quad \text{as } h \to 0$$

locally uniformly in $(\tilde{s}, \tilde{\mathbf{y}}) \in R$, where $R$ is the cylinder from Picard's Theorem.

It is sometimes convenient to represent an abstract one-step method via its *increment function*.

**Lemma.** Assume that $h \mapsto \boldsymbol{\Psi}(s, \mathbf{y}, h, \mathbf{f})$ is continuously differentiable in a neighborhood of 0. Then, $\boldsymbol{\Psi}$ is consistent if and only if there is a continuous increment function $h \mapsto$

$\boldsymbol{\psi}(s, \mathbf{y}, h, \mathbf{f})$ such that

$$\boldsymbol{\Psi}(s, \mathbf{y}, h, \mathbf{f}) = \mathbf{y} + h\boldsymbol{\psi}(s, \mathbf{y}, h, \mathbf{f}), \quad \boldsymbol{\psi}(s, \mathbf{y}, 0, \mathbf{f}) = \mathbf{f}(s, \mathbf{y}).$$

We also define the *global error*

$$e_n := \mathbf{y}(x_n) - \mathbf{y}_n.$$

Then $e := e_N = \mathbf{y}(x_N) - \mathbf{y}_N$ is the error in the solution at $x = X$.

**Theorem.** Let $\boldsymbol{\Psi}$ be a consistent one-step method and assume that its increment function $\boldsymbol{\psi}$ is Lipschitz continuous with respect to $\mathbf{y}$, that is, that there exists a positive constant $L_{\boldsymbol{\psi}}$ such that, for $0 \le h \le h_0$ and for the same region $R$ of Picard's theorem,

$$\|\boldsymbol{\psi}(x, \mathbf{y}, h, \mathbf{f}) - \boldsymbol{\psi}(x, \mathbf{z}, h, \mathbf{f})\| \le L_{\boldsymbol{\psi}}\|\mathbf{y} - \mathbf{z}\| \quad \text{for} (x, \mathbf{y}), (x, \mathbf{z}) \text{ in } R.$$

Then, assuming that $(x_n, \mathbf{y}_n)$ remains in $R$, it follows that

$$e \le \left( \frac{\exp\left(L_{\boldsymbol{\psi}}(x_N - x_0)\right) - 1}{L_{\boldsymbol{\psi}}} \right) \max_{n=0,\ldots,N-1} \|\boldsymbol{\tau}(x_n, \mathbf{y}(x_n), h, \mathbf{f})\|.$$

**Proof.** For a generic $n \in \{1, \ldots, N-1\}$,

$$\begin{aligned}
e_{n+1} &= \|\mathbf{y}(x_{n+1}) - \mathbf{y}_{n+1}\|, \\
&= \|\mathbf{y}(x_{n+1}) - \boldsymbol{\Psi}(x_n, \mathbf{y}_n, h, \mathbf{f})\|, \\
&= \|\mathbf{y}(x_{n+1}) - \boldsymbol{\Psi}(x_n, \mathbf{y}(x_n), h, \mathbf{f}) + \boldsymbol{\Psi}(x_n, \mathbf{y}(x_n), h, \mathbf{f}) - \boldsymbol{\Psi}(x_n, \mathbf{y}_n, h, \mathbf{f})\|, \\
&\le \|\mathbf{y}(x_{n+1}) - \boldsymbol{\Psi}(x_n, \mathbf{y}(x_n), h, \mathbf{f})\| + \|\boldsymbol{\Psi}(x_n, \mathbf{y}(x_n), h, \mathbf{f}) - \boldsymbol{\Psi}(x_n, \mathbf{y}_n, h, \mathbf{f})\|, \\
&= h\|\boldsymbol{\tau}(x_n, \mathbf{y}(x_n), h, \mathbf{f})\| + \|\left(\mathbf{y}(x_n) + h\boldsymbol{\psi}(x, \mathbf{y}(x_n), h, \mathbf{f})\right) - \left(\mathbf{y}_n + h\boldsymbol{\psi}(x, \mathbf{y}_n, h, \mathbf{f})\right)\|, \\
&\le h\|\boldsymbol{\tau}(x_n, \mathbf{y}(x_n), h, \mathbf{f})\| + \|\mathbf{y}(x_n) - \mathbf{y}_n\| + h\|\boldsymbol{\psi}(x, \mathbf{y}(x_n), h, \mathbf{f}) - \boldsymbol{\psi}(x, \mathbf{y}_n, h, \mathbf{f})\|, \\
&= h\|\boldsymbol{\tau}(x_n, \mathbf{y}(x_n), h, \mathbf{f})\| + e_n + h\|\boldsymbol{\psi}(x, \mathbf{y}(x_n), h, \mathbf{f}) - \boldsymbol{\psi}(x, \mathbf{y}_n, h, \mathbf{f})\|, \\
&\le h\|\boldsymbol{\tau}(x_n, \mathbf{y}(x_n), h, \mathbf{f})\| + e_n + hL_{\boldsymbol{\psi}}\|\mathbf{y}(x_n) - \mathbf{y}_n\|, \\
&= h\|\boldsymbol{\tau}(x_n, \mathbf{y}(x_n), h, \mathbf{f})\| + (1 + hL_{\boldsymbol{\psi}})e_n.
\end{aligned}$$

Iterating recursively, this implies that (note that $e_0 = 0$)

$$\begin{aligned}
e_{n+1} &\le (1 + hL_{\boldsymbol{\psi}})^{n+1}e_0 + h\textstyle\sum_{k=0}^{n}(1 + hL_{\boldsymbol{\psi}})^k \max_{m=0,\ldots,n} \|\boldsymbol{\tau}(x_m, \mathbf{y}(x_m), h, \mathbf{f})\| \\
&= \tfrac{(1+hL_{\boldsymbol{\psi}})^{n+1}-1}{L_{\boldsymbol{\psi}}} \max_{m=0,\ldots,n} \|\boldsymbol{\tau}(x_m, \mathbf{y}(x_m), h, \mathbf{f})\|.
\end{aligned}$$

To conclude the proof, note that $1 + hL_{\boldsymbol{\psi}} \le \exp hL_{\boldsymbol{\psi}}$. □

A method is said to have *order of accuracy* (or just *order*) $p$ if $e \le Ch^p$ for some constant $C$. The related notion of *consistency order* is the largest $\tilde{p}$ such that the consistency error $\|\boldsymbol{\tau}(s, \mathbf{y}, h, \mathbf{f})\| \le \tilde{C}h^{\tilde{p}}$ for some $\tilde{C}$. The consistency order $\tilde{p}$ measures the local error whereas $p$ does the global error; they are usually the same, as the above theorem suggests.

Listing 1: l11_ivp1.m

```
clear, set(0,'DefaultFigureWindowStyle','docked')
N = 10; h = 1/N; lambda = -20; %modify these parameters to experiment
```

```
expEul = nan(1, N+1); impEul = nan(1, N+1); impMpr = nan(1, N+1);
y0 = 1; expEul(1) = y0; impEul(1) = y0; impMpr(1) = y0;

for ii = 1:N
    expEul(ii+1) = expEul(ii)*(1+h*lambda);
    impEul(ii+1) = impEul(ii)/(1-h*lambda);
    impMpr(ii+1) = impMpr(ii)*(1+h*lambda/2)/(1-h*lambda/2);
end

t = linspace(0, 1, N+1);figure(1);
plot(t, exp(lambda*t), t, impEul, '*-', t, impMpr, 'k*-',t, expEul, '*-')
legend({'exact', 'impEul', 'impMpr','expEul'})
```

Listing 2: l11_ivp2.m

```
clear, set(0,'DefaultFigureWindowStyle','docked')
N = 40; T = 2*pi; h = T/N; A = [0 1; -1 0];
expEul = nan(2, N+1); impEul = nan(2, N+1); impMpr = nan(2, N+1);
y0 = [1; 0]; expEul(:,1) = y0; impEul(:,1) = y0; impMpr(:,1) = y0;

for ii = 1:N
    expEul(:,ii+1) = (eye(2)+h*A)*expEul(:,ii);
    impEul(:,ii+1) = (eye(2)-h*A)\impEul(:,ii);
    impMpr(:,ii+1) = (eye(2)-h*A/2)\((eye(2)+h*A/2)*impMpr(:,ii));
end
figure(2); subplot(1,2,1);
plot(expEul(1,:), expEul(2,:), '*-', 'linewidth', 4);
plot(impEul(1,:), impEul(2,:), '*-', 'linewidth', 4);
plot(impMpr(1,:), impMpr(2,:), '*-', 'linewidth', 4);
axis equal
legend({'expEul', 'impEul', 'impMpr'})
subplot(1,2,2), t = linspace(0, T, N+1); Q = @(y) sqrt(sum(y.^2, 1));
plot(t, Q(expEul), '*-', t, Q(impEul), '*-', t, Q(impMpr), '*-','linewidth', 4)
title('Q(y(t))','FontSize',24);
fprintf('max(abs(Q(impMpr)-1)) = %e\n', max(abs(Q(impMpr)-1)))
```

**Numerical Analysis Hilary Term 2021**
**Lecture 14: Runge–Kutta methods**

**Runge–Kutta methods:** Runge–Kutta (RK) methods form a broad class of algorithms for the numerical solution of IVPs. The class includes both explicit and implicit schemes. When applications call for an integrator with some kind of stability or conservation property, there usually exists a suitable RK method. In particular, RK methods can be made arbitrarily high-order without the loss of stability.

Here we state some results without proof; they are nonexaminable. For a detailed discussion, we refer to the books

- Süli and Mayer, "Introduction to Numerical Analysis"
- Hairer, Norsett, and Wanner, "Solving Ordinary Differential Equations"
- Butcher, "Numerical Methods for Ordinary Differential Equations"

**Definition 1.** *The family of* $s$-stage Runge–Kutta methods *is defined by*

$$\boldsymbol{\Psi}(x, \mathbf{y}, h, \mathbf{f}) = \mathbf{y} + h \sum_{i=1}^{s} b_i \mathbf{k}_i \,, \tag{1}$$

*where the* stages $\mathbf{k}_i s$ *(recall that* $\mathbf{y} \in \mathbb{R}^d$*, and so do the* $\mathbf{k}_i s$*) are the solutions of the coupled system of (generally nonlinear) equations*

$$\mathbf{k}_i := \mathbf{f}(x + c_i h, \mathbf{y} + h \sum_{j=1}^{s} a_{ij} \mathbf{k}_j) \,, \quad i = 1, \ldots, s \,. \tag{2}$$

*The coefficients* $\{c_i\}_{i=1}^s$ *are always given by*

$$c_i := \sum_{j=1}^{s} a_{ij} \quad i = 1, \ldots, s \,.$$

**Definition 2.** *The coefficients of a Runge–Kutta method are commonly summarized in a* Butcher tableau[1]

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^\top \end{array} \,.$$

**Example 3.** *The explicit Euler method, the implicit Euler method, and the implicit midpoint rule are Runge–Kutta methods. Their Butcher tables are*

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array} \,, \quad \begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array} \,, \quad \textit{and} \quad \begin{array}{c|c} 1/2 & 1/2 \\ \hline & 1 \end{array} \,, \quad \textit{respectively.}$$

[1]The use of this tableau was introduced by J. C. Butcher in 1963 with the article *Coefficients for the study of Runge–Kutta integration processes.*

It is convenient at this point to restrict our attention to autonomous IVPs. (Recall that a nonautonomous system can always be made autonomous by increasing its dimension.) The process of making an IVP autonomous commutes with Runge–Kutta discretisation if and only if

$$\sum_{i=1}^{s} b_i = 1, \quad c_k = \sum_{j=1}^{s} a_{kj} \quad k = 1, \ldots, s\,,$$

which we assume henceforth. (In other words, if these conditions hold, the RK discretisation of the autonomised system is the autonomisation of the RK discretisation of the original problem.)

By computing appropriate Taylor expansions, it is possible to derive algebraic conditions the Runge–Kutta coefficients must satisfy for the method to have a targeted consistency order. For example:

**Lemma 4.** *A Runge–Kutta method is consistent if and only if* $\sum_{i=1}^{s} b_i = 1$. *If the condition*

$$\sum_{i=1}^{s} b_i c_i = \frac{1}{2}$$

*is also satisfied, the Runge–Kutta method has consistency order 2, and if the conditions*

$$\sum_{i=1}^{s} b_i c_i^2 = \frac{1}{3} \quad \textit{and} \quad \sum_{i=1}^{s} b_i \sum_{j=1}^{s} a_{ij} c_j = \frac{1}{6}$$

*are also satisfied, the Runge–Kutta method has consistency order 3.*

The following table indicates the number of conditions as described above that a Runge–Kutta method must satisfy to have order $p$:

| $p$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| #conditions | 1 | 2 | 4 | 8 | 17 | 37 | 85 | 200 | 486 | 1205 | 20247374 |

.

The number of stages of a Runge–Kutta method provides an interesting upper bound on its consistency order.

**Lemma 5.** *The (consistency) order $p$ of an $s$-stage Runge–Kutta method is bounded by $p \leq 2s$. If the Runge–Kutta method is explicit, then $p \leq s$.*

To evolve a numerical solution from $x_n$ to $x_{n+1}$ with a Runge–Kutta method, one needs to compute the stages $\mathbf{k}_i$. If the Runge–Kutta method is explicit, these stages can be computed sequentially (and at a low-cost) starting from $\mathbf{k}_1$ (a Runge–Kutta method is explicit if $a_{ij} = 0$ whenever $j \geq i$, i.e. the matrix $\mathbf{A}$ is strictly lower-triangular). An example of this is the explicit Euler method. If $\mathbf{A}$ is lower-triangular (i.e. possibly $a_{ii} \neq 0$), then the scheme is said to be *diagonally-implicit*; one can compute the stages $\mathbf{k}_i$ sequentially, solving a sequence of nonlinear problems. The implicit Euler and implicit midpoint rules are examples of diagonally-implicit RK methods. Finally, if $\mathbf{A}$ enjoys neither of these structures, the RK method is said to be fully implicit; one must solve a large coupled nonlinear system for all stages simultaneously.

It is possible to construct Runge–Kutta methods that achieve maximal order. So-called *Butcher barriers* quantify the minimal amount of stages that an explicit Runge–Kutta method of order $p$ requires. The following table shows some of these minimal amount of stages:

| $p$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | $\geq 9$ |
|---|---|---|---|---|---|---|---|---|---|
| minimal value of $s$ | 1 | 2 | 3 | 4 | 6 | 7 | 9 | 11 | $\geq p+3$ |

This implies that a Runge–Kutta method that has maximal order must be implicit.

**Construction of explicit RK methods:** To construct explicit Runge–Kutta methods, we start by recalling that the analytic solution of

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}) \,, \qquad \mathbf{y}(x_0) = \mathbf{y}_0 \,, \tag{3}$$

is given by the (implicit) formula

$$\mathbf{y}(x+h) = \mathbf{y}(x) + \int_x^{x+h} \mathbf{f}(\tau, \mathbf{y}(\tau)) \, \mathrm{d}\tau = \mathbf{y}(x) + h \int_0^1 \mathbf{f}(x + h\tau, \mathbf{y}(x + h\tau)) \, \mathrm{d}\tau \,.$$

Approximating the latter integral with a quadrature rule on $[0, 1]$ with $s$ nodes $c_1, \ldots, c_s$ and weights $b_1, \ldots, b_s$ returns

$$\mathbf{y}(x+h) \approx \mathbf{y}(x) + h \sum_{i=1}^{s} b_i \mathbf{f}(x + c_i h, \mathbf{y}(x + c_i h)) \,. \tag{4}$$

Note that this approximation requires the values $\mathbf{y}(x + c_i h)$. To make the method explicit, we approximate the values $\mathbf{y}(x_0 + c_i h)$ with explicit Runge–Kutta methods we already know. This way, we can construct $s$-stage explicit Runge–Kutta methods by induction.

**Example 6.** *If we choose the 1-point Gauss quadrature rule, that is,*

$$\mathbf{y}(x+h) \approx \mathbf{y}(x) + h\mathbf{f}(x + h/2, \mathbf{y}(x + h/2)) \tag{5}$$

*and approximate* $\mathbf{y}(x + h/2)$ *with the explicit Euler method, the resulting scheme reads*

$$\mathbf{\Psi}(x, \mathbf{y}, h, \mathbf{f}) = \mathbf{y} + h\mathbf{f}\left(x + h/2, \mathbf{y} + \frac{h}{2}\mathbf{f}(x, \mathbf{y})\right) \,. \tag{6}$$

**Example 7.** *If we use the trapezium rule, that is,*

$$\mathbf{y}(x+h) \approx \mathbf{y}(x) + \frac{h}{2}\mathbf{f}(x, \mathbf{y}(x)) + \frac{h}{2}\mathbf{f}(x + h, \mathbf{y}(x + h)) \,,$$

*and approximate* $\mathbf{y}(x + h)$ *with the explicit Euler method, the resulting scheme reads*

$$\mathbf{\Psi}(x, \mathbf{y}, h, \mathbf{f}) = \mathbf{y} + \frac{h}{2}\mathbf{f}(x, \mathbf{y}) + \frac{h}{2}\mathbf{f}\left(x + h, \mathbf{y} + h\mathbf{f}(x, \mathbf{y})\right) \,. \tag{7}$$

Both of these are 2nd-order Runge–Kutta methods. Their Butcher tables read

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ \hline & 0 & 1 \end{array} \quad \text{and} \quad \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1/2 & 1/2 \end{array},$$

respectively.

A similar approach leads to the most famous explicit Runge–Kutta method *RK4*, a 4-stage 4th-order explicit Runge–Kutta method whose Butcher table reads

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 2/6 & 2/6 & 1/6 \end{array}.$$

We have seen that $s$-stage explicit Runge–Kutta methods have at most order $s$. Next, we construct $s$-stage implicit Runge–Kutta methods whose order is at least $s$.

**Definition 8.** *Let $c_1, \dots, c_s \in [0,1]$ be (pairwise distinct) collocation points. The corresponding collocation method is the one-step method defined by*

$$\boldsymbol{\Psi}(x, \mathbf{y}, h, \mathbf{f}) = \tilde{\mathbf{y}}(h)\,,$$

*where $\tilde{\mathbf{y}}$ is the unique polynomial of degree $s$ that satisfies*

$$\tilde{\mathbf{y}}(0) = \mathbf{y} \quad \textit{and} \quad \tilde{\mathbf{y}}'(c_i h) = \mathbf{f}(x + c_i h, \tilde{\mathbf{y}}(c_i h))\,, \quad \textit{for } i = 1, \dots, s\,. \tag{8}$$

**Lemma 9.** *Let $Q$ be the highest-order quadrature rule on $[0,1]$ that can be constructed using the nodes $c_1, \dots, c_s$, and let $p_Q$ be its order ($p_Q$ = 1 + the maximal degree of polynomials it integrates exactly). If $\mathbf{f}$ is sufficiently smooth and $h > 0$ is sufficiently small, the collocation method associated to $c_1, \dots, c_s$ has order $p_Q$.*

**Corollary 10.** *If $\mathbf{f}$ is sufficiently smooth and $h > 0$ is sufficiently small, the order of the collocation method associated to $c_1, \dots, c_s$ is at least $s$ and at most $2s$ (Gauss quadrature).*

It is not obvious, but collocation methods are indeed Runge–Kutta methods.

**Lemma 11.** *Collocation methods are Runge–Kutta methods. Their coefficients are*

$$a_{ij} = \int_0^{c_i} L_j(\tau)\,\mathrm{d}\tau\,, \quad b_i = \int_0^1 L_i(\tau)\,\mathrm{d}\tau\,, \tag{9}$$

*where $\{L_i\}_{i=1}^s$ are the Lagrange polynomials associated to $c_1, \dots, c_s$.*

**Stability of Runge–Kutta methods** We have seen that numerical methods for IVPs may encounter stability issues. For simplicity, we only consider autonomous ODEs.

**Definition 12.** *A* fixed point *of* $\mathbf{y}' = \mathbf{f}(\mathbf{y})$ *is a point* $\mathbf{y}^*$ *such that* $\mathbf{f}(\mathbf{y}^*) = \mathbf{0}$. *A fixed point* $\mathbf{y}^*$ *is* asymptotically stable *(or attractive) if there exists a ball* $B_\delta(\mathbf{y}^*)$ *(of radius* $\delta > 0$ *and centered at* $\mathbf{y}^*$*) such that, whenever* $\mathbf{y}_0 \in B_\delta(\mathbf{y}^*)$*, the solution to* $\mathbf{y}' = \mathbf{f}(\mathbf{y})$, $\mathbf{y}(0) = \mathbf{y}_0$ *satisfies* $\lim_{x\to\infty} \mathbf{y}(x) = \mathbf{y}^*$.

**Theorem 13.** *A fixed point* $\mathbf{y}^*$ *of an autonomous ODE is asymptotically stable if*

$$\sigma\left(\mathbf{Df}(\mathbf{y}^*)\right) \subset \mathbb{C}^- := \{z \in \mathbb{C} : \mathrm{Re} z < 0\},$$

*where* $\sigma\left(\mathbf{Df}(\mathbf{y}^*)\right)$ *denotes the set of eigenvalues of the matrix* $\mathbf{Df}(\mathbf{y}^*)$.

This theorem implies that, to study the asymptotic stability of $\mathbf{y}^*$, we can restrict our considerations to the linearised ODE $\mathbf{y}' = \mathbf{Df}(\mathbf{y}^*)(\mathbf{y} - \mathbf{y}^*)$, that is, we can restrict our attention to linear ODEs. To further simplify the analysis, we restrict our attention to a single eigenvalue, yielding the *Dahlquist test equation*

$$y' = zy\,, \quad y(0) = 1\,, \quad \text{and} \quad \mathrm{Re} z < 0\,. \tag{10}$$

Clearly, the solution of the Dahlquist test equation is $y(x) = \exp(zx)$, which satisfies $\lim_{x\to\infty} y(x) = 0$. Therefore, $y^* = 0$ is an attractive fixed point.

The solution of the Dahlquist test equation obtained with a Runge–Kutta method has a special structure:

**Definition 14.** *Let* $\mathbf{\Psi}$ *be a Runge–Kutta method. The function*

$$S : \mathbb{C} \to \mathbb{C}\,, \quad z \mapsto S(z) := \mathbf{\Psi}(0, 1, 1, f : y \mapsto zy)\,,$$

*is called the* stability function *of* $\mathbf{\Psi}$. *To shorten the notation, we henceforth write* $\mathbf{\Psi}(0, 1, 1, z)$ *instead of* $\mathbf{\Psi}(0, 1, 1, f : y \mapsto zy)$.

**Lemma 15.** *If* $\mathbf{\Psi}$ *is a Runge–Kutta method, then* $\mathbf{\Psi}(0, \ell, h, z) = \mathbf{\Psi}(0, 1, 1, zh)\ell$.

**Theorem 16.** *Let* $\{y_k\}_{k\in\mathbb{N}}$ *be the Runge–Kutta solution to the Dahlquist test equation obtained with a time step* $h > 0$. *Then,* $y_k = S(zh)^k$.

**Proof.** By direct computation, we can see that

$$y_1 = \mathbf{\Psi}(0, 1, h, z) = \mathbf{\Psi}(0, 1, 1, zh) = S(zh)$$

and that

$$y_2 = \mathbf{\Psi}(0, y_1, h, z) = \mathbf{\Psi}(0, 1, 1, zh)y_1 = S(zh)y_1 = S(zh)^2\,.$$

Therefore, we conclude that $y_k = S(h\lambda)^k$. □

It is desirable that the discrete solution $\{y_k\}_{k\in\mathbb{N}}$ satisfies $\lim_{k\to\infty} y_k = 0$, mimicking the behavior of the exact solution to the Dahlquist test equation. When this happens, we say that $\{y_k\}_{k\in\mathbb{N}}$ is *asymptotically stable*.

**Definition 17.** *The region in the complex plane*

$$S_{\mathbf{\Psi}} := \{z \in \mathbb{C} : |S(z)| < 1\}$$

*is called the* stability region *of the Runge–Kutta method. Clearly,* $\{y_k\}_{k\in\mathbb{N}}$ *is asymptotically stable if* $zh \in S_\Psi$.

It is not so difficult to see that the stability function of an explicit Runge–Kutta method is a polynomial, which implies that $S_{\mathbf{\Psi}}$ is bounded. Therefore, the numerical approximation computed with an explicit Runge–Kutta method cannot be asymptotically stable if the time step $h$ is too large. This is what we saw in our previous numerical experiments. However, the stability function of an implicit Runge–Kutta method is a rational function, and hence may not suffer from this limitation.

**Definition 18.** *A Runge–Kutta method is said to be* $A$-stable[2] *if* $\mathbb{C}^- \subset S_{\mathbf{\Psi}}$.

The Gauss collocation methods form a family of arbitrarily high-order A-stable methods whose stability region is exactly $\mathbb{C}^-$.

$A$-stability guarantees that $\{y_k\}_{k\in\mathbb{N}}$ will eventually converge to zero. However, the decay can be very slow compared to that of the exact solution.

**Example 19.** *Let* $\{y_k\}$ *be the approximate solution to the Dalhquist test equation obtained with the implicit midpoint rule and a fixed step size* $h$. *By direct computation, we can see that stability function of the implicit midpoint rule is*

$$S(z) = \frac{1 + z/2}{1 - z/2}.$$

*The exact solution converges exponentially to zero with rate* $\mathrm{Re}z$. *In particular, the smaller (more negative) the* $\mathrm{Re}z$, *the quicker the convergence. On the other hand,* $\{y_k\}$ *is a geometric sequence with ratio* $S(zh)$. *This also converges to zero, but the more negative the* $\mathrm{Re}z$, *the closer* $|S(zh)|$ *to* 1, *and the slower the decay of* $\{y_k\}$. *This implies that, if* $\mathrm{Re}z \ll 0$, *the qualitative behavior of* $\{y_k\}$ *can be very different from the one of the exact solution.*

Therefore, if the initial value problem has a strongly attractive fixed point, it is advisable to further ensure that $\lim_{\mathrm{Re}z\to-\infty} |S(z)| = 0$.

**Definition 20.** *An A-stable method that further satisfies* $\lim_{\mathrm{Re}z\to-\infty} |S(z)| = 0$ *is said to be* $L$-stable *(or stiffly accurate).*

One can verify that the implicit Euler method is $L$-stable, but it is not the only one. An example of a family of $L$-stable RK methods is the Gauss–Radau family. This is a family of collocation methods where the final quadrature point is fixed to $c_s = 1$ and the remaining points $c_1, \ldots, c_{s-1}$ are chosen to obtain an associated quadrature rule of maximal order $2s - 1$.

[2] This concept was introduced by G. Dahlquist in 1963 with the article *A special stability problem for linear multistep methods.*

**Numerical Analysis Hilary Term 2021**
**Lecture 15–16: Multistep methods**

## Linear multi-step methods

Runge-Kutta methods deliver an approximate solution to

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}) \,, \qquad \mathbf{y}(x_0) = \mathbf{y}_0 \,, \tag{1}$$

but tacitly assume that it is possible to evaluate the right-hand side $\mathbf{f}(x, \mathbf{y})$ anywhere (and use a lot of such function evaluations). Instead, linear multi-step methods require values of $\mathbf{f}$ at grid points only.

**Definition 1.** *Let $X > x_0$ be a final time, $N, k \in \mathbb{N}$, $N \geq k$, $h := (X - x_0)/N$, and $x_n := x_0 + hn$. A* linear $k$-step method *is an iterative method that computes the approximation $\mathbf{y}_{n+k}$ to $\mathbf{y}(x_{n+k})$ by solving*

$$\sum_{j=0}^{k} \alpha_j \mathbf{y}_{n+j} = h \sum_{j=0}^{k} \beta_j \mathbf{f}(x_{n+j}, \mathbf{y}_{n+j}) \,, \tag{2}$$

*where $\{\alpha_j\}_{j=0}^k$ and $\{\beta_j\}_{j=0}^k$ are real coefficients. To avoid degenerate cases, we assume that $\alpha_k \neq 0$ and that $\alpha_0^2 + \beta_0^2 \neq 0$.*

Note that if $\beta_k = 0$, the method is explicit.

It is also possible to construct multi-step methods on nonequidistant grids, and good timestepping software does so for you.

In the same way Runge-Kutta methods are summarized with Butcher tables, linear multi-step methods can be summarized with two polynomials.

**Definition 2.** *For the $k$-step method defined by (2),*

$$\rho(z) = \sum_{j=0}^{k} \alpha_j z^j \quad \textit{and} \quad \sigma(z) = \sum_{j=0}^{k} \beta_j z^j \tag{3}$$

*are called the* first *and* second characteristic polynomials*.*

**Example 3.** *A simple linear 3-step method can be constructed using Simpson's quadrature rule. Indeed,*

$$\begin{aligned} \mathbf{y}(x_{n+1}) &= \mathbf{y}(x_{n-1}) + \textstyle\int_{x_{n-1}}^{x_{n+1}} \mathbf{f}(x, \mathbf{y}(x)) \, \mathrm{d}x \\ &\approx \mathbf{y}(x_{n-1}) + \tfrac{2h}{6} \left( \mathbf{f}(x_{n-1}, \mathbf{y}(x_{n-1})) + 4\mathbf{f}(x_n, \mathbf{y}(x_n)) + \mathbf{f}(x_{n+1}, \mathbf{y}(x_{n+1})) \right) . \end{aligned}$$

*This motivates the following linear 2-step method*

$$\mathbf{y}_{n+2} - \mathbf{y}_n = h \left( \frac{2}{6} \mathbf{f}(x_n, \mathbf{y}_n) + \frac{8}{6} \mathbf{f}(x_{n+1}, \mathbf{y}_{n+1}) + \frac{2}{6} \mathbf{f}(x_{n+2}, \mathbf{y}_{n+2}) \right) \tag{4}$$

*Its first and second characteristic polynomials are*

$$\rho(z) = z^2 - 1 \quad \textit{and} \quad \sigma(z) = \frac{2}{6}(z^2 + 4z + 1) \,. \tag{5}$$

There is a formal calculus that can be used to construct families of multi-step methods.

**Definition 4.** *For a fixed small $h > 0$, we define:*

- *the shift operator* $E : \mathbf{y}(x) \mapsto \mathbf{y}(x+h)$,
- *its inverse* $E^{-1} : \mathbf{y}(x) \mapsto \mathbf{y}(x-h)$,
- *the difference operator* $\Delta : \mathbf{y}(x) \mapsto \mathbf{y}(x) - \mathbf{y}(x-h)$,
- *the identity operator* $\mathbf{I} : \mathbf{y}(x) \mapsto \mathbf{y}(x)$,
- *and the differential operator* $D : \mathbf{y}(x) \mapsto \mathbf{y}'(x)$.

**Lemma 5.** *Suppose that $\mathbf{y}(x)$ is analytic (hence infinitely differentiable) at $x$. Then formally, $hD = -\log(\mathbf{I} - \Delta)$.*

**Proof.** First, using Taylor expansion, we can show that

$$\begin{aligned} E\mathbf{y}(x) &= \mathbf{y}(x) + h\mathbf{y}'(x) + \tfrac{h^2}{2}\mathbf{y}''(x) + \dots \\ &= \mathbf{y}(x) + hD\mathbf{y}(x) + \tfrac{h^2}{2}D^2\mathbf{y}(x) + \dots = \exp(hD)\mathbf{y}(x)\,, \end{aligned}$$

and thus, $E = \exp(hD)$. This implies that $hD = \log(E)$.

Then, using the definition, we see that $E^{-1} = \mathbf{I} - \Delta$, and thus $E = (\mathbf{I} - \Delta)^{-1}$.

Therefore, $hD = \log(E) = \log((\mathbf{I} - \Delta)^{-1}) = -\log(\mathbf{I} - \Delta)$. □

**Example 6.** *We can construct a multi-step method using the previous lemma. Indeed, by Taylor expansion of the logarithm* $\log(1-x) = -\sum_{i=1}^{\infty} x^i/i$,

$$hD = -\log(\mathbf{I} - \Delta) = \left(\Delta + \frac{1}{2}\Delta^2 + \frac{1}{3}\Delta^3 + \dots\right), \tag{6}$$

*and thus*

$$h\mathbf{f}(x_n, \mathbf{y}(x_n)) = \left(\Delta + \frac{1}{2}\Delta^2 + \frac{1}{3}\Delta^3 + \dots\right)\mathbf{y}(x_n)\,. \tag{7}$$

*To construct a family of multi-step methods, we truncate the infinite series at different orders and replace $\mathbf{y}(x_n)$ with $\mathbf{y}_n$. These methods are called* backward differentiation formulas*, and their simplest instances are*

$$\begin{aligned} \mathbf{y}_n - \mathbf{y}_{n-1} &= h\mathbf{f}(x_n, \mathbf{y}_n)\,, && \textit{(implicit Euler)} \\ \tfrac{3}{2}\mathbf{y}_n - 2\mathbf{y}_{n-1} + \tfrac{1}{2}\mathbf{y}_{n-2} &= h\mathbf{f}(x_n, \mathbf{y}_n)\,, \\ \tfrac{11}{6}\mathbf{y}_n - 3\mathbf{y}_{n-1} + \tfrac{3}{2}\mathbf{y}_{n-2} - \tfrac{1}{3}\mathbf{y}_{n-3} &= h\mathbf{f}(x_n, \mathbf{y}_n)\,. \end{aligned}$$

**Example 7.** *Explicit Euler's method arises from truncating the series*

$$hD = \left(\Delta - \frac{1}{2}\Delta^2 - \frac{1}{6}\Delta^3 + \dots\right)E\,, \tag{8}$$

*which can be derived similarly.*

**Example 8.** *Another two important families are the* Adams-Moulton methods *and the* Adams-Bashforth methods*, which originate from the formal equalities*

$$E\Delta = h\left(\mathbf{I} - \tfrac{1}{2}\Delta - \tfrac{1}{12}\Delta^2 - \tfrac{1}{24}\Delta^3 - \tfrac{19}{720}\Delta^4 + \ldots\right) D\,,$$
$$E\Delta = h\left(\mathbf{I} + \tfrac{1}{2}\Delta + \tfrac{5}{12}\Delta^2 + \tfrac{3}{8}\Delta^3 + \tfrac{251}{720}\Delta^4 + \ldots\right) D\,,$$

*respectively.*

*For example, writing* $\mathbf{f}_{n+i} = \mathbf{f}_{n+i}(x_{n+i}, \mathbf{y}_{n+i})$ *for simplicity, the three-step Adams–Moulton method is (an implicit method)*

$$\mathbf{y}_{n+3} = \mathbf{y}_{n+2} + \frac{1}{24}h\left(9\mathbf{f}_{n+3} + 19\mathbf{f}_{n+2} - 5\mathbf{f}_{n+1} - 9\mathbf{f}_n\right),$$

*and the four-step Adams-Bashforth method is (explicit)*

$$\mathbf{y}_{n+4} = \mathbf{y}_{n+3} + \frac{1}{24}h\left(55\mathbf{f}_{n+3} - 59\mathbf{f}_{n+2} + 37\mathbf{f}_{n+1} - 9\mathbf{f}_n\right)$$

To compute $\mathbf{y}_k$ with a linear $k$-step method, we need the values $\mathbf{y}_0, \ldots, \mathbf{y}_{k-1}$. These (except $\mathbf{y}_0$) must be approximated with either a one-step method or another multi-step method that uses fewer steps. At any rate, they will contain numerical errors. Clearly, a meaningful multistep method should be robust with respect to small perturbations of these initial values.

**Definition 9.** *A linear $k$-step method is said to be* zero-stable *if there is a constant $K > 0$ such that for every $N \in \mathbb{N}$ sufficiently large and for any two different sets of initial data $\mathbf{y}_0, \ldots, \mathbf{y}_{k-1}$ and $\tilde{\mathbf{y}}_0, \ldots, \tilde{\mathbf{y}}_{k-1}$, the two sequences $\{\mathbf{y}_n\}_{n=0}^N$ and $\{\tilde{\mathbf{y}}_n\}_{n=0}^N$ that stem from the linear $k$-step method with $h = (X - x_0)/N$ satisfy*

$$\max_{0\le n\le N} \|\mathbf{y}_n - \tilde{\mathbf{y}}_n\| \le K \max_{j\le k-1} \|\mathbf{y}_j - \tilde{\mathbf{y}}_j\|\,. \tag{9}$$

Zero-stability of a $k$-step method can be verified algebraically with the following property, which is known as the *root condition*.

**Definition 10.** *A linear $k$-step method satisfies the* root condition *if all zeros of its first characteristic polynomial $\rho(z)$ lie inside the closed unit disc, and every zero that lies on the unit circle is simple.*

**Theorem 11.** *A linear multi-step method is zero-stable for any ODE $\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y})$ with Lipschitz right-hand side, if and only if the linear multi-step method satisfies the root condition.*

This theorem implies that zero-stability of a multi-step method can be determined by merely considering its behavior when applied to the trivial differential equation $y' = 0$; it is for this reason that it is called *zero*-stability.

**Consistency and convergence**

**Definition 12.** *The* consistency error *of a linear k-step method with* $\sigma(1) \neq 0$ *is*

$$\boldsymbol{\tau}(h) = \frac{\sum_{j=0}^{k} \alpha_j \mathbf{y}(x_j) - h \sum_{j=0}^{k} \beta_j \mathbf{y}'(x_j)}{h \sum_{j=0}^{k} \beta_j}, \tag{10}$$

*where* $\mathbf{y}$ *is a smooth function.*

**Definition 13.** *A linear multi-step method has* (consistency) order $p$ *if* $\boldsymbol{\tau}(h) = O(h^p)$.

By adequate Taylor expansion, we can obtain the following theorem.

**Theorem 14.** *A linear multi-step method has consistency order* $p$ *if and only if* $\sigma(1) \neq 0$ *and*

$$\sum_{j=0}^{k} \alpha_j = 0 \quad \textit{and} \quad \sum_{j=0}^{k} \alpha_j j^q = q \sum_{j=0}^{k} \beta_j j^{q-1} \quad \textit{for} \quad q = 1, \ldots, p\,. \tag{11}$$

**Definition 15.** *A multi-step method is said to be* consistent *if these conditions are satisfied at least for* $p = 1$.

**Theorem 16.** *A linear multi-step method is consistent iff*

$$\rho(1) = 0 \quad \textit{and} \quad \rho'(1) = \sigma(1) \neq 0. \tag{12}$$

In general, these conditions can be reformulated more elegantly.

**Theorem 17.** *Equation (11) is equivalent to* $\rho(e^h) - h\sigma(e^h) = O(h^{p+1})$.

To define the concept of convergence for linear $k$-step methods, we need to specify some criteria about the choice of the starting conditions.

**Definition 18.** *A set of starting conditions* $\mathbf{y}_i = \boldsymbol{\eta}_i(h)$, $i = 0, \ldots, k-1$ *is consistent with the initial value* $\mathbf{y}_0$ *if* $\boldsymbol{\eta}_s(h) \to \mathbf{y}_0$ *as* $h \to 0$ *for every* $s = 0, \ldots, k-1$.

**Definition 19.** *A linear k-step method is* convergent *if, for every initial value problem* $\mathbf{y} = \mathbf{f}(x, \mathbf{y})$, $\mathbf{y}(x_0) = \mathbf{y}_0$ *(that satisfies the assumptions of Picard's theorem) and for any choice of consistent starting conditions*

$$\mathbf{y}_0 = \boldsymbol{\eta}_0(h), \ldots, \mathbf{y}_{k-1} = \boldsymbol{\eta}_{k-1}(h)\,, \tag{13}$$

*we have that*

$$\lim_{h \to 0} \mathbf{y}_N = \mathbf{y}(X) \quad (\textit{with } N = (X - x_0)/h) \tag{14}$$

**Theorem 20 (Dahlquist's Equivalence Theorem).** *For consistent linear k-step method with consistent starting values, zero-stability is necessary and sufficient for convergence.*

*Moreover, if* $\boldsymbol{\tau}(h) = O(h^p)$ *and* $\|\mathbf{y}(x_s) - \boldsymbol{\eta}_s(h)\| = O(h^p)$ *for* $s = 0, \ldots, k-1$, *then* $\max_{0 \leq n \leq N} \|\mathbf{y}(x_n) - \mathbf{y}_n\| = O(h^p)$.

For Runge–Kutta methods, we showed that one can construct $s$-stage methods of order $2s$. Unfortunately, it is not possible to construct linear $k$-step methods of order $2k$ without violating the zero-stability requirement.

**Theorem 21 (The first Dahlquist-barrier).** *The order $p$ of a zero-stable linear $k$-step method satisfies*

- $p \leq k+2$ *if $k$ is even,*
- $p \leq k+1$ *if $k$ is odd,*
- $p \leq k$ *if $\beta_k/\alpha_k \leq 0$ (in particular if the method is explicit).*

**Stability of linear multi-step methods** Similar to one-step methods, stability is investigated by applying a linear multi-step method to the Dahlquist test equation $y' = zy$, $z \in \mathbb{C}$, $y(0) = 1$, and $h = 1$. Recall that the solution to this ODE is $y(x) = \exp(zx)$, that $|y(x)| \to 0$ as $t \to \infty$ whenever $\mathrm{Re}(z) < 0$, and that we call its numerical approximation $\{y_n\}_{n\in\mathbb{N}}$ (absolutely) stable if $y_n \to 0$ as $n \to \infty$ when $\mathrm{Re}(z) < 0$.

Our goal is to investigate when the sequence $\{y_n\}_{n\in\mathbb{N}}$ computed with a linear $k$-step method is stable. First of all, note that the $n$-th iterate $y_n$ satisfies

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = \sum_{j=0}^{k} \beta_j z y_{n+j}\,, \quad \text{or equivalently,} \quad \sum_{j=0}^{k} (\alpha_j - z\beta_j) y_{n+j} = 0\,. \tag{15}$$

With the following lemma from the theory of difference equations, we know that $y_n$ is of the form

$$y_n = p_1(n) r_1^n + \ldots + p_\ell(n) r_\ell^n\,, \tag{16}$$

where the $r_j$s are the roots of the polynomial $\pi(x) = \sum_{j=0}^{k} (\alpha_j - z\beta_j) x^j$, and the $p_j(n)$s are polynomials of degree $m_j - 1$, where $m_j$ is the multiplicity of $r_j$.

**Lemma 22.** *Let $\{\gamma_i\}_{i=0}^{k}$ be real coefficients and let $\{x_i\}_{i=0}^{k-1}$ be initial values. Let $\{x_n\}_{n\in\mathbb{N}}$ be the sequence defined by the kth order linear difference equation*

$$\sum_{i=0}^{k} \gamma_i x_{n+i} = 0 \quad . \tag{17}$$

*Then, $x_n$ is of the form*

$$x_n = p_1(n) r_1^n + \ldots + p_\ell(n) r_\ell^n\,, \tag{18}$$

*where $r_1$, …, $r_\ell$ are the roots of the polynomial $\pi(x) = \sum_{i=0}^{k} \gamma_i x^i$ and $p_1$, …, $p_\ell$ are polynomials of degree $m_1 - 1$, …, $m_\ell - 1$, where $m_i$ is the multiplicity of $r_i$.*

With (16), we can fully analyze the asymptotic behavior of $\{y_n\}_{n\in\mathbb{N}}$. Indeed:

- if $\pi(x)$ has a zero $r_j$ outside the unit disc, than $y_n$ grows as $|r_j|^n$,
- if an $r_j$ is on the unit circle and has multiplicity $m_j > 1$, then $y_n \sim n^{m_j - 1}$,

- otherwise, $y_n \to 0$ geometrically as $n \to \infty$.

This computation shows that the polynomial $\pi$ plays a crucial role in this stability analysis. Therefore, similarly to one-step methods, we introduce the following definitions.

**Definition 23.** *The* stability polynomial *of a linear $k$-step method is*

$$\pi(x) = \pi(x; z) := \sum_{j=0}^{k} (\alpha_j - z\beta_j)x^j = \rho(x) - z\sigma(x)\,. \tag{19}$$

**Definition 24.** *The* stability domain *of a linear multistep method is*

$$S := \{z \in \mathbb{C} : \textit{if } \pi(x; z) = 0, \textit{ then } |x| \le 1\textit{; multiple zeros satisfy } |x| < 1\}\,. \tag{20}$$

Note that $0 \in S$ if the method is zero-stable (as $\pi(x; 0) = \rho(x)$).

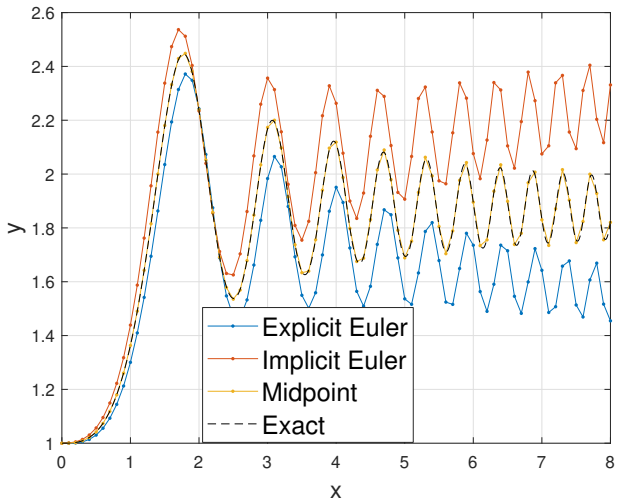
Dahlquist's second barrier theorem places sharp limits on the stability domains of linear multi-step methods.

**Theorem 25 (Dahlquist's second barrier).** *An A-stable linear multi-step method must be implicit and of order $p \le 2$. The trapezium rule is the second-order A-stable linear multi-step method with the smallest error constant.*

It is possible to break the Dahlquist barrier by hybridising between multi-stage and multi-step methods. Such methods are called *general linear methods*[1].

**Example 26.** *We conclude with an example illustrating some of the results. Consider the scalar IVP $y' = \sin(x^2)y$, $y(0) = 1$. We use explicit Euler, implicit Euler, implicit midpoint, explicit 4-stage Runge-Kutta, and 4th order Adam-Bashforth method to solve it.*

*Here are the solutions.*




*We now look at the error $y(x_n) - y_n$, shown in Figure 1. There we also examine the multistep method*

$$\mathbf{y}_{n+2} = -4\mathbf{y}_{n+1} + 5\mathbf{y}_n + h(4\mathbf{f}(x_{n+1}, \mathbf{y}_{n+1}) - 2\mathbf{f}(x_n, \mathbf{y}_n)) \tag{21}$$

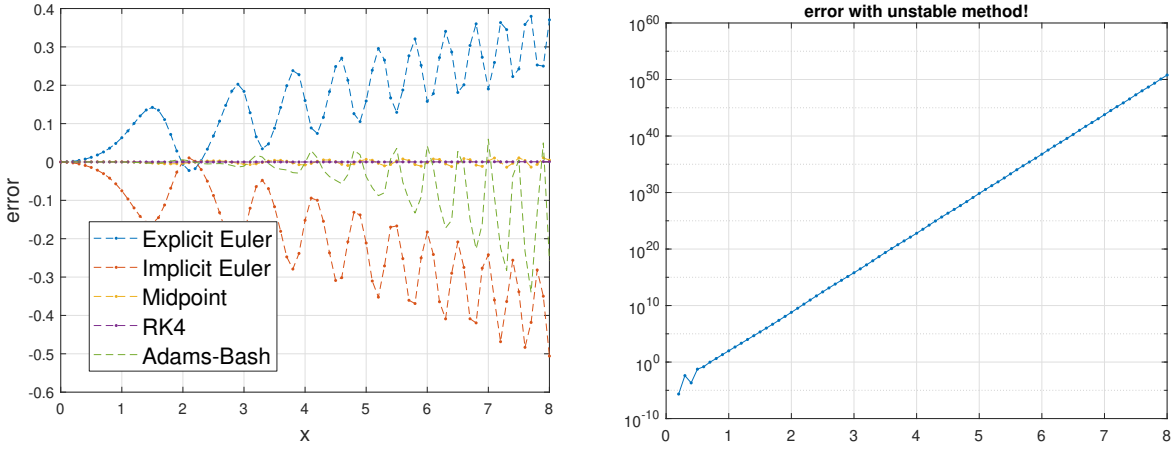
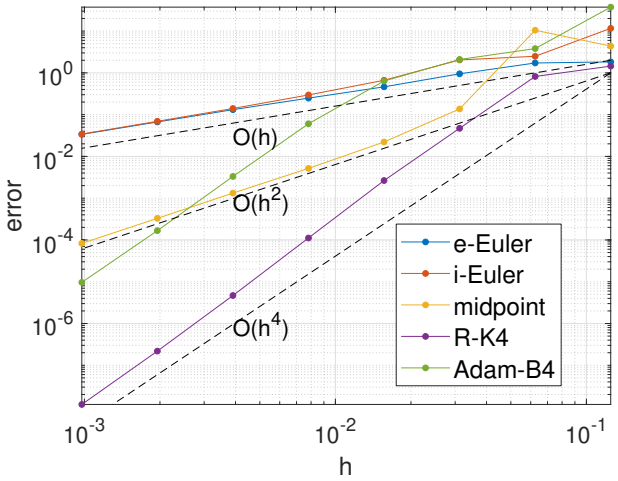



Figure 1: Errors with stable methods (left) and an unstable method (21)

*which has consistency order 3, but is not zero-stable; we thus expect it to not converge. In fact the solution blows up and the error diverges to* $\infty$*—it hardly gets any worse than that!*

*Finally, we can vary the step size* $h$ *and examine the convergence as* $h \to 0$*. Higher-order methods should have better accuracy especially for small* $h$*. We confirm this in the figure (note the loglog scale).*




*(MATLAB code is lec16_demo.m)*

This concludes this course—for further courses related to numerical analysis, check out e.g.

- Numerical Solution of Differential Equations (Part B)
- Approximation of Functions (Part C)
- Numerical Linear Algebra (Part C)
- Finite Element Method for PDEs (Part C)
- Continuous Optimisation (Part C)

[1] See *General linear methods*, J. C. Butcher, Acta Numerica (2006).