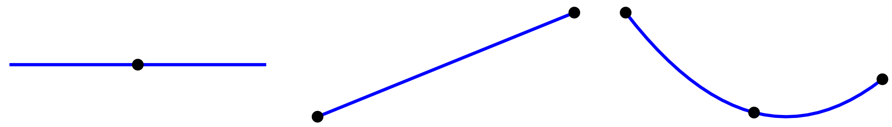
Numerical Analysis Hilary Term 2020
Lecture 1: Lagrange Interpolation

These lecture notes are adapted from the numerical analysis textbook by Süli and Mayers. This first lecture comes from Chapter 6 of the book.

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

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

$\Longrightarrow$

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

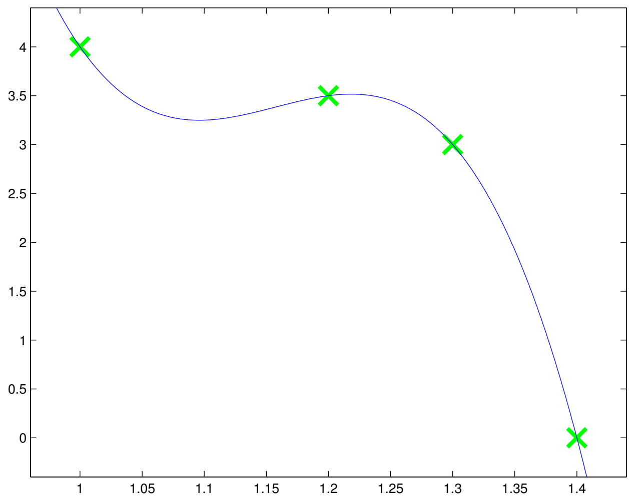
The polynomial (2) is the **Lagrange interpolating polynomial**.

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

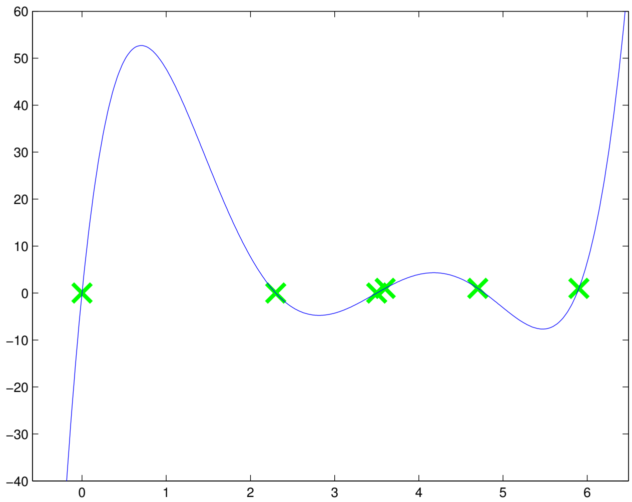
Proof. Consider two interpolating polynomials $p_n, q_n \in \Pi_n$. Their difference $d_n = p_n - q_n \in \Pi_n$ satisfies $d_n(x_k) = 0$ for $k = 0, 1, \ldots, n$. i.e., d_n is a polynomial of degree at most n but has at least $n + 1$ distinct roots. Algebra $\Longrightarrow$ $d_n \equiv 0 \Longrightarrow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

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

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

Now note that ϕ vanishes at $n+2$ points x and x_k, $k = 0, 1, \ldots, n$. $\Longrightarrow$ ϕ' vanishes at $n+1$ points $\xi_0, \ldots, \xi_n$ between these points $\Longrightarrow$ ϕ'' vanishes at n points between these new points, and so on until $\phi^{(n+1)}$ vanishes at an (unknown) point ξ 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). There is a famous example due to Runge, where the error from the interpolating polynomial approximation to $f(x) = (1+x^2)^{-1}$ for $n+1$ equally-spaced points on $[-5, 5]$ diverges near ± 5 as n tends to infinity: try this example with `lagrange` from the website in Matlab[1]

Building Lagrange interpolating polynomials from lower degree ones.

Notation: Let $Q_{i,j}$ be the Lagrange interpolating polynomial at x_k, $k = i, \ldots, j$.

Theorem.

$$Q_{i,j}(x) = \frac{(x - x_i)Q_{i+1,j}(x) - (x - x_j)Q_{i,j-1}(x)}{x_j - x_i} \tag{4}$$

Proof. Let $s(x)$ denote the right-hand side of (4). Because of uniqueness, we simply wish to show that $s(x_k) = f_k$. For $k = i+1, \ldots, j-1$, $Q_{i+1,j}(x_k) = f_k = Q_{i,j-1}(x_k)$, and hence

$$s(x_k) = \frac{(x_k - x_i)Q_{i+1,j}(x_k) - (x_k - x_j)Q_{i,j-1}(x_k)}{x_j - x_i} = f_k.$$

[1] There is a beautiful solution to this issue, Chebyshev interpolation: choose $\{x_k\}$ cleverly, essentially to minimise $\max_{x \in [x_0, x_n]} |(x - x_0)(x - x_1)\cdots(x - x_n)|$ in (3). This results in taking more points near the endpoints. See Trefethen's book Approximation Theory and Approximation Practices, SIAM.

We also have that $Q_{i+1,j}(x_j) = f_j$ and $Q_{i,j-1}(x_i) = f_i$, and hence

$$s(x_i) = Q_{i,j-1}(x_i) = f_i \text{ and } s(x_j) = Q_{i+1,j}(x_j) = f_j.$$

□

Comment: This can be used as the basis for constructing interpolating polynomials. In books: may find topics such as the Newton form and divided differences.

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

$$p_{2n+1}(x) = \sum_{k=0}^{n} [f_k H_{n,k}(x) + g_k K_{n,k}(x)] \tag{5}$$

interpolates the data as required. The polynomial (5) 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 2020
Lecture 2: Newton–Cotes Quadrature

See Chapter 7 of Süli and Mayers.

Terminology: Quadrature $\equiv$ numerical integration

Setup: given $f(x_k)$ at $n+1$ equally spaced points $x_k = x_0 + k \cdot h$, $k = 0, 1, \ldots, n$, where $h = (x_n - x_0)/n$. Suppose that $p_n(x)$ interpolates this data.

Idea: **Approximate and Integrate**. Having obtained the polynomial p_n from data $\{(x_k, f(x_k))\}_{k=0}^n$ by Lagrange interpolation, we can compute the integral $\int_{x_0}^{x_n} p_n(x)\,\mathrm{d}x$. Question:

$$\int_{x_0}^{x_n} f(x)\,\mathrm{d}x \approx \int_{x_0}^{x_n} p_n(x)\,\mathrm{d}x? \tag{1}$$

We investigate the error in such an approximation below, but note that

$$\begin{aligned} \int_{x_0}^{x_n} p_n(x)\,\mathrm{d}x &= \int_{x_0}^{x_n} \sum_{k=0}^{n} f(x_k) \cdot L_{n,k}(x)\,\mathrm{d}x \\ &= \sum_{k=0}^{n} f(x_k) \cdot \int_{x_0}^{x_n} L_{n,k}(x)\,\mathrm{d}x \\ &= \sum_{k=0}^{n} w_k f(x_k), \end{aligned} \tag{2}$$

where the coefficients

$$w_k = \int_{x_0}^{x_n} L_{n,k}(x)\,\mathrm{d}x \tag{3}$$

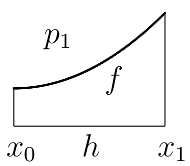
$k = 0, 1, \ldots, n$, are independent of f. A formula

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{k=0}^{n} w_k f(x_k)$$

with $x_k \in [a, b]$ and w_k independent of f for $k = 0, 1, \ldots, n$ is called a **quadrature formula**; the coefficients w_k are known as **weights**. The specific form (1)–(3), based on equally spaced points, is called a **Newton–Cotes formula** of order n.

Examples:

Trapezium Rule: $n = 1$ (also known as the trapezoid or trapezoidal rule):

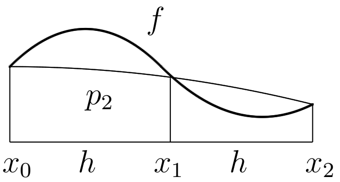


$$\int_{x_0}^{x_1} f(x)\,\mathrm{d}x \approx \frac{h}{2}[f(x_0) + f(x_1)]$$

Proof.

$$\begin{aligned} \int_{x_0}^{x_1} p_1(x)\,\mathrm{d}x &= f(x_0) \int_{x_0}^{x_1} \overbrace{\frac{x - x_1}{x_0 - x_1}}^{L_{1,0}(x)}\,\mathrm{d}x + f(x_1) \int_{x_0}^{x_1} \overbrace{\frac{x - x_0}{x_1 - x_0}}^{L_{1,1}(x)}\,\mathrm{d}x \\ &= f(x_0)\frac{(x_1 - x_0)}{2} + f(x_1)\frac{(x_1 - x_0)}{2} \end{aligned}$$

Simpson's Rule: $n = 2$:



$$\int_{x_0}^{x_2} f(x)\,\mathrm{d}x \approx \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)]$$

Note: The trapezium rule is exact if $f \in \Pi_1$, since if $f \in \Pi_1 \implies p_1 = f$. Similarly, Simpson's Rule is exact if $f \in \Pi_2$, since if $f \in \Pi_2 \implies p_2 = f$. The highest degree of polynomial exactly integrated by a quadrature rule is called the **(polynomial) degree of accuracy** (or degree of exactness).

Error: we can use the error in interpolation directly to obtain

$$\int_{x_0}^{x_n} [f(x) - p_n(x)]\,\mathrm{d}x = \int_{x_0}^{x_n} \frac{\pi(x)}{(n+1)!} f^{(n+1)}(\xi(x))\,\mathrm{d}x$$

so that

$$\left| \int_{x_0}^{x_n} [f(x) - p_n(x)]\,\mathrm{d}x \right| \leq \frac{1}{(n+1)!} \max_{\xi \in [x_0, x_n]} |f^{(n+1)}(\xi)| \int_{x_0}^{x_n} |\pi(x)|\,\mathrm{d}x, \tag{4}$$

which, e.g., for the trapezium rule, $n = 1$, gives

$$\left| \int_{x_0}^{x_1} f(x)\,\mathrm{d}x - \frac{(x_1 - x_0)}{2}[f(x_0) + f(x_1)] \right| \leq \frac{(x_1 - x_0)^3}{12} \max_{\xi \in [x_0, x_1]} |f''(\xi)|.$$

In fact, we can prove a tighter result using the Integral Mean-Value Theorem[1]:

Theorem. $\displaystyle\int_{x_0}^{x_1} f(x)\,\mathrm{d}x - \frac{(x_1 - x_0)}{2}[f(x_0) + f(x_1)] = -\frac{(x_1 - x_0)^3}{12} f''(\xi)$ for some $\xi \in (x_0, x_1)$.

Proof. See problem sheet. □

For $n > 1$, (4) gives pessimistic bounds. But one can prove better results such as:

Theorem. Error in Simpson's Rule: if f'''' is continuous on (x_0, x_2), then

$$\left| \int_{x_0}^{x_2} f(x)\,\mathrm{d}x - \frac{(x_2 - x_0)}{6}[f(x_0) + 4f(x_1) + f(x_2)] \right| \leq \frac{(x_2 - x_0)^5}{720} \max_{\xi \in [x_0, x_2]} |f''''(\xi)|.$$

Proof. Recall $\displaystyle\int_{x_0}^{x_2} p_2(x)\,\mathrm{d}x = \tfrac{1}{3}h[f(x_0) + 4f(x_1) + f(x_2)]$, where $h = x_2 - x_1 = x_1 - x_0$. Consider $f(x_0) - 2f(x_1) + f(x_2) = f(x_1 - h) - 2f(x_1) + f(x_1 + h)$. Then, by Taylor's Theorem,

$$\begin{array}{rcl} f(x_1 - h) & & f(x_1) - hf'(x_1) + \frac{1}{2}h^2 f''(x_1) - \frac{1}{6}h^3 f'''(x_1) + \frac{1}{24}h^4 f''''(\xi_1) \\ -2f(x_1) & = & -2f(x_1) + \\ +f(x_1 + h) & & f(x_1) + hf'(x_1) + \frac{1}{2}h^2 f''(x_1) + \frac{1}{6}h^3 f'''(x_1) + \frac{1}{24}h^4 f''''(\xi_2) \end{array}$$

[1] **Integral Mean-Value Theorem**: if f and g are continuous on $[a, b]$ and $g(x) \geq 0$ on this interval, then there exists an $\eta \in (a, b)$ for which $\displaystyle\int_a^b f(x)g(x)\,\mathrm{d}x = f(\eta)\int_a^b g(x)\,\mathrm{d}x$ (see problem sheet).

for some $\xi_1 \in (x_0, x_1)$ and $\xi_2 \in (x_1, x_2)$, and hence

$$\begin{aligned} f(x_0) - 2f(x_1) + f(x_2) = &\; h^2 f''(x_1) + \tfrac{1}{24} h^4 [f''''(\xi_1) + f''''(\xi_2)] \\ = &\; h^2 f''(x_1) + \tfrac{1}{12} h^4 f''''(\xi_3), \end{aligned} \tag{5}$$

the last result following from the Intermediate-Value Theorem[2] for some $\xi_3 \in (\xi_1, \xi_2) \subset (x_0, x_2)$. Now for any $x \in [x_0, x_2]$, we may use Taylor's Theorem again to deduce

$$\begin{aligned} \int_{x_0}^{x_2} f(x)\,\mathrm{d}x = &\; f(x_1) \int_{x_1-h}^{x_1+h} \mathrm{d}x + f'(x_1) \int_{x_1-h}^{x_1+h} (x - x_1)\,\mathrm{d}x \\ &+ \tfrac{1}{2} f''(x_1) \int_{x_1-h}^{x_1-h} (x-x_1)^2\,\mathrm{d}x + \tfrac{1}{6} f'''(x_1) \int_{x_1-h}^{x_1+h} (x - x_1)^3\,\mathrm{d}x \\ &+ \tfrac{1}{24} \int_{x_1-h}^{x_1+h} f''''(\eta_1(x))(x-x_1)^4\,\mathrm{d}x \\ = &\; 2hf(x_1) + \tfrac{1}{3} h^3 f''(x_1) + \tfrac{1}{60} h^5 f''''(\eta_2) \\ = &\; \tfrac{1}{3} h[f(x_0) + 4f(x_1) + f(x_2)] + \tfrac{1}{60} h^5 f''''(\eta_2) - \tfrac{1}{36} h^5 f''''(\xi_3) \\ = &\; \int_{x_0}^{x_2} p_2(x)\,\mathrm{d}x + \frac{1}{180} \left(\frac{x_2 - x_0}{2} \right)^5 (3f''''(\eta_2) - 5f''''(\xi_3)) \end{aligned}$$

where $\eta_1(x)$ and $\eta_2 \in (x_0, x_2)$, using the Integral Mean-Value Theorem and (5). Thus, taking moduli,

$$\left| \int_{x_0}^{x_2} [f(x) - p_2(x)]\,\mathrm{d}x \right| \le \frac{8}{2^5 \cdot 180} (x_2 - x_0)^5 \max_{\xi \in [x_0, x_2]} |f''''(\xi)|$$

as required. □

Note: Simpson's Rule is exact if $f \in \Pi_3$ since then $f'''' \equiv 0$.

In fact, it is possible to compute a slightly stronger bound.

Theorem. Error in Simpson's Rule II: if f'''' is continuous on (x_0, x_2), then

$$\int_{x_0}^{x_2} f(x)\,\mathrm{d}x = \frac{x_2 - x_0}{6} [f(x_0) + 4f(x_1) + f(x_2)] - \frac{(x_2 - x_0)^5}{2880} f''''(\xi)$$

for some $\xi \in (x_0, x_2)$.

Proof. See Süli and Mayers, Thm. 7.2. □

[2] **Intermediate-Value Theorem**: if f is continuous on a closed interval $[a, b]$, and c is any number between $f(a)$ and $f(b)$ inclusive, then there is at least one number ξ in the closed interval such that $f(\xi) = c$. In particular, since $c = (df(a) + ef(b))/(d + e)$ lies between $f(a)$ and $f(b)$ for any positive d and e, there is a value ξ in the closed interval for which $d \cdot f(a) + e \cdot f(b) = (d + e) \cdot f(\xi)$.

Numerical Analysis Hilary Term 2020
Lecture 3: Newton-Cotes Quadrature (continued)

See Chapter 7 of Süli and Mayers.

Motivation: we've seen oscillations in polynomial interpolation—the Runge phenomenon–for high-degree polynomials.

Idea: split a required integration interval $[a,b] = [x_0, x_n]$ into n equal intervals $[x_{i-1}, x_i]$ for $i = 1, \ldots, n$. Then use a **composite rule**:

$$\int_a^b f(x)\,\mathrm{d}x = \int_{x_0}^{x_n} f(x)\,\mathrm{d}x = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)\,\mathrm{d}x$$

in which each $\int_{x_{i-1}}^{x_i} f(x)\,\mathrm{d}x$ is approximated by quadrature.
Thus rather than increasing the degree of the polynomials to attain high accuracy, instead increase the number of intervals.

Trapezium Rule:

$$\int_{x_{i-1}}^{x_i} f(x)\,\mathrm{d}x = \frac{h}{2}[f(x_{i-1}) + f(x_i)] - \frac{h^3}{12} f''(\xi_i)$$

for some $\xi_i \in (x_{i-1}, x_i)$

Composite Trapezium Rule:

$$\begin{aligned} \int_{x_0}^{x_n} f(x)\,\mathrm{d}x &= \sum_{i=1}^{n} \left[\frac{h}{2}[f(x_{i-1}) + f(x_i)] - \frac{h^3}{12} f''(\xi_i) \right] \\ &= \frac{h}{2}[f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)] + e_h^{\mathrm{T}} \end{aligned}$$

where $\xi_i \in (x_{i-1}, x_i)$ and $h = x_i - x_{i-1} = (x_n - x_0)/n = (b-a)/n$, and the error e_h^{T} is given by

$$e_h^{\mathrm{T}} = -\frac{h^3}{12} \sum_{i=1}^{n} f''(\xi_i) = -\frac{nh^3}{12} f''(\xi) = -(b-a)\frac{h^2}{12} f''(\xi)$$

for some $\xi \in (a,b)$, using the Intermediate-Value Theorem n times. Note that if we halve the stepsize h by introducing a new point halfway between each current pair (x_{i-1}, x_i), the factor h^2 in the error should decrease by four.

Another composite rule: if $[a,b] = [x_0, x_{2n}]$,

$$\int_a^b f(x)\,\mathrm{d}x = \int_{x_0}^{x_{2n}} f(x)\,\mathrm{d}x = \sum_{i=1}^{n} \int_{x_{2i-2}}^{x_{2i}} f(x)\,\mathrm{d}x$$

in which each $\int_{x_{2i-2}}^{x_{2i}} f(x)\,\mathrm{d}x$ is approximated by quadrature.

Simpson's Rule:

$$\int_{x_{2i-2}}^{x_{2i}} f(x)\,\mathrm{d}x = \frac{h}{3}[f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})] - \frac{(2h)^5}{2880} f''''(\xi_i)$$

for some $\xi_i \in (x_{2i-2}, x_{2i})$.

Composite Simpson's Rule:

$$\begin{aligned}\int_{x_0}^{x_{2n}} f(x)\,\mathrm{d}x = & \sum_{i=1}^{n}\left[\frac{h}{3}[f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})] - \frac{(2h)^5}{2880}f''''(\xi_i)\right] \\ = & \frac{h}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \cdots \\ & + 2f(x_{2n-2}) + 4f(x_{2n-1}) + f(x_{2n})] + e_h^{\mathrm{S}}\end{aligned}$$

where $\xi_i \in (x_{2i-2}, x_{2i})$ and $h = x_i - x_{i-1} = (x_{2n} - x_0)/2n = (b-a)/2n$, and the error e_h^{S} is given by

$$e_h^{\mathrm{S}} = -\frac{(2h)^5}{2880}\sum_{i=1}^{n} f''''(\xi_i) = -\frac{n(2h)^5}{2880}f''''(\xi) = -(b-a)\frac{h^4}{180}f''''(\xi)$$

for some $\xi \in (a, b)$, using the Intermediate-Value Theorem n times. Note that if we halve the stepsize h by introducing a new point half way between each current pair (x_{i-1}, x_i), the factor h^4 in the error should decrease by sixteen (assuming f is smooth enough).

Adaptive (or automatic) procedure: if S_h is the value given by Simpson's rule with a stepsize h, then

$$S_h - S_{\frac{1}{2}h} \approx -\frac{15}{16}e_h^{\mathrm{S}}.$$

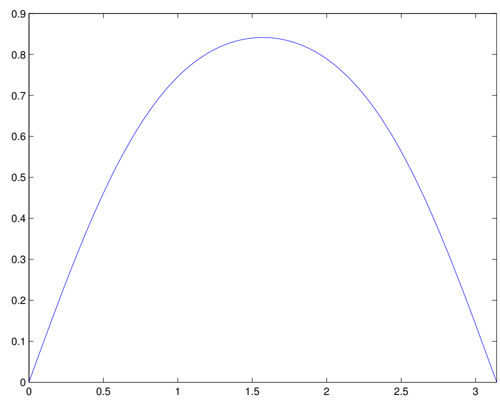
This suggests that if we wish to compute $\int_a^b f(x)\,\mathrm{d}x$ with an absolute error ε, we should compute the sequence $S_h, S_{\frac{1}{2}h}, S_{\frac{1}{4}h}, \ldots$ and stop when the difference, in absolute value, between two consecutive values is smaller than $\frac{16}{15}\varepsilon$. That will ensure that (approximately) $|e_h^{\mathrm{S}}| \le \varepsilon$.

Sometimes much better accuracy may be obtained: for example, as might happen when computing Fourier coefficients, if f is periodic with period $b-a$ so that $f(a+x) = f(b+x)$ for all x.

Matlab:

```
>> help adaptive_simpson
 ADAPTIVE_SIMPSON  Adaptive quadrature with Simpson's rule
    S = ADAPTIVE_SIMPSON(F, A, B, TOL, NMAX) computes an approximation
    to the integral of F on the interval [A, B] . It will take a
    maximum of NMAX steps and will attempt to determine the integral
    to a tolerance of TOL.  If omitted, NMAX will default to 100.

    The function uses an adaptive Simpson's rule, as described
    in lectures.

>> format long g   % see more than 5 digits
>> f = @(x) sin(x);
>> s = adaptive_simpson(f, 0, pi, 1e-7)
Step 1 integral is 2.0943951024.
Step 2 integral is 2.0045597550, with error estimate 0.089835.
Step 3 integral is 2.0002691699, with error estimate 0.0042906.
```

```
Step 4 integral is 2.0000165910, with error estimate 0.00025258.
Step 5 integral is 2.0000010334, with error estimate 1.5558e-05.
Step 6 integral is 2.0000000645, with error estimate 9.6884e-07.
Step 7 integral is 2.0000000040, with error estimate 6.0498e-08.
Successful termination at iteration 7.
s =
         2.00000000403226

>> g = @(x) sin(sin(x));
>> fplot(g, [0 pi])
```




```
>> s = adaptive_simpson(g, 0, pi, 1e-7)
Step 1 integral is 1.7623727094.
Step 2 integral is 1.8011896009, with error estimate 0.038817.
Step 3 integral is 1.7870879453, with error estimate 0.014102.
Step 4 integral is 1.7865214631, with error estimate 0.00056648.
Step 5 integral is 1.7864895607, with error estimate 3.1902e-05.
Step 6 integral is 1.7864876112, with error estimate 1.9495e-06.
Step 7 integral is 1.7864874900, with error estimate 1.2118e-07.
Step 8 integral is 1.7864874825, with error estimate 7.5634e-09.
Successful termination at iteration 8.
s =
          1.7864874824541


>> s = adaptive_simpson(g, 0, pi, 1e-7, 3)
Step 1 integral is 1.7623727094.
Step 2 integral is 1.8011896009, with error estimate 0.038817.
Step 3 integral is 1.7870879453, with error estimate 0.014102.
*** Unsuccessful termination: maximum iterations exceeded ***
The integral *might* be 1.7870879453.
s =
         1.78708794526495
```

Numerical Analysis Hilary Term 2020
Lecture 4: Gaussian Elimination

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.

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

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]

Hence the work in forward substitution is 1 flop to compute x_1 plus 3 flops to compute x_2 plus . . . plus $2i - 1$ flops to compute x_i plus . . . 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.

[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".

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

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.

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

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

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

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.

Numerical Analysis Hilary Term 2020
Lecture 5: 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

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

where I is the identity matrix.

Example: $n = 4$,

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

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

$$\begin{matrix} M(3,2,-l_{32}) & \cdot & M(3,1,-l_{31}) & \cdot & M(2,1,-l_{21}) & \cdot & A = U = (\,\urcorner\,). \\ l_{32} = \dfrac{a_{32}}{a_{22}} & & l_{31} = \dfrac{a_{31}}{a_{11}} & & l_{21} = \dfrac{a_{21}}{a_{11}} & & \text{(upper triangular)} \end{matrix}$$

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

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

This is true for general n:

Theorem. For any dimension n, GE can be expressed as $A = LU$, where $U = (\,\urcorner\,)$ is upper triangular resulting from GE, and $L = (\,\llcorner\,)$ 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

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

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 jth column, find

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

then swap (interchange) rows j and k.

For example,

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

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

Proof. If A is the first matrix above at the jth 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 2020
Lecture 6: 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 $Q = [q_1 \;\; q_2 \;\vdots\; q_n]$, i.e., q_j is the jth column of Q. Then

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

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 set $\iff$ they are linearly independent (check!) $\implies$ they form an orthonormal basis for $\mathbb{R}^n$ as there are n of them. $\square$

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. See problem sheet.
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

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

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

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

Proposition. $H(w)$ is an orthogonal matrix.
Proof.

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

$\square$

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

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

Note
$$\begin{bmatrix} 1 & 0 \\ 0 & H(\hat{w}) \end{bmatrix} = H(w_2), \quad \text{where } w_2 = \begin{bmatrix} 0 \\ \hat{w} \end{bmatrix}.$$

Thus if we continue in this manner for the $n-1$ steps, we obtain
$$\underbrace{H(w_{n-1}) \cdots H(w_3) H(w_2) H(w)}_{Q^{\mathrm{T}}} A = \begin{bmatrix} \alpha & \times & \cdots & \times \\ 0 & \beta & \cdots & \times \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \gamma \end{bmatrix} = (\,\urcorner\,).$$

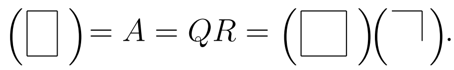
The matrix Q^{T} is orthogonal as it is the product of orthogonal (Householder) matrices, so we have constructively proved that

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 $m \le n \implies$

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



Another useful family of orthogonal matrices are the **Givens rotation** matrices:

$$J(i,j,\theta) = \begin{bmatrix} 1 & & & & & & \\ & \cdot & & & & & \\ & & c & & s & & \\ & & & \cdot & & & \\ & & -s & & c & & \\ & & & & & \cdot & \\ & & & & & & 1 \end{bmatrix} \begin{matrix} \\ \\ \leftarrow\ i\text{th row} \\ \\ \leftarrow\ j\text{th row} \\ \\ \\ \end{matrix}$$

$$\begin{matrix} \uparrow & \uparrow \\ i & j \end{matrix}$$

where $c = \cos\theta$ and $s = \sin\theta$.

Exercise: Prove that $J(i,j,\theta)J(i,j,\theta)^{\mathrm{T}} = I$— obvious though, since the columns form an orthonormal basis.

Note that if $x = (x_1,\ x_2,\ \ldots, x_n)^{\mathrm{T}}$ and $y = J(i,j,\theta)x$, then

$$\begin{aligned} y_k &= x_k \ \text{ for } k \ne i,j \\ y_i &= cx_i + sx_j \\ y_j &= -sx_i + cx_j \end{aligned}$$

and so we can ensure that $y_j = 0$ by choosing $x_i \sin\theta = x_j \cos\theta$, i.e.,

$$\tan\theta = \frac{x_j}{x_i} \ \text{ or equivalently } \ s = \frac{x_j}{\sqrt{x_i^2 + x_j^2}} \ \text{ and } \ c = \frac{x_i}{\sqrt{x_i^2 + x_j^2}}. \tag{1}$$

Thus, unlike the Householder matrices, which introduce lots of zeros by pre-multiplication, the Givens matrices introduce a single zero in a chosen position by pre-multiplication. Since (1) can always be satisfied, we only ever think of Givens matrices $J(i,j)$ for a specific vector or column with the angle chosen to make a zero in the jth position, e.g., $J(1,2)x$ tacitly implies that we choose $\theta = \tan^{-1} x_2/x_1$ so that the second entry of $J(1,2)x$ is zero. Similarly, for a matrix $A \in \mathbb{R}^{m\times n}$, $J(i,j)A := J(i,j,\theta)A$, where $\theta = \tan^{-1} a_{ji}/a_{ii}$, i.e., it is the ith column of A that is used to define θ so that $(J(i,j)A)_{ji} = 0$.

We shall return to these in a later lecture.

Numerical Analysis Hilary Term 2020
Lecture 7: Matrix Eigenvalues

We are concerned with 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 $x(\neq 0) \in \mathbb{C}^n$.

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 $\{\lambda_i, i = 1, \ldots, n\}$ and $\{\mu_i, i = 1, \ldots, n\}$ 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 $i = 1, \ldots, n$ and x_i are linearly independent so that $[x_1, x_2, \ldots, x_n] =:$ 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.
- 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 $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. 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 λ is an eigenvalue of A. Then, λ lies in the union of the **Gerschgorin discs**

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

Proof. If λ 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.,

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

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

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

Then certainly $\sum_{j=1}^{n} a_{kj}x_j = \lambda x_k$, or

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

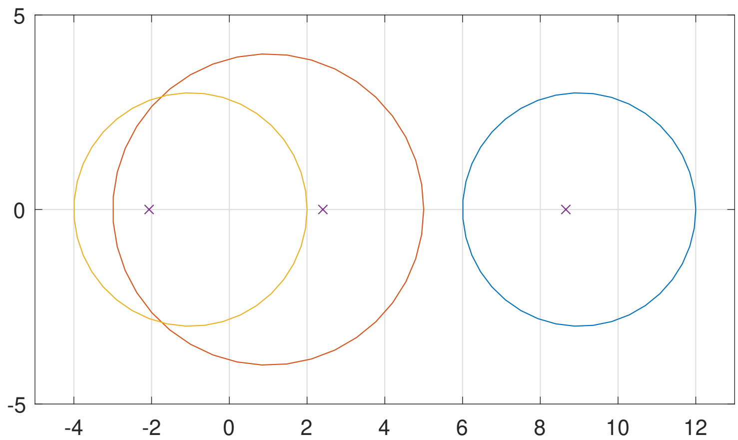
Dividing by x_k, (which, we know, is $\neq 0$) and taking absolute values,

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

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 ℓ (say) discs is disjoint from the other discs, then it contains ℓ eigenvalues.

Proof. Consider $B(\theta) = \theta A + (1 - \theta)D$, where $D = \mathrm{diag}(A)$, the diagonal matrix whose diagonal entries are those from A. As θ 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 θ 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 ℓ Gerschgorin discs by continuity of the eigenvalues it must contain exactly ℓ eigenvalues (as they can't jump!). □

Iterative Methods: methods such as LU or QR factorizations 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 kth iteration (rather than kth 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 **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 λ_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 β_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 β_{k-1}. Thus

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

Therefore, as above,

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

and the sign of λ_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. For simplicity we consider the **QR Algorithm** only in the case when A is symmetric, but the algorithm is applicable also to nonsymmetric matrices.

Numerical Analysis Hilary Term 2020
Lectures 8–9: The Symmetric QR Algorithm

We consider only the case where A is symmetric.

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, ...
  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.

$$A^k = (Q_1 \cdots Q_k)(R_k \cdots R_1) = Q^{(k)} R^{(k)} \tag{1}$$

is the QR factorization of A^k, and

$$A_k = (Q^{(k)})^T A Q^{(k)}. \tag{2}$$

Proof. (2) follows from a repeated application of the above proposition.

We use induction for (1): $k = 1$ 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 $AQ^{(k-1)} = Q^{(k-1)} Q_k R_k$, and so

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

giving (1). □

The lemma shows in particular that the first column q_1 of $Q^{(k)}$ is the result of power method applied k times to the initial vector $e_1 = [1, 0, \ldots, 0]^T$ (verify). 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), (2) 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^{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

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

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^{T} part of the first row of A is unchanged. However, then

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

where $B = H(\hat{w})CH^{\mathrm{T}}(\hat{w})$, as $u^{\mathrm{T}}H(\hat{w})^{\mathrm{T}} = (\alpha, \;\; 0, \;\; \cdots, \;\; 0)$; 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: 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, ...
  form the QR factorization of A_k − μ_k I = Q_k R_k
  and set A_{k+1} = R_k Q_k + μ_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 λ 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 recall (1), and take the inverse:

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

and take the transpose:

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

Noting that $(R^{(k)})^{-T}$ is lower triangular, this shows that the **final** column of $Q^{(k)}$ is the result of power method applied to $e_n = [0, 0, \ldots, 0, 1]^T$ now with the **inverse** A^{-1}. Thus the last column $Q^{(k)}$ is converging to the eigenvector for the smallest eigenvalue λ_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).

Finally, the introduction of shift changes the factor to $|\frac{\lambda_{\sigma(n)}-\mu}{\lambda_{\sigma(n-1)}-\mu}|$, where σ is a permutation such that $|\lambda_{\sigma(1)} - \mu| \geq |\lambda_{\sigma(2)} - \mu| \geq \cdots \geq |\lambda_{\sigma(n)} - \mu|$. If μ is close to an eigenvalue, this implies (potentially very) fast convergence; in fact it can be shown that (proof omitted and non-examinable) rather than linear convergence, $a_{m,m-1}$ 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, ... 2
  while a_{m-1,m} > tol
    [Q, R] = qr(A - a_{m,m} * I)
    A = R * Q + a_{m,m} * I
  end while
  record eigenvalue λ_m = a_{m,m}
  A ← leading m-1 by m-1 submatrix of A
end
record eigenvalue λ_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 λ 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 2020
Lecture 10: 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 ℓ^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 L^∞- or ∞-norm.
3. For integrable functions on (a, b),

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

is the 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 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 P holds *almost everywhere* (a.e.) on $[a, b]$ if property 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) 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 θ 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 $\langle f, g\rangle = \int_0^1 f(x)g(x)\,\mathrm{d}x$ 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 $u, v \in V$,

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

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

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

which is a quadratic in λ. The minimizer of ϕ is at $\lambda_* = \langle u, v\rangle/\|v\|^2$, and thus since $\phi(\lambda_*) \ge 0$, $\|u\|^2 - \langle u, v\rangle^2/\|v\|^2 \ge 0$, 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 $u, v \in V$,

$$\|u + v\| \le \|u\| + \|v\|.$$

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,

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

Taking square-roots yields

$$\|u + v\| \le \|u\| + \|v\|.$$

□

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.

Numerical Analysis Hilary Term 2020
Lecture 11: 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 $f \in \mathrm{L}^2_w(a, b)$ 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

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

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

Proof.

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

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

Remark: the converse is true too (see problem sheet 4).

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 e^x on $[0,1]$ from Π_1 in $\langle f,g\rangle = \int_a^b f(x)g(x)\,\mathrm{d}x$. We want

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

$\iff$

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

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 $A\alpha = 0 \implies \alpha^{\mathrm{T}} A\alpha = 0$

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

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

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

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. □

Remark: This result 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 2020
Lecture 12: 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 Π_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

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

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

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

So A is diagonal if

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

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

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

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

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

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

□

Notes: 1. The G–S procedure does this successively for $k = 0, 1, \ldots, n$.
2. ϕ_k is always of exact degree k, so $\{\phi_0, \ldots, \phi_\ell\}$ is a basis for Π_ℓ $\forall \ell \ge 0$.
3. ϕ_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**,

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

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**,

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

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**,

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

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 $\sigma_k \ne 0$), then $\langle \phi_k, q \rangle = \sigma_k \langle \phi_k, \phi_k \rangle \ne 0$.

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 Π_{k+1}. Now take the inner product on both sides with ϕ_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 ϕ_i and ϕ_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 ϕ_{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. $\square$

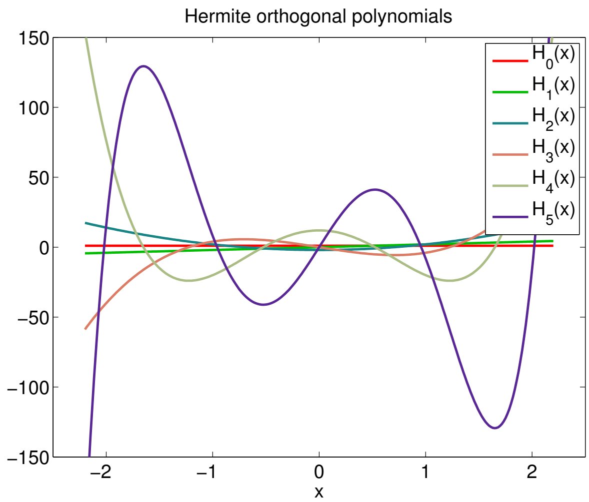
Example. The inner product $\langle f, g\rangle = \int_{-\infty}^{\infty} \mathrm{e}^{-x^2} f(x)g(x)\,\mathrm{d}x$

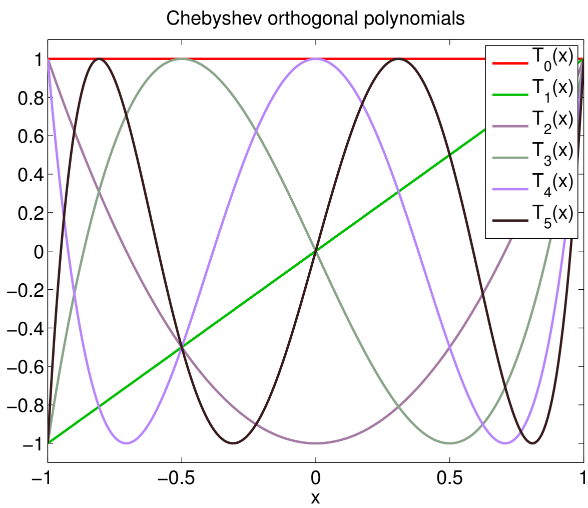
gives orthogonal polynomials called the **Hermite polynomials**,

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

Listing 1: hermite_polys.m

```matlab
%% Demonstrate Hermite Orthogonal Polynomials
lw = 'linewidth';
x = linspace(-2.2, 2.2, 256);
H_old = ones(size(x));
H = figure(1); clf;
plot(x, H_old, 'r-', lw,2)
set(get(H,'children'), 'fontsize', 16);
hold on;  pause

H = 2*x;
plot(x, H, lw,2, 'color',[0 0.75 0])
pause

for n = 1:4
  % use the three-term recurrence
  H_new = (2*x).*H - (2*n)*H_old;
  plot(x, H_new, lw,2, 'color',rand(3,1))
  pause;
  H_old = H;  H = H_new;
end
legend('H_0(x)', 'H_1(x)', 'H_2(x)', 'H_3(x)', 'H_4(x)', 'H_5(x)')
xlabel('x'); title('Hermite orthogonal polynomials')
```





Numerical Analysis Hilary Term 2020
Lecture 13: Gaussian quadrature

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 $k = 0, 1, \ldots$, ϕ_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 ϕ_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., ϕ_k has at least one root in (a, b).

Suppose that there are ℓ points $a < r_1 < r_2 < \cdots < r_\ell < b$ where ϕ_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 ϕ_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 ℓ) must be of degree $\geq k$, i.e., $\ell \geq k$. However, ϕ_k is of exact degree k, and therefore the number of its distinct roots, ℓ, must be $\leq k$. Hence $\ell = k$, and ϕ_k has k distinct roots in (a, b). □

Quadrature revisited. 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

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

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 ϕ_{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 ϕ_{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 **Gaussian 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 $\int_1^2 \frac{1}{x} \, \mathrm{d}x = \ln 2 = 0.6931472\ldots$.

Theorem. Error in Gaussian 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 Gaussian quadrature is exact for polynomials of this degree and the second by interpolation. Thus

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

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

Remark: the "direct" approach of finding Gaussian 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

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

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

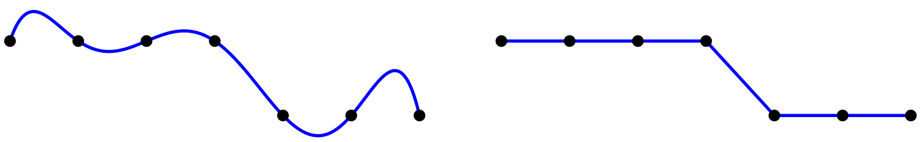
[1] See e.g., the research paper by Hale and Townsend, "Fast and accurate computation of Guass–Legendre and Gauss–Jacobi quadrature nodes and weights" SIAM J. Sci. Comput. 2013.

Table 1: Abscissas x_j (zeros of Legendre polynomials) and weight factors w_j for Gaussian quadrature: $\int_{-1}^{1} f(x)\,\mathrm{d}x \simeq \sum_{j=0}^{n} w_j f(x_j)$ for $n = 0$ to 6.

	x_j	w_j
$n = 0$	0.000000000000000e+0	2.000000000000000e+0
$n = 1$	5.773502691896258e−1	1.000000000000000e+0
	−5.773502691896258e−1	1.000000000000000e+0
	7.745966692414834e−1	5.555555555555556e−1
$n = 2$	0.000000000000000e+0	8.888888888888889e−1
	−7.745966692414834e−1	5.555555555555556e−1
	8.611363115940526e−1	3.478548451374539e−1
$n = 3$	3.399810435848563e−1	6.521451548625461e−1
	−3.399810435848563e−1	6.521451548625461e−1
	−8.611363115940526e−1	3.478548451374539e−1
	9.061798459386640e−1	2.369268850561891e−1
	5.384693101056831e−1	4.786286704993665e−1
$n = 4$	0.000000000000000e+0	5.688888888888889e−1
	−5.384693101056831e−1	4.786286704993665e−1
	−9.061798459386640e−1	2.369268850561891e−1
	9.324695142031520e−1	1.713244923791703e−1
	6.612093864662645e−1	3.607615730481386e−1
$n = 5$	2.386191860831969e−1	4.679139345726910e−1
	−2.386191860831969e−1	4.679139345726910e−1
	−6.612093864662645e−1	3.607615730481386e−1
	−9.324695142031520e−1	1.713244923791703e−1
	9.491079123427585e−1	1.294849661688697e−1
	7.415311855993944e−1	2.797053914892767e−1
	4.058451513773972e−1	3.818300505051189e−1
$n = 6$	0.000000000000000e+0	4.179591836734694e−1
	−4.058451513773972e−1	3.818300505051189e−1
	−7.415311855993944e−1	2.797053914892767e−1
	−9.491079123427585e−1	1.294849661688697e−1

Numerical Analysis Hilary Term 2020
Lectures 14–15: Piecewise Polynomial Interpolation: Splines

Sometimes a 'global' approximation like Lagrange interpolation is not appropriate, e.g., for 'rough' data.



On the left the Lagrange interpolant p_6 'wiggles' through the points, while on the right a **piecewise** linear interpolant ('join the dots'), or linear **spline** interpolant, s appears to represent the data better.

Remark: for any given data s clearly exists and is unique.
Suppose that $a = x_0 < x_1 < \cdots < x_n = b$. Then, s is linear on each interval $[x_{i-1}, x_i]$ for $i = 1, \ldots, n$ and continuous on $[a, b]$. The x_i, $i = 0, 1, \ldots, n$, are called the **knots** of the **linear spline**.

Notation: $f \in \mathrm{C}^k[a, b]$ if $f, f', \ldots, f^k$ exist and are continuous on $[a, b]$.

Theorem. Suppose that $f \in \mathrm{C}^2[a, b]$. Then,

$$\|f - s\|_\infty \leq \frac{1}{8} h^2 \|f''\|_\infty$$

where $h = \max_{1 \leq i \leq n} (x_i - x_{i-1})$ and $\|f''\|_\infty = \max_{x \in [a,b]} |f''(x)|$.

Proof. For $x \in [x_{i-1}, x_i]$, the error from linear interpolation is

$$f(x) - s(x) = \frac{1}{2} f''(\eta)(x - x_{i-1})(x - x_i)$$

where $\eta = \eta(x) \in (x_{i-1}, x_i)$. However, $|(x - x_{i-1})(x - x_i)| = (x - x_{i-1})(x_i - x) = -x^2 + x(x_{i-1} + x_i) - x_{i-1}x_i$, which has its maximum value when $2x = x_i + x_{i-1}$, i.e., when $x - x_{i-1} = x_i - x = \frac{1}{2}(x_i - x_{i-1})$. Thus, for any $x \in [x_{i-1}, x_i]$, $i = 1, 2, \ldots, n$, we have

$$|f(x) - s(x)| \leq \frac{1}{2} \|f''\|_\infty \max_{x \in [x_{i-1}, x_i]} |(x - x_{i-1})(x - x_i)| = \frac{1}{8} h^2 \|f''\|_\infty. \qquad \square$$

Note that s may have discontinuous derivatives, but is a locally defined approximation, since changing the value of one data point affects the approximation in only two intervals.
To get greater smoothness but retain some 'locality', we can define **cubic splines** $s \in \mathrm{C}^2[a, b]$. For a given 'partition', $a = x_0 < x_1 < \cdots < x_n = b$, these are (generally different!) cubic polynomials in each interval (x_{i-1}, x_i), $i = 1, \ldots, n$, which are 'joined' at each knot to have continuity and continuity of s' and s''. **Interpolating cubic splines** also satisfy $s(x_i) = f_i$ for given data f_i, $i = 0, 1, \ldots, n$.

Remark: if there are n intervals, there are $4n$ free coefficients (four for each cubic 'piece'), *but* $2n$ interpolation conditions (one each at the ends of each interval), $n - 1$ derivative continuity conditions (at $x_1, \ldots, x_{n-1}$) and $n - 1$ second derivative continuity conditions (at the same points), giving a total of $4n-2$ conditions (which are linear in the free coefficients). Thus the spline is not unique. So we need to add two extra conditions to generate a spline that might be unique. There are three common ways of doings this:

(a) specify $s'(x_0) = f'(x_0)$ and $s'(x_n) = f'(x_n)$; or

(b) specify $s''(x_0) = 0 = s''(x_n)$ — this gives a **natural** cubic spline; or

(c) enforce continuity of s''' at x_1 and x_{n-1} (which implies that the first two pieces are the same cubic spline, i.e., on $[x_0, x_2]$, and similarly for the last two pieces, i.e., on $[x_{n-2}, x_n]$, from which it follows that x_1 and x_{n-1} are not knots! — this is usually described as the 'not a knot' end-conditions).

We may describe a cubic spline within the i-th interval as

$$s_i(x) = \begin{cases} a_i x^3 + b_i x^2 + c_i x + d_i & \text{for } x \in (x_{i-1}, x_i) \\ 0 & \text{otherwise} \end{cases}$$

and overall, to ensure interpolation (of f), as

$$s(x) = \begin{cases} \displaystyle\sum_{i=1}^{n} s_i(x) & \text{for } \; x \in [x_0, x_n] \setminus \{x_0, x_1, \dots, x_n\} \\ f(x_i) & \text{for } \; x = x_i, \;\; i = 0, 1, \dots, n. \end{cases}$$

The $4n$ linear conditions for an interpolating cubic spline s are:

$$\begin{array}{ccc} & s_i(x_i^-) = f(x_i) & \\ s_1(x_0) = f(x_0) & s_{i+1}(x_i^+) = f(x_i) & s_n(x_n) = f(x_n) \\ s_1'(x_0) = f'(x_0) \;\; \text{(a)} & s_i'(x_i^-) - s_{i+1}'(x_i^+) = 0 & s_n'(x_n) = f'(x_n) \;\; \text{(a)} \\ \text{or} \;\; s_1''(x_0) = 0 \;\; \text{(b)} & s_i''(x_i^-) - s_{i+1}''(x_i^+) = 0 & \text{or} \;\; s_n''(x_n) = 0 \;\; \text{(b)} \\ & i = 1, \dots, n-1. & \end{array} \tag{1}$$

We may write this as $Ay = g$, with

$$y = (a_1, b_1, c_1, d_1, a_2, \dots, d_{n-1}, a_n, b_n, c_n, d_n)^{\mathrm{T}}$$

and the various entries of g are $f(x_i)$, $i = 0, 1, \dots, n$, and $f'(x_0)$, $f'(x_n)$ and zeros for (a) and zeros for (b).
So if A is nonsingular, this implies that $y = A^{-1}g$, that is there is a unique set of coefficients $\{a_1, b_1, c_1, d_1, a_2, \dots, d_{n-1}, a_n, b_n, c_n, d_n\}$. We now prove that if $Ay = 0$ then $y = 0$, and thus that A is nonsingular for cases (a) and (b) — it is also possible, but more complicated, to show this for case (c).

Theorem. If $f(x_i) = 0$ at the knots x_i, $i = 0, 1, \dots, n$, and additionally $f'(x_0) = 0 = f'(x_n)$ for case (a), then $s(x) = 0$ for all $x \in [x_0, x_n]$.

Proof. Consider

$$\begin{aligned} \int_{x_0}^{x_n} (s''(x))^2 \,\mathrm{d}x &= \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} (s_i''(x))^2 \,\mathrm{d}x \\ &= \sum_{i=1}^{n} \left[s_i'(x) s_i''(x) \right]_{x_{i-1}}^{x_i} - \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} s_i'(x) s_i'''(x) \,\mathrm{d}x \end{aligned}$$

using integration by parts. However,

$$\int_{x_{i-1}}^{x_i} s_i'(x)s_i'''(x)\,\mathrm{d}x = s_i'''(x)\int_{x_{i-1}}^{x_i} s_i'(x)\,\mathrm{d}x = s_i'''(x)\,[s_i(x)]_{x_{i-1}}^{x_i} = 0$$

since $s_i'''(x)$ is constant on the interval (x_{i-1}, x_i) and $s_i(x_{i-1}) = 0 = s_i(x_i)$. Thus, matching first and second derivatives at the knots, telescopic cancellation gives

$$\begin{aligned}\int_{x_0}^{x_n} (s''(x))^2\,\mathrm{d}x = &\ \sum_{i=1}^{n} [s_i'(x)s_i''(x)]_{x_{i-1}}^{x_i} \\ = &\ s_1'(x_1)s_1''(x_1) - s_1'(x_0)s_1''(x_0) \\ &+ s_2'(x_2)s_2''(x_2) - s_2'(x_1)s_2''(x_1) + \cdots \\ &+ s_{n-1}'(x_{n-1})s_{n-1}''(x_{n-1}) - s_{n-1}'(x_{n-2})s_{n-1}''(x_{n-2}) \\ &+ s_n'(x_n)s_n''(x_n) - s_n'(x_{n-1})s_n''(x_{n-1}) \\ = &\ s_n'(x_n)s_n''(x_n) - s_1'(x_0)s_1''(x_0).\end{aligned}$$

However, in case (a), $f'(x_0) = f'(x_n) \implies s_1'(x_0) = 0 = s_n'(x_n)$, while in case (b) $s_1''(x_0) = 0 = s_n''(x_n)$. Thus, either way,

$$\int_{x_0}^{x_n} (s''(x))^2\,\mathrm{d}x = 0,$$

which implies that $s_i''(x) = 0$ and thus $s_i(x) = c_i x + d_i$. Since $s(x_{i-1}) = 0 = s(x_i)$, $s(x)$ is identically zero on $[x_0, x_n]$. □

Constructing cubic splines. Note that (1) provides a constructive method for finding an interpolating spline, but generally this is not used. Motivated by the next result, it is better to find a good basis.

Proposition. The set of natural cubic splines on a given set of knots $x_0 < x_1 < \cdots < x_n$ is a vector space.

Proof. If $p, q \in \mathrm{C}^2[a,b] \implies \alpha p + \beta q \in \mathrm{C}^2[a,b]$ and $p, q \in \Pi_3 \implies \alpha p + \beta q \in \Pi_3$ for every α, $\beta \in \mathbb{R}$. Finally, the natural end-conditions (b) $\implies (\alpha p + \beta q)''(x_0) = 0 = (\alpha p + \beta q)''(x_n)$ whenever p'' and q'' are zero at x_0 and x_n. □

Best spline bases: the **Cardinal splines**, C_i, $i = 0, 1, \ldots, n$, defined as the interpolatory natural cubic splines satisfying

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

are a basis for which

$$s(x) = \sum_{i=0}^{n} f(x_i) C_i(x)$$

is the interpolatory natural cubic spline to f.

Preferred are the **B-splines** (locally) defined by $B_i(x_i) = 1$ for $i = 2, 3, \ldots, n-2$, $B_i(x) \equiv 0$ for $x \notin (x_{i-2}, x_{i+2})$, B_i a cubic spline with knots x_j, $j = 0, 1, \ldots, n$, with special definitions for B_0, B_1, B_{n-1} and B_n.

Example/construction: Cubic B-spline with knots $0, 1, 2, 3, 4$. On $[0, 1]$,

$$B(x) = ax^3$$

for some a in order that B, B' and B'' are continuous at $x = 0$ (recall that $B(x)$ is required to be identically zero for $x < 0$). So

$$B(1) = a, \quad B'(1) = 3a, \quad \text{and} \quad B''(1) = 6a.$$

On $[1, 2]$, since B is a cubic polynomial, using Taylor's Theorem,

$$\begin{aligned} B(x) = & \quad B(1) + B'(1)(x-1) + \frac{B''(1)}{2}(x-1)^2 + \beta(x-1)^3 \\ = & \quad a + 3a(x-1) + 3a(x-1)^2 + \beta(x-1)^3 \end{aligned}$$

for some β, and since we require $B(2) = 1$, then $\beta = 1 - 7a$. Now, in order to continue, by symmetry, we must have $B'(2) = 0$, i.e.,

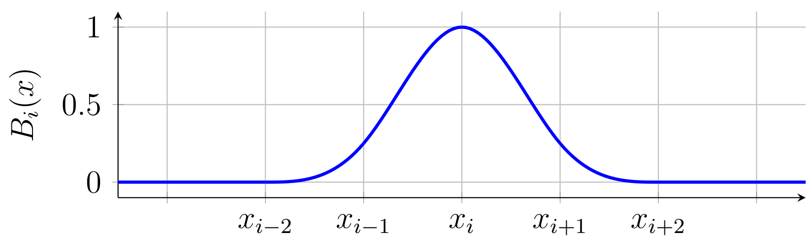
$$3a + 6a(x-1)_{x=2} + 3(1-7a)(x-1)^2_{x=2} = 3 - 12a = 0$$

and hence $a = \frac{1}{4}$. So

$$B(x) = \begin{cases} 0 & \text{for } x < 0 \\ \frac{1}{4}x^3 & \text{for } x \in [0,1] \\ -\frac{3}{4}(x-1)^3 + \frac{3}{4}(x-1)^2 + \frac{3}{4}(x-1) + \frac{1}{4} & \text{for } x \in [1,2] \\ -\frac{3}{4}(3-x)^3 + \frac{3}{4}(3-x)^2 + \frac{3}{4}(3-x) + \frac{1}{4} & \text{for } x \in [2,3] \\ \frac{1}{4}(4-x)^3 & \text{for } x \in [3,4] \\ 0 & \text{for } x > 4. \end{cases}$$

More generally: B-spline on $x_i = a + hi$, where $h = (b-a)/n$.

$$B_i(x) = \begin{cases} 0 & \text{for } x < x_{i-2} \\ \dfrac{(x - x_{i-2})^3}{4h^3} & \text{for } x \in [x_{i-2}, x_{i-1}] \\ -\dfrac{3(x-x_{i-1})^3}{4h^3} + \dfrac{3(x-x_{i-1})^2}{4h^2} + \dfrac{3(x-x_{i-1})}{4h} + \dfrac{1}{4} & \text{for } x \in [x_{i-1}, x_i] \\ -\dfrac{3(x_{i+1}-x)^3}{4h^3} + \dfrac{3(x_{i+1}-x)^2}{4h^2} + \dfrac{3(x_{i+1}-x)}{4h} + \dfrac{1}{4} & \text{for } x \in [x_i, x_{i+1}] \\ \dfrac{(x_{i+2} - x)^3}{4h^3} & \text{for } x \in [x_{i+1}, x_{i+2}] \\ 0 & \text{for } x > x_{i+2}. \end{cases}$$



The 'end' B-splines B_0, B_1, B_{n-1} and B_n are defined analogously by introducing 'phantom' knots x_{-2}, x_{-1}, x_{n+1} and x_{n+2}. The (cubic) B-spline *basis* is only locally affected if some x_i is changed. But note this is not true of the resulting spline itself.

Spline interpolation: find the cubic spline

$$s(x) = \sum_{j=0}^{n} c_j B_j(x),$$

which interpolates f_i at x_i for $i = 0, 1, \ldots, n$. Require

$$f_i = \sum_{j=0}^{n} c_j B_j(x_i) = c_{i-1}B_{i-1}(x_i) + c_i B_i(x_i) + c_{i+1}B_{i+1}(x_i).$$

For equally-spaced data

$$f_i = \tfrac{1}{4}c_{i-1} + c_i + \tfrac{1}{4}c_{i+1},$$

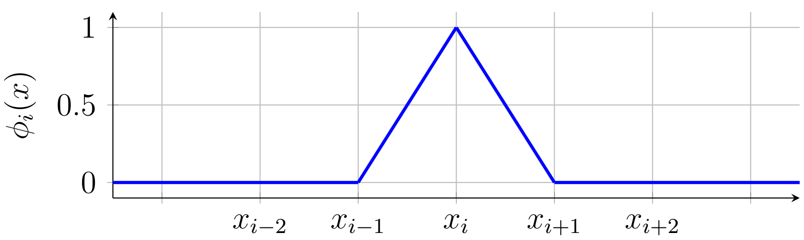
i.e.,

$$\begin{bmatrix} \ddots & \ddots & \ddots & & & & \\ & \frac{1}{4} & 1 & \frac{1}{4} & & & \\ & & \frac{1}{4} & 1 & \frac{1}{4} & & \\ & & & \frac{1}{4} & 1 & \frac{1}{4} & \\ & & & & \ddots & \ddots & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ c_{i-2} \\ c_{i-1} \\ c_i \\ c_{i+1} \\ c_{i+2} \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ f_{i-1} \\ f_i \\ f_{i+1} \\ \vdots \end{bmatrix}.$$

The first few and last few rows of this system depend on the type of spline under consideration. For natural cubic splines, see Problem Sheet 4, Question 20.

For linear splines, a similar local basis of 'hat functions' or **Linear B-splines** $\phi_i(x)$ exist:

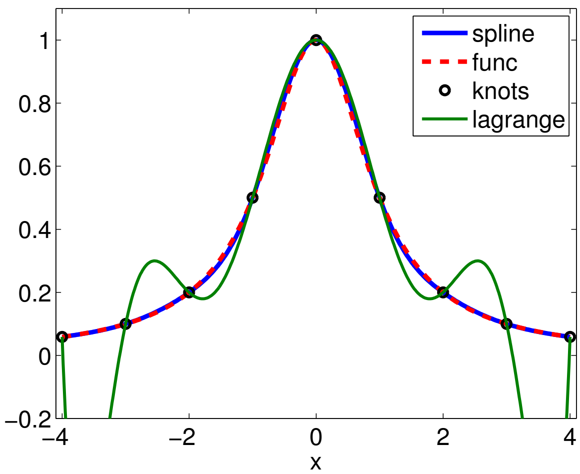
$$\phi_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{x_i - x_{i-1}} & x \in (x_{i-1}, x_i) \\ \dfrac{x - x_{i+1}}{x_i - x_{i+1}} & x \in (x_i, x_{i+1}) \\ 0 & x \notin (x_{i-1}, x_{i+1}) \end{cases}$$



and provide a C^0 piecewise basis.

Listing 1: demo_lec14_spline_vs_lagrange.m

```
N = 9;  % number of interpolation points
x = linspace(-4, 4, N);  % the knots

% values at knots
f = @(x) 1 ./ (1 + x.^2);
fp = @(x) -2*x ./ (1 + x.^2)^2;
ypoints = f(x);

% an extended vector padded with the slope at the first and last
% interpolation points, see "help spline": end-point choices available
% with the matlab command spline (called option (a) in lecture notes).
y = [fp(x(1))  ypoints  fp(x(end))];

% a data structure containing the pieces of the spline
s = spline(x, y);

fine = linspace(-4, 4, 500);
h = figure(1); clf; lw = 'linewidth';
plot(fine, ppval(s, fine), lw,2); % see "help ppval"
ff = f(fine);

% Plot function
hold on
plot(fine, f(fine), 'r--', lw,2);
plot(x, ypoints, 'ko', lw,2)
% Compare to Lagrange interpolating polynomial
p = lagrange_poly(x, ypoints);
plot(fine, polyval(p, fine), 'g-', 'color',[0 0.5 0], lw,2);

set(get(h, 'children'), 'fontsize', 16)
legend('spline', 'func', 'knots', 'lagrange')
ylim([-0.2 1.1]); xlim([-4.1 4.1])
xlabel('x')
```



Error analysis for cubic splines

Theorem. Amongst all functions $t \in \mathrm{C}^2[x_0, x_n]$ that interpolate f at x_i, $i = 0, 1, \ldots, n$, the unique function that minimizes

$$\int_{x_0}^{x_n} [t''(x)]^2 \, \mathrm{d}x$$

is the natural cubic spline s. Moreover, for any such t,

$$\int_{x_0}^{x_n} [t''(x)]^2 \, \mathrm{d}x - \int_{x_0}^{x_n} [s''(x)]^2 \, \mathrm{d}x = \int_{x_0}^{x_n} [t''(x) - s''(x)]^2 \, \mathrm{d}x.$$

Proof. See exercises (uses integration by parts and telescopic cancellation, and is similar to the proof of existence above). □

We will also need:

Lemma. (Cauchy–Schwarz inequality): if $f, g \in \mathrm{C}[a, b]$, then

$$\left[\int_a^b f(x) g(x) \, \mathrm{d}x\right]^2 \leq \int_a^b [f(x)]^2 \, \mathrm{d}x \int_a^b [g(x)]^2 \, \mathrm{d}x.$$

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

$$0 \leq \int_a^b [f(x) - \lambda g(x)]^2 \, \mathrm{d}x = \int_a^b [f(x)]^2 \, \mathrm{d}x - 2\lambda \int_a^b [f(x) g(x)] \, \mathrm{d}x + \lambda^2 \int_a^b [g(x)]^2 \, \mathrm{d}x.$$

The result then follows directly since the discriminant of this quadratic must be nonpositive. □

Theorem. For the natural cubic spline interpolant s of $f \in \mathrm{C}^2[x_0, x_n]$ at $x_0 < x_1 < \cdots < x_n$ with $h = \max_{1 \leq i \leq n}(x_i - x_{i-1})$, we have that

$$\|f' - s'\|_\infty \leq h^{\frac{1}{2}} \left[\int_{x_0}^{x_n} [f''(x)]^2 \, \mathrm{d}x\right]^{\frac{1}{2}} \quad \text{and} \quad \|f - s\|_\infty \leq h^{\frac{3}{2}} \left[\int_{x_0}^{x_n} [f''(x)]^2 \, \mathrm{d}x\right]^{\frac{1}{2}}.$$

Proof. Let $e := f - s$. Take any $x \in [x_0, x_n]$, in which case $x \in [x_{j-1}, x_j]$ for some $j \in \{1, \ldots, n\}$. Then $e(x_{j-1}) = 0 = e(x_j)$ as s interpolates f. So by the Mean Value Theorem, there is a $c \in (x_{j-1}, x_j)$ with $e'(c) = 0$. Hence $e'(x) = \int_c^x e''(t) \, \mathrm{d}t$. Then the Cauchy–Schwarz inequality gives that

$$|e'(x)|^2 \leq \left|\int_c^x \mathrm{d}t\right| \left|\int_c^x [e''(t)]^2 \, \mathrm{d}t\right|.$$

However, the first required inequality then follows since for $x \in [x_{j-1}, x_j]$, $\left|\int_c^x \mathrm{d}t\right| \leq h$ and because the previous theorem gives that

$$\left|\int_c^x [e''(t)]^2 \, \mathrm{d}t\right| \leq \left|\int_c^x [f''(t)]^2 \, \mathrm{d}t\right| \leq \int_{x_0}^{x_n} [f''(x)]^2 \, \mathrm{d}x.$$

The remaining result follows from Taylor's Theorem. □

Theorem. Suppose that $f \in C^4[a,b]$ and s satisfies end-conditions (a). Then,

$$\|f - s\|_\infty \le \frac{5}{384} h^4 \|f^{(4)}\|_\infty$$

and

$$\|f' - s'\|_\infty \le \frac{9+\sqrt{3}}{216} h^3 \|f^{(4)}\|_\infty,$$

where $h = \max_{1 \le i \le n}(x_i - x_{i-1})$.

Proof. Beyond the scope of this course. □

Similar bounds exist for natural cubic splines and splines satisfying end-condition (c).

Numerical Analysis Hilary Term 2020
Lecture 16: Richardson Extrapolation

Extrapolation is based on the general idea that if T_h is an approximation to T, computed by a numerical approximation with (small!) parameter h, and if there is an error formula of the form

$$\begin{aligned} T &= T_h + K_1 h + K_2 h^2 + \cdots + O(h^n) && (1)\\ \text{then } T &= T_k + K_1 k + K_2 k^2 + \cdots + O(k^n) && (2) \end{aligned}$$

for some other value, k, of the small parameter. In this case subtracting (1) from (2) gives

$$(k-h)T = kT_h - hT_k + K_2(kh^2 - hk^2) + \cdots$$

i.e., the linear combination

$$\underbrace{\frac{kT_h - hT_k}{k-h}}_{\text{“extrapolated formula”}} = T + \underbrace{K_2 kh}_{\text{2nd order error}} + \cdots$$

In particular if only *even* terms arise:

$$\begin{aligned} T &= T_h + K_2 h^2 + K_4 h^4 + \cdots + O(h^{2n})\\ \text{and } k = \tfrac{1}{2}h : T &= T_{\frac{h}{2}} + K_2\frac{h^2}{4} + K_4\frac{h^4}{16} + \cdots + O(\frac{h^{2n}}{2^{2n}})\\ \text{then } T &= \frac{4T_{\frac{h}{2}} - T_h}{3} - \frac{K_4}{4}h^4 + \cdots + O(h^{2n}). \end{aligned}$$

This is the first step of **Richardson extrapolation**. Call this new, more accurate formula

$$T_h^{(2)} := \frac{4T_{\frac{h}{2}} - T_h}{3},$$

where $T_h^{(1)} := T_h$. Then the idea can be applied again:

$$\begin{aligned} T &= T_h^{(2)} + K_4^{(2)} h^4 + \cdots + O(h^{2n})\\ \text{and } T &= T_{\frac{h}{2}}^{(2)} + K_4^{(2)}\frac{h^4}{16} + \cdots + O(h^{2n})\\ \text{so } T &= \underbrace{\frac{16T_{\frac{h}{2}}^{(2)} - T_h^{(2)}}{15}}_{T_h^{(3)}} + K_6^{(3)} h^6 + \cdots + O(h^{2n}) \end{aligned}$$

is a more accurate formula again. Inductively we can define

$$T_h^{(j)} := \frac{1}{4^{j-1}-1}\left[4^{j-1}T_{\frac{h}{2}}^{(j-1)} - T_h^{(j-1)}\right]$$

for which

$$T = T_h^{(j)} + O(h^{2j})$$

so long as there are high enough order terms in the error series.

Example: approximation of π by inscribed polygons in unit circle. For a regular n-gon, the circumference $= 2n\sin(\pi/n) \le 2\pi$, so let $c_n = n\sin(\pi/n) \le \pi$, or if we put $h = 1/n$,

$$c_n = \frac{1}{h}\sin(\pi h) = \pi - \frac{\pi^3 h^2}{6} + \frac{\pi^5 h^4}{120} + \cdots$$

so that we can use Richardson extrapolation. Indeed $c_2 = 2$ and

$$\begin{aligned} c_{2n} = & \; 2n\sin(\pi/2n) = 2n\sqrt{\tfrac{1}{2}(1-\cos(\pi/n))} \\ = & \; 2n\sqrt{\tfrac{1}{2}(1-\sqrt{1-\sin^2(\pi/n)})} = 2n\sqrt{\tfrac{1}{2}(1-\sqrt{1-(c_n/n)^2})}\ . \end{aligned}$$

So[1] $c_4 = 2.8284$, $c_8 = 3.0615$, $c_{16} = 3.1214$. Extrapolating between c_4 and c_8 we get $c_4^{(2)} = 3.1391$ and similarly from c_8 and c_{16} we get $c_8^{(2)} = 3.1214$. Extrapolating again between $c_4^{(2)}$ and $c_8^{(2)}$, we get $c_4^{(3)} = 3.141590\ldots$.

Example 2: Romberg Integration. Consider the Composite Trapezium Rule for integrating $T = \int_a^b f(x)\,\mathrm{d}x$:

$$T_h = \frac{h}{2}\left[f(a) + f(b) + 2\sum_{j=1}^{2^n-1} f(x_j)\right]$$

with $x_0 = a$, $x_j = a + jh$ and $h = (b-a)/2^n$. Recall from Lecture 3 that the error is $(b-a)\frac{h^2}{12}f''(\xi)$ for some $\xi \in (a,b)$. If there were an (asymptotic) error series of the form

$$\int_a^b f(x)\,\mathrm{d}x - T_h = K_2h^2 + K_4h^4 + \cdots$$

we could apply Richardson extrapolation as above to yield

$$T - \frac{4T_{\frac{h}{2}} - T_h}{3} = K_4h^4 + \cdots$$

There is such as series: the Euler–Maclaurin formula

$$\begin{aligned} \int_a^b f(x)\,\mathrm{d}x - T_h = & -\sum_{k=1}^{r}\frac{B_{2k}}{(2k)!}h^{2k}[f^{(2k-1)}(b) - f^{(2k-1)}(a)] \\ & +(b-a)\frac{h^{2r+1}B_{2r+2}}{(2r+2)!}f^{(2r+2)}(\xi) \end{aligned}$$

where $\xi \in (a,b)$ and B_{2k} are called the Bernoulli numbers, defined by

$$\frac{x}{e^x - 1} = \sum_{\ell=0}^{\infty} B_l\frac{x^\ell}{\ell!}$$

[1] This expression is sensitive to roundoff errors, so we rewrite it as $c_{2n} = c_n/\sqrt{\frac{1}{2} + \frac{1}{2}\sqrt{1-(c_n/n)^2}}$.

so that $B_2 = \frac{1}{6}$, $B_4 = -\frac{1}{30}$, etc.

Romberg Integration *is* composite Trapezium for $n = 0, 1, 2, 3, ...$, and the repeated application of Richardson extrapolation. Changing notation ($T_h \to T_n$, $h =$ stepsize, $2^n =$ number of composite steps), we have

$$\begin{aligned} T_0 = & \frac{b-a}{2}[f(a) + f(b)] = R_{0,0} \\ T_1 = & \frac{b-a}{4}[f(a) + f(b) + 2f(a + \tfrac{1}{2}(b-a))] \\ = & \tfrac{1}{2}[R_{0,0} + (b-a)f(a + \tfrac{1}{2}(b-a))] = R_{1,0}. \end{aligned}$$

Extrapolation then gives

$$R_{1,1} = \frac{4R_{1,0} - R_{0,0}}{3}.$$

with error $O(h^4)$. Also

$$\begin{aligned} T_2 = & \frac{b-a}{8}[f(a) + f(b) + 2f(a + \tfrac{1}{2}(b-a)) \\ & + 2f(a + \tfrac{1}{4}(b-a)) + 2f(a + \tfrac{3}{4}(b-a))] \\ = & \frac{1}{2}\left[R_{1,0} + \frac{b-a}{2}[f(a + \tfrac{1}{4}(b-a)) + f(a + \tfrac{3}{4}(b-a))\right] = R_{2,0}. \end{aligned}$$

Extrapolation gives

$$R_{2,1} = \frac{4R_{2,0} - R_{1,0}}{3}$$

with error $O(h^4)$. Extrapolation again gives

$$R_{2,2} = \frac{16R_{2,1} - R_{1,1}}{15}$$

now with error $O(h^6)$. At the ith stage

$$T_i = R_{i,0} = \frac{1}{2}\left[R_{i-1,0} + \frac{b-a}{2^{i-1}} \underbrace{\sum_{j=1}^{2^{i-1}} f\left(a + \left(j - \frac{1}{2}\right)\frac{b-a}{2^{i-1}}\right)}_{\text{evaluations at new interlacing points}}\right].$$

Extrapolate

$$R_{i,j} = \frac{4^j R_{i,j-1} - R_{i-1,j-1}}{4^j - 1} \quad \text{for} \quad j = 1, 2, \ldots$$

This builds a triangular table:

	$R_{0,0}$				
	$R_{1,0}$	$R_{1,1}$			
	$R_{2,0}$	$R_{2,1}$	$R_{2,2}$		
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	
	$R_{i,0}$	$R_{i,1}$	$R_{i,2}$	$\ldots$	$R_{i,i}$
Theorem:	Composite Trapezium	Composite Simpson			

Notes 1. The integrand must have enough derivatives for the Euler–Maclaurin series to exist (the whole procedure is based on this!).

2. $R_{n,n} \to \int_a^b f(x)\,\mathrm{d}x$ in general much faster than $R_{n,0} \to \int_a^b f(x)\,\mathrm{d}x$.

A final observation: because of the Euler–Maclaurin series, if $f \in \mathrm{C}^{2n+2}[a,b]$ and is *periodic* of period $b-a$, then $f^{(j)}(a) = f^{(j)}(b)$ for $j = 0, 1, \ldots, 2n-1$, so

$$\int_a^b f(x)\,\mathrm{d}x - T_h = (b-a)\frac{h^{2n+1}B_{2n+2}}{(2n+2)!}f^{(2n+2)}(\xi)$$

c.f.,

$$\int_a^b f(x)\,\mathrm{d}x - T_h = (b-a)\frac{h^2}{12}f''(\xi)$$

for nonperiodic functions! That is, the Composite Trapezium Rule is extremely accurate for the integration of periodic functions. If $f \in \mathrm{C}^{\infty}[a,b]$, then $T_h \to \int_a^b f(x)\,\mathrm{d}x$ faster than any power of h.

Example: the circumference of an ellipse with semi-axes A and B is

$$\int_0^{2\pi} \sqrt{A^2 \sin^2 \phi + B^2 \cos^2 \phi}\ \mathrm{d}\phi.$$

For $A = 1$ and $B = \frac{1}{4}$, $T_8 = 4.2533$, $T_{16} = 4.2878$, $T_{32} = 4.2892 = T_{64} = \cdots$.