Constructor University Bremen Spring Semester 2024

Dr. D. Sinden

JTMS-MAT-13: Numerical Methods

Summary from 4 April 2024

Notes from course on numerical methods. This document can be downloaded from:
`https://djps.github.io/courses/numericalmethods24/notes`

Note that the proofs for theorems marked with an * were presented in class.

Contents

Recommended Reading

- J. F. Epperson "*An Introduction to Numerical Methods and Analysis*", Wiley 2nd Edition (2013).
- R. L. Burden and J. D. Faires "*Numerical Analysis*", Brooks/Cole 9th Edition (2011).

1 Taylor Series

The Taylor series, or the Taylor expansion of a function, is defined as

Definition 1.1 (Taylor Series). For a function $f : \mathbb{R} \mapsto \mathbb{R}$ which is infinitely differentiable at a point c, the Taylor series of $f(c)$ is given by

$$\sum_{k=0}^{\infty} \frac{f^{(k)}(c)}{k!} (x-c)^k .$$

This is a power series, which is convergent for some radius.

Theorem 1 (Taylor's Theorem). For a function $f \in C^{n+1}([a,b])$, i.e. f is $(n+1)$-times continuously differentiable in the interval $[a,b]$, then for some c in the interval, the function can be written as

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(c)}{k!} (x-c)^k + \frac{f^{(n+1)}(\xi)}{(n+1)!} (x-c)^{n+1}$$

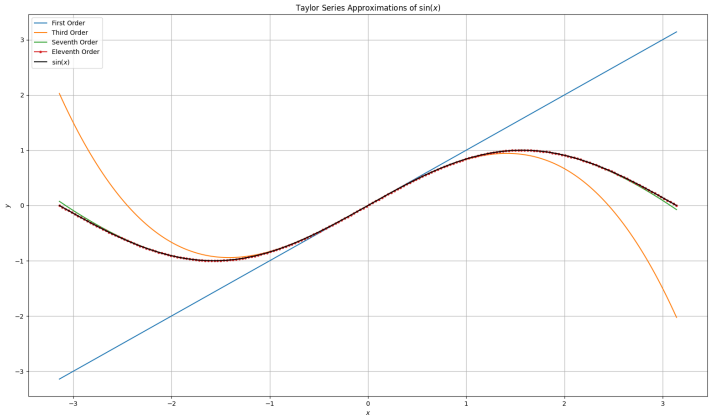
for some value $\xi \in [a,b]$ where

$$\lim_{\xi \to c} \frac{f^{(n+1)}(\xi)}{(n+1)!} (x-c)^{n+1} = 0.$$

Example. With $f(x) = \sin(x)$ around $c = 0$. Thus, as $f' = \cos(x)$, it can be shown that

$$\sin(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \dots$$

Note that in this example, only odd powers of x contribute to the expansion.


Taylor Series Approximations of sin(x)
First Order
Third Order
Seventh Order
Eleventh Order
sin(x)
x
y

2 Errors

Definition 2.1 (Absolute and Relative Errors).
Let $\tilde{a}$ be an approximation to a, then the **absolute error** is given by

$$|\tilde{a} - a|.$$

If $|a| \neq 0$, the **relative error** may be given by

$$\left|\frac{\tilde{a} - a}{a}\right|.$$

The error bound is the magnitude of the admissible error.

Theorem 2. For both addition and subtraction the bounds for the absolute error are added. In division and multiplication the bounds for the relative errors are added.

Definition 2.2 (Linear Sensitivity to Uncertainties).
If $y(x)$ is a smooth function, i.e. is differentiable, then $|y'|$ can be interpreted as the **linear sensitivity** of $y(x)$ to uncertainties in x.

For functions of several variables, i.e. $f : \mathbb{R}^n \to \mathbb{R}$, then

$$|\Delta y| \leq \sum_{i=1}^{n} \left|\frac{\partial y}{\partial x_i}\right| |\Delta x_i|$$

where $|\Delta x_i| = |\tilde{x}_i - x_i|$ for an approximation $\tilde{x}_i$, thus $|\Delta y_i| = |\tilde{y}_i - y_i| = |f(\tilde{x}_i) - (x_i)|$.

3 Number Representations

Definition 3.1 (Base Representation).
Let $b \in \mathbb{N}\backslash\{1\}$. Every number $x \in \mathbb{N}_0$ can be written as a unique expansion with respect to base b as

$$(x)_b = a_0 b^0 + a_1 b^1 + \ldots + a_n b^n = \sum_{i=0}^{n} a_i b^i$$

A number can be written in a nested form:

$$\begin{aligned}(x)_b &= a_0 b^0 + a_1 b^1 + \ldots a_n b^n \\ &= a_0 + b\,(a_1 + b\,(a_2 + b\,(a_3 + \ldots + b a_n)\,\ldots)\end{aligned}$$

with $a_i < \mathbb{N}_0$ and $a_i < b$, i.e. $a_i \in \{0, \ldots, b-1\}$.
For a real number, $x \in \mathbb{R}$, write

$$\begin{aligned}x &= \sum_{i=0}^{n} a_i b^i + \sum_{i=1}^{\infty} \alpha_i b^{-i} \\ &= a_n \ldots a_0 \cdot \alpha_1 \alpha_2 \ldots\end{aligned}$$

Algorithm (Euclid).
Euclid's algorithm can convert number x in base 10, i.e. $(x)_{10}$ into another base, b, i.e. $(x)_b$.

1. Input $(x)_{10}$
2. Determine the smallest integer n such that $x < b^{n+1}$
3. Let $y = x$. Then for $i = n, \ldots, 0$

$$\begin{aligned}a_i &= y \text{ div } b^i \\ y &= y \bmod b^i\end{aligned}$$

   which at each steps provides an a_i and updates y.
4. Output as $(x)_b = a_n a_{n-1} \cdots a_0$

Algorithm (Horner).

1. Input $(x)_{10}$
2. Set $i = 0$

3. Let $y = x$. Then while $y > 0$

$$\begin{aligned} a_i &= y \text{ div } b \\ y &= y \bmod b \\ i &= i + 1 \end{aligned}$$

which at each steps provides an a_i and updates y.

4. Output as $(x)_b = a_n a_{n-1} \cdots a_0$

Definition 3.2 (Normalized Floating Point Representations)**.** *Normalized* floating point representations with respect to some base b, store a number x as

$$x = 0 \cdot a_1 \dots a_k \times b^n$$

where the $a_i \in \{0, 1, \dots b-1\}$ are called the **digits**, k is the **precision** and n is the **exponent**. The set $a_1, \dots, a_k$ is called the **mantissa**. Impose that $a_1 \neq 0$, it makes the representation unique.

Theorem 3. Let x and y be two normalized floating point numbers with $x > y > 0$ and base $b = 2$. If there exists integers p and $q \in \mathbb{N}_0$ such that

$$2^{-p} \leq 1 - \frac{y}{x} \leq 2^{-q}$$

then, at most p and at least q significant bits (i.e. significant figures written in base 2) are lost during subtraction.

4 Linear Systems

Definition 4.1 (Systems of Linear Equations)**.** A system of linear equations (or a linear system) is a collection of one or more linear equations involving the same variables. If there are m equations with n unknown variables to solve for, i.e.

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n &= b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n &= b_2 \\ &\vdots \\ a_{m,1}x_1 + a_{m,2}x_2 + \cdots + a_{m,n}x_n &= b_m \end{aligned}$$

then the system of linear equations can be written in matrix form $Ax = b$, where

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad \text{and} \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix},$$

so that $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$.

Definition 4.2 (Banded Systems)**.** A **banded** matrix is a matrix whose non-zero entries are confined to a diagonal band, comprising the main diagonal and zero or more diagonals on either side.

Definition 4.3 (Symmetric Systems)**.** A square matrix A is **symmetric** if $A = A^T$, that is, $a_{i,j} = a_{j,i}$ for all indices i and j.

A square matrix is said to be **Hermitian** if the matrix is equal to its conjugate transpose, i.e. $a_{i,j} = \overline{a_{j,i}}$ for all indices i and j. A Hermitian matrix is written as A^H.

Definition 4.4 (Positive Definite Matrices)**.** A matrix, M, is said to be **positive definite** if it is symmetric (or Hermitian) and all its eigenvalues are real and positive.

Definition 4.5 (Nonsingular Matrices)**.** A matrix is **non-singular** or **invertible** if there exists a matrix A^{-1} such that $A^{-1}A = AA^{-1} = I$, where I is the identity matrix.

Remarks (Properties of Nonsingular Matrices)**.** For a nonsingular matrix, the following all hold:

- Nonsingular matrix has full rank
- A square matrix is nonsingular if and only if the determinant of the matrix is non-zero.
- If a matrix is singular, both versions of Gaussian elimination will fail due to division by zero, yielding a floating exception error.

Definition 4.6. If $\tilde{x}$ is an approximate solution to the linear problem $Ax = b$, then the **residual** is defined as $r = A\tilde{x} - b$.

If the magnitude of the residual, $|r|$, is large due to rounding, the matrix is said to be **ill-conditioned**.

4.1 Direct Methods

Algorithm (Gaussian Elimination)**.**
Gaussian elimination is a method to solve systems of linear equations based on forward elimination (a series of row-wise operations) to convert the matrix, A, to upper triangular form (echelon form), and then back-substitution to solve the system. The row operations are:

- row swapping
- row scaling, i.e. multiplying by a non-zero scalar
- row addition, i.e. adding a multiple of one row to another

procedure Forward Elimination
  for $k = 1$ to $n - 1$ **do**
    for $i = k + 1$ to n **do**
      for $j = k$ to n **do**
        $a_{i,j} = a_{i,j} - \frac{a_{i,k}}{a_{k,k}} a_{k,j}$
      end for
      $b_i = b_i - \frac{a_{i,k}}{a_{k,k}} b_k$
    end for
  end for
end procedure
procedure Back Substitution
  $x_n = \frac{b_n}{a_{n,n}}$
  for $i = n - 1$ to 1 **do**
    $y = b_i$
    for $j = n$ to $i + 1$ **do**
      $y = y - a_{i,j} x_j$
    end for
    $x_i = \frac{y}{a_{i,i}}$
  end for
end procedure

Algorithm (Gaussian Elimination with Scaled Partial Pivoting)**.** A pivot element is the element of a matrix which is selected first to do certain calculations. Pivoting helps reduce errors due to rounding during forward elimination.
To use partial pivoting to produce a matrix in row-echelon form

Find maximal absolute values vector s with entries $s_i = \max j = 1, \ldots, n \, |a_{i,j}|$
for $k = 1$ to $n - 1$ **do**
  for $i = k$ to n **do**
    Compute $\left| \frac{a_{i,k}}{s_i} \right|$
  end for
  Find row with largest relative pivot element, denote this as row j
  Swap rows k and j in the matrix A
  Swap entries k and j in the vector s
  Do forward elimination on row k
end for

Definition 4.7 (Upper and Lower Triangular Matrices)**.** A square matrix is said to be a **lower triangular matrix** if all the elements above the main diagonal are zero and an **upper triangular** if all the entries below the main diagonal are zero.

Theorem 4 (LU**-Decomposition).** Let $A \in \mathbb{R}^{n \times n}$ be invertible. Then there exists a decomposition of A such that $A = LU$, where L is a lower triangular matrix and U is an upper triangular matrix, And

$$L = U_1^{-1} U_2^{-1} \cdots U_{n-1}^{-1}$$

where each matrix U_i is a matrix which describes the i^{th} step in forward elimination part of Gaussian elimination

$$U = U_{n-1} \cdots U_2 U_1 A$$

Definition 4.8 (Cholesky-Decomposition)**.** A symmetric, positive definite matrix can be decomposed as $A = LL^T$.

Algorithm (Cholesky-Decomposition)**.** Given a matrix A, the lower triangular matrix L can be constructed via

for i=1 to n **do**
  for $j = 1$ to $i - 1$ **do**
    $y = a_{i,j}$
    for $k = 1$ to j-1 **do**
      $y = y - l_{i,k} l_{j,k}$
    end for
    $l_{i,j} = y / l_{j,j}$
  end for
  $y = a_{i,i}$
  for k= 1 to $i - 1$ **do**
    $y = y - l_{i,k} l_{i,k}$
  end for
  if $y \leq 0$ **then**
    there is no solution
  else
    $l_{i,i} = \sqrt{y}$
  end if
end for

4.2 Indirect Methods

For a non-singular matrix A, consider the iterative scheme $Qx_{k+1} = (Q - A)\, x_k + b$. This is equivalent to

$$x_{k+1} = \left(I - Q^{-1}A\right) x_k + Q^{-1}b.$$

Definition 4.9 (Spectral Radius of a Matrix)**.** The **spectral radius** of a matrix A is defined as

$$\rho\left(A\right) = \max\left\{\left|\lambda_1\right|, \left|\lambda_2\right|, \ldots \left|\lambda_n\right|\right\}$$

where the λ_i are the eigenvalues of the matrix.

Theorem 5 (Convergence). The iterative scheme converges if and only if the spectral radius of the matrix $I - Q^{-1}A$ is less than one, i.e. $\rho\left(I - Q^{-1}A\right) < 1$

Definition 4.10 (Richardson Iteration)**.** Let $Q = I$. Then **Richardson iteration** computes the sequence of vectors

$$x_{k+1} = (I - A)\, x_k + b$$

This may converge, depending on A.

The **modified Richardson iteration** scales $Q = \omega I$, so that

$$x_{k+1} = x_k + \omega\left(b - Ax_k\right).$$

Definition 4.11 (Jacobi Iteration)**.** The **Jacobi iteration** has $Q = D$, so

$$x_{k+1} = \left(I - D^{-1}A\right) x_k + D^{-1}b$$

Definition 4.12 (Diagonally Dominant Matrices)**.** A matrix $A \in \mathbb{R}^{n\times n}$ is said to be **diagonally dominant** if, for every row, the absolute value of the diagonal element is greater or equal to the sum of the magnitudes of all other elements, i.e.

$$\|a_{i,i}\| \geq \sum_{j=1, j\neq i}^{n} \|a_{i,j}\| \quad \text{for all } i \in (1, n)$$

Theorem 6 (Convergence of Jacobi Scheme). If a matrix A is diagonally dominant, then the Jacobi scheme converges for any initial guess x_0.

Definition 4.13 (Gauss-Seidel Scheme)**.** Let $Q = L + D$, then the **Gauss-Seidel** scheme is given by

$$(D + L)x^{(n+1)} = -Ux^{(n)} + b$$

Theorem 7 (Convergence of Gauss-Seidel). If a matrix A is diagonally dominant, then the Gauss-Seidel scheme converges for any initial guess x_0.

Definition 4.14 (Successive Over Relaxation)**.** The scheme uses $Q = L + \frac{1}{\omega}D$, thus

$$(D + \omega L)x^{(n+1)} = -\left((\omega - 1)\, D + \omega U\right) x^{(n)} + \omega b.$$

Theorem 8 (Convergence of Successive Over Relaxation). Let A be a symmetric matrix with positive entries on the diagonal and let $\omega \in (0, 2)$. Then, if and only if A is positive definite will the method of successive over relaxation converge.

5 Nonlinear Solvers

5.1 Bisection Method

Definition 5.1 (Bisection Method)**.**
The bisection method, when applied in the interval $[a, b]$ to a function $f \in C^0([a, b])$ with $f(a)f(b) < 0$

Bisect the interval into two subintervals $[a, c]$ and $[c, b]$ such that $a < c < b$.

- If $f(c) = 0$ or is sufficiently close, then c is a root
- else, if $f(c)f(a) < 0$ continue in the interval $[a, c]$
- else, if $f(c)f(b) < 0$ continue in the interval $[c, b]$

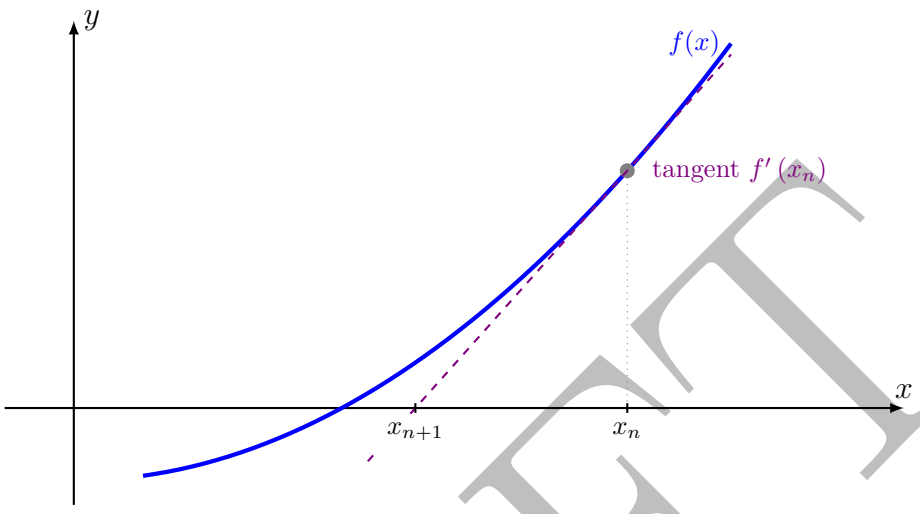
Theorem 9 (Bisection Method).
The bisection method, when applied in the interval $[a, b]$ to a function $f \in C^0([a, b])$ with $f(a)f(b) < 0$ will compute, after n steps, an approximation c_n of the root r with error

$$|r - c_n| < \frac{b - a}{2n}$$

5.2 Newton's Method

Definition 5.2.
Let a function $f \in C^1([a, b])$, then for an initial guess x_0, Newton's method is

$$x_{n+1} = x_n - \frac{f(x_n)}{f(x_n)}$$



Theorem 10.
When Newton's method converges, it converges to a root, r, of f, i.e. $f(r) = 0$.

Theorem 11.
Let $f \in C^1([a,b])$, with

1. $f(a)f(b) < 0$,
2. $f'(x) \neq 0$ for all $x \in (a,b)$,
3. $f''(x)$ exists, is continuous and either $f''(x) > 0$ or $f''(x) < 0$ for all $x \in (a,b)$.

Then $f(x) = 0$ has exactly one root, r, in the interval and the sequence generated by Newton iterations converges to the root when the initial guess is chosen according to

- if $f(a) < 0$ and $f''(a) < 0$ or $f(a) > 0$ and $f''(a) > 0$ then $x \in [a, r]$

or

- if $f(a) < 0$ and $f''(a) > 0$ or $f(a) > 0$ and $f''(a) < 0$ then $x \in [r, b]$

The iterate in the sequence satisfies

$$|x_n - r| < \frac{f(x_n)}{\min\limits_{x\in[a,b]} |f'(x)|}$$

Theorem 12.
Let $f \in C^1([a,b])$, with

1. $f(a)f(b) < 0$
2. $f'(x) \neq 0$ for all $x \in (a,b)$
3. $f''(x)$ exists and is continuous, i.e. $f(x) \in C^2([a,b])$

Then, if x_0 is close enough to the root r, Newton's method converges quadratically.

5.3 Secant Methods

Definition 5.3.
The secant method is defined as

$$x_{n+1} = x_n - f(x_n)\frac{x_{n-1} - x_n}{f(x_{n-1}) - f(x_n)}$$

Theorem 13.
Let $f \in C^2([a,b])$, and $r \in (a,b)$ such that $f(r) = 0$ and $f'(r) \neq 0$. Furthermore, let

$$x_{n+1} = x_n - f(x_n)\frac{x_{n-1} - x_n}{f(x_{n-1}) - f(x_n)}$$

Then there exists a $\delta > 0$ such that when $|r - x_0| < \delta$ and $|r - x_1| < \delta$, then the following holds:

1. $\lim_{n\to\infty} |r - x_n| = 0 \Leftrightarrow \lim_{n\to\infty} x_n = r$
2. $|r - x_{n+1}| \leq \mu |r - x_n|^\alpha$ with $\alpha = \dfrac{1+\sqrt{5}}{2}$

5.4 Convergence

Definition 5.4. If a sequence x_n converges to r as $n \to \infty$, then it is said to **converge linearly** if there exists a $\mu \in (0,1)$ such that

$$\lim_{n\to\infty} \frac{|x_{n+1} - r|}{|x_n - r|} = \mu$$

The sequences converges **super-linearly** if

$$\lim_{n\to\infty} \frac{|x_{n+1} - r|}{|x_n - r|} = 0$$

and **sub-linearly** if

$$\lim_{n\to\infty} \frac{|x_{n+1} - r|}{|x_n - r|} = 1$$

More generally, a sequence converges with order q if there exists a $\mu > 0$ such that

$$\lim_{n\to\infty} \frac{|x_{n+1} - r|}{|x_n - r|^q} = \mu$$

Thus a sequence is said to converge quadratically when $q = 2$ and exhibit cubic convergence when $q = 3$.

Method	Regularity	Proximity to r	Initial points	Function calls	Convergence
Bisection	$\mathcal{C}^0$	No	2	1	Linear
Newton	$\mathcal{C}^2$	Yes	1	2	Quadratic
Secant	$\mathcal{C}^2$	Yes	1	1	Superlinear

5.5 Systems of Nonlinear Equation

Definition 5.5 (Multi-Dimensional Newton Method)**.** For a vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^n$, which takes as an argument the vector

$$x = (x_1, x_2 \dots x_n),$$

the **Jacobian** matrix is defined as

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & & & \\ \vdots & & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

where f_1 is the first component of the vector-valued function f.

If the derivatives are evaluated at the vector x, the Jacobian matrix can be parameterised as $J(x)$. Newton's method can then be written as a vector equation,

$$x_{m+1} = x_m - J^{-1}(x_m) f(x_n)$$

where $J^{-1}(x_n)$ is the inverse of the Jacobian matrix evaluated at the m-iterate of the vector approximation vector which is denoted by x_m.

In practice, as matrix inversion can be computationally expensive, the system

$$J(x_n)(x_{n+1} - x_n) = -f(x_n)$$

is solved for the unknown vector $x_{m+1} - x_m$.

6 Interpolation

Definition 6.1. Given as set of points $p_0, \ldots, p_n \in \mathbb{R}$ and corresponding nodes $u_0, \ldots, u_n \in \mathbb{R}$, a function $f : \mathbb{R} \to \mathbb{R}$ with $f(u_i) = p_i$ is an **interpolating function**.

This can be generalised to higher dimensions, i.e. $f : \mathbb{R} \to \mathbb{R}^N$.

Definition 6.2. If the interpolating function is a polynomial, it can be written as

$$p(u) = \sum_{i=0}^{n} \alpha_i \varphi_i(u).$$

So that for every j, the polynomial satisfies $p(u_j) = \sum_{i=0}^{n} \alpha_i \varphi_i(u_j)$, for weights α_j. Thus, solving for values of α which fit the interpolating function to the data, leads to a linear system of the form

$$\Phi\alpha = p$$

where p is the vector defined the polynomial evaluated at the node points, i.e. $p = p(u_j)$ and Φ is the **collocation matrix**, given by

$$\Phi = \begin{pmatrix} \varphi_0(u_0) & \varphi_1(u_1) & \cdots & \varphi_n(u_n) \\ \vdots & & & \vdots \\ \varphi_0(u_n) & \cdots & \cdots & \varphi_n(u_n) \end{pmatrix}$$

Thus $\alpha = \Phi^{-1} p$.

The collocation matrix is invertible if and only if the set of functions φ are linearly independent.

Definition 6.3. If

$$p(u) = \sum_{i=0}^{n} \alpha_i \varphi_i(u)$$

So that for every j, $p(u_j) = \sum_{i=0}^{n} \alpha_i \varphi_i(u_j)$, thus the α_i lead to a linear system of the form

$$\Phi\alpha = p$$

where Φ is the **Vandermonde matrix**.

Definition 6.4 (Lagrange Polynomials). The **Lagrange form of an interpolating polynomial** is given by

$$p(x) = \sum_{i=0}^{n} \alpha_i l_i(x)$$

where $l_i \in \mathbb{P}\_n$ are such that $l_i(x_j) = \delta_{ij}$. The polynomials $l_i(x) \in \mathbb{P}\_n$ for $i = 0, \ldots, n$, are called **characteristic polynomials** and are given by

$$l_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}.$$

Algorithm (Aitken's Algorithm).
Aitken's algorithm is an iterative process for evaluating Lagrange interpolation polynomials at an arbitrary point, u^*, without explicitly constructing them. If the interpolating polynomial is given by p, and is derived from n data points (u_i, y_i) for $i = 0, \ldots, n$

$$p(u) = \sum_{i=0}^{n} p_i^n l_i^n(u)$$

The interpolation is achieved by constructing a series of polynomials, evaluated at the $u = u^*$, where $p_i^k(u)$ is given by

$$p_i^{k+1}(u) = p_i^k(u)\left(\frac{u - u_{n-k}}{u_i - u_{n-k}}\right) + p_{n-k}^k(u)\left(1 - \frac{u - u_{n-k}}{u_i - u_{n-k}}\right)$$

with initial values $p_i^0 = y_i$.

$$\begin{array}{lll} p(u_0) = p_0^0 & & \\ & p_0^1(u) & \\ p(u_1) = p_1^0 & & p_0^2(u) \\ & p_1^1(u) & & p_0^3(u) \\ p(u_2) = p_2^0 & & p_1^2(u) \\ & p_2^1(u) & \\ p(u_4) = p_3^0 & & \end{array}$$

where the coefficients are evaluated from left to right.

6.1 Piecewise Polynomial Interpolation

Definition 6.5. A function $s(u)$ is called a **spline** of degree k on the domain $[a,b]$ if $s \in C^{k-1}([a,b])$ and there exists nodes $a = u_0 < u_1 < \ldots < u_m = b$ such that s is a polynomial of degree k for $i = 0, \ldots m-1$.

Definition 6.6 (B-Splines)**.** A spline is said to be a **b-spline** if it is of the form

$$s(u) = \sum_{i=0}^{m} \alpha_i \mathcal{N}\_i^n(u)$$

where $\mathcal{N}^n$ are the **basis spline functions** of degree n with minimal support. (That is they are positive in the domain and zero outside). The functions are defined recursively. Let u_i be the set of nodes $u_0, u_1, \ldots, u_m$, then

$$\mathcal{N}\_i^0(u) = \begin{cases} 1 & \text{for} \quad u_i \le u \le u_{i+1} \\ 0 & \text{else.} \end{cases}$$

and

$$\mathcal{N}\_i^n(u) = \alpha_i^{n-1}(u)\mathcal{N}\_i^{n-1}(u) + \left(1 - \alpha_{i+1}^{n-1}(u)\right)\mathcal{N}\_i + 1^{n-1}(u)$$

where

$$\alpha_i^{n-1}(u) = \frac{u - u_i}{u_{i+n} - u_i}$$

is a local parameter.

Given data with nodes u_i and values p_i, to interpolate with splines, of order n, requires solving

$$\text{Find} \quad s = \sum_{i=0}^{m} \alpha_i \mathcal{N}_i^n(u) \quad \text{such that} \quad s(u_i) = p_i \quad \text{for} \quad i = 0, \ldots, m$$

which is matrix form is $\Phi\alpha = p$, where the collocation matrix, $\Phi \in \mathbb{R}^{(m+1)\times(m+1)}$ is given by

$$\Phi = \begin{pmatrix} \mathcal{N}\_0^n(u_0) & \cdots & \mathcal{N}\_m^n(u_0) \\ \vdots & & \vdots \\ \mathcal{N}\_0^n(u_m) & \cdots & \mathcal{N}\_m^n(u_m) \end{pmatrix}$$

6.2 Least-Squares Approximation

Definition 6.7 (Least-Squares Approximation)**.** Given a set of points $y = (y_0, y_1, \dots y_n)$ at nodes x_i, seek a continuous function of x, with a given form characterized by m parameters $\beta = (\beta_0, \beta_1, \dots, \beta_m)$, i.e. $f(x, \beta)$, which approximates the points while minimizing the error, defined by the sum of the squares

$$E = \sum_{i=0}^{n} (y - f(x_i, \beta))^2 .$$

The minimum is found when

$$\frac{\partial E}{\partial \beta_j} = 0 \quad \text{for all} \quad j = 1, \dots m$$

i.e.

$$-2 \sum_{i=0}^{n} (y_i - f(x_i, \beta_j)) \frac{\partial f(x_i, \beta)}{\partial \beta_j} = 0 \quad \text{for all} \quad j = 1, \dots m.$$

Definition 6.8 (Linear Least-Squares Approximation)**.** If the function f is a function of the form

$$y = \sum_{j=1}^{m} \beta_j \varphi_j(x)$$

then the least squares problem can be expressed as

$$\frac{\partial E}{\partial \beta_j} = \sum_{j=1}^{m} \left(\sum_{i=1}^{n} \varphi_j(x_i) \varphi_k(x_i) \right).$$

Thus, the weights β can be determined by solving the linear system,

$$\Phi \Phi^T \beta = \Phi y,$$

i.e. $\beta = \left(\Phi \Phi^T\right)^{-1} \Phi y$, where Φ is the collocation matrix.

7 Numerical Differentiation

Definition 7.1 (Finite-Difference Quotients)**.** Consider the approximations to the first-order derivative:

1. **Forward Difference Quotient:**

$$D_j^+ u = \frac{u_{j+1} - u_j}{h}$$

2. **Backwards Difference Quotient:**

$$D_j^- u = \frac{u_j - u_{j-1}}{h}$$

3. **Central Difference Quotient:**

$$D_j^0 u = \frac{u_{j+1} - u_{j-1}}{2h}$$

The forward and backwards difference schemes are first order approximations to the derivative. The central difference scheme is a second order accurate approximation.

Definition 7.2 (Richardson Extrapolation)**.** This is a method for deriving higher order approximations for derivatives from lower order approximations. Consider a first-order approximation to the derivative, $\varphi(h)$, such as backwards or forwards differencing, then

$$f'(x) = \varphi(h) + a_2 h^2 + a_3 h^3 + \dots$$

Now evaluate the derivative at $h = h/2$, so that

$$f'(x) = \varphi\left(\frac{h}{2}\right) + a_2 \left(\frac{h}{2}\right)^2 + a_3 \left(\frac{h}{2}\right)^3 + \dots$$

Combining the two terms so that the low order term cancel, i.e. via $f'(x) - 4f'(x)$, then a better approximation can be found as

$$f'(x) = \varphi(h) - 4\varphi\left(\frac{h}{2}\right) + \mathcal{O}\left(h^3\right).$$

The process can also be applied to second order accurate schemes, such as central differencing, to produce more accurate approximations to higher order derivatives.

Definition 7.3 (Higher Order Derivatives). with $f(x+h)$ and $f(x-h)$, so

$$f''(x) = \frac{f(x-h) - 2f(x) + f(x+h)}{h^2} + \mathcal{O}\left(h^2\right)$$

8 Numerical Integration

Definition 8.1 (Riemann Sum). Create a partition, p, of the domain of integration: define nodes $a = x_0 < x_1 < \ldots < x_n = b$, so that for $i = 0, 1, \ldots n-1$, there are sub-intervals $[x_i, x_{i+1}]$. Then approximate the area under the curve by summing the areas in each subinterval defined as

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=0}^{n-1} (x_{i+1} - x_i)\, f(x^*) \quad x^* \in [x_i, x_{i+1}]\,.$$

If f is continuous, the value of x_i^* may be chosen arbitrarily in the interval $[x_i, x_{i+1}]$. Then the **Lower** and **Upper** sums are given by

$$L(f,p) = \sum_{i=0}^{n-1} (x_{i+1} - x_i)\, m_i \quad \text{where} \quad m_i = \min_{x \in [x_i, x_{i+1}]} f(x)$$

$$U(f,p) = \sum_{i=0}^{n-1} (x_{i+1} - x_i)\, M_i \quad \text{where} \quad M_i = \max_{x \in [x_i, x_{i+1}]} f(x)$$

so that

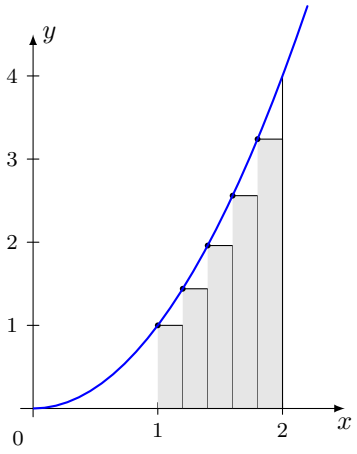
$$L(f,p) \leq \int_a^b f(x)\,,\mathrm{d}x \leq U(f,p)\,.$$

Definition 8.2 (Trapezoidal Rule). Rather than rectangles, use trapezoids to approximate the integral in a sub domain

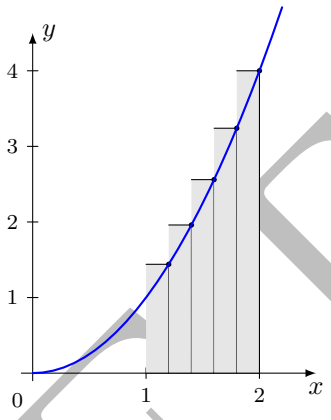
$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=0}^{n-1} (x_{i+1} - x_i)\, \frac{f(x_i) + f(x_{i+1})}{2}.$$

If the nodes of the partition are equally spaced, so that $h = x_{i+1} - x_i$, then the formula is given by

$$\frac{h}{2}\left(f(x_0) + f(x_n)\right) + h \sum_{i=1}^{n-2} f(x_i)\,.$$



(a) Lower Riemann Sum



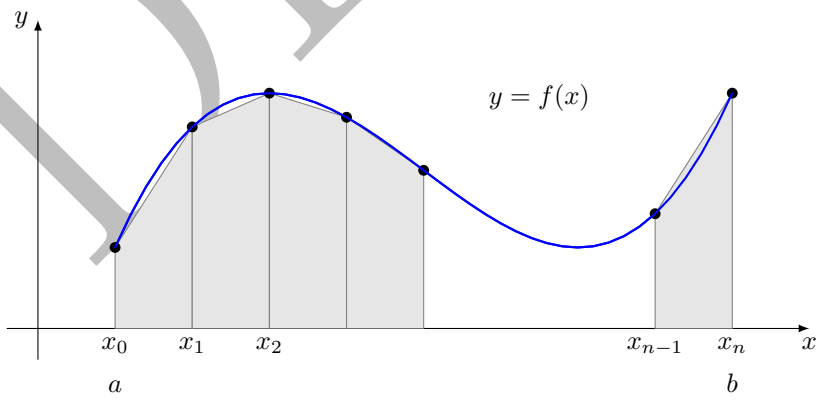
(b) Upper Riemann Sum

Figure 1: Riemann Sums

Theorem 14 (Error for Trapezoidal Rule). Let $f \in C^2([a,b])$ and p be equidistant partition of $[a,b]$, with $h = x_{i+1} - x_i$. Then the error for the trapezium rule is

$$\left| \int_a^b f(x)\, \mathrm{d}x - T(f,p) \right| = \frac{1}{12} \left| (b-a)\, h^2 f''(\xi) \right|$$

for a $\xi \in (a,b)$.



Definition 8.3 (Simpson's Rule)**.** The integral is approximated as

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right).$$

Simpson's rule uses quadratic interpolation. It can be applied in a composite manner, i.e. on many subdomains. It has an asymptotic error of $\mathcal{O}\left(h^4\right)$.

Algorithm (Romberg Algorithm)**.** Romberg's method uses the Trapezoidal Rule and then Richardson Extrapolation to estimate integrals. First consider a sequence of partitions, p_i, of equal spacing given by $h_i = \dfrac{b-a}{2^i}$ for $i = 0, \ldots, n$, which yield a sequence of integrals $R_i^0 = T_i(f, p_i)$. Refinements of the integrals can then be produced by Richardson Extrapolation

$$\begin{array}{ll} R_0^0 & \\ & R_1^1 \\ R_1^0 & \\ & R_2^1 \\ R_2^0 & \\ \vdots & \\ R_n^0 & \end{array}$$

Thus, consider the two integrals

$$\int_a^b f(x)\,\mathrm{d}x = R_{i-1}^0 + a_2h^2 + a_4h^4 + \ldots$$

$$\int_a^b f(x)\,\mathrm{d}x = R_i^0 + a_2\left(\frac{h}{2}\right)^2 + a_4\left(\frac{h}{2}\right)^4 + \ldots$$

Note that there no odd terms in the error. Then,

$$R_i^1 = \frac{1}{3}\left(4R_i^0 - R_{i-1}^0\right).$$

which has an error $\mathcal{O}\left(h^4\right)$. The extrapolated values are equivalent to integrals approximated by Simpson's rule. The recurrence formula can be derived

$$R_i^m = \frac{1}{4^m - 1}\left(4^m R_i^{m-1} - R_{i-1}^{m-1}\right).$$

8.1 Gauss Quadrature

Generalise the quadrature formula so that an integral is approximated as

$$I_n(f) = \sum_{i=0}^{n} \alpha_i f(x_i)$$

The above equation is a weighted sum of the values of f at the points x_i, for $i = 0, \ldots, n$. These points are said to be the *nodes* of the quadrature formula, while the $\alpha_i \in \mathbb{R}$ are its *coefficients* or *weights*. Both weights and nodes depend in general on n.

- Can the weights be chosen such that the error in an integral is minimized?
- Furthermore, can the nodes be chosen such that the integral can be improved?

Definition 8.4 (Orthogonal functions)**.** Two real-valued functions $f(x)$ and $g(x)$ are said to be **orthogonal** if

$$\langle f, g \rangle = \int_a^b f(x)g(x), \mathrm{d}x = 0.$$

Theorem 15 (Gaussian Quadrature). Let $q(x)$ be a non-trivial polynomial of degree $n+1$ such that

1. it has $n+1$ distinct roots, denoted as x_i, in $[a, b]$,
2. the polynomial satisfies

$$\int_a^b x^k q(x) \, \mathrm{d}x = 0 \quad \text{for} \quad k = 0, \ldots, n.$$

   i.e. is orthogonal to x^k.

Then, denote the integral as

$$I[f] = \int_a^b f(x) \, \mathrm{d}x = \sum_{i=0}^{n} A_i f(x_i)$$

with $A_i = \int_a^b L_i(x) \, \mathrm{d}x$ for all polynomials $f(x)$ of degree less than or equal to $2n+1$. The integral $I[f]$ integrates all polynomials of degree $2n+1$ *exactly.*
The degree of exactness of $I[f]$ is $2n+1$.

Definition 8.5 (Gauss-Legendre Quadrature)**.** The Legendre polynomials are a set of orthogonal polynomials where

$$\int_{-1}^{1} P_m(x)P_n(x)\,\mathrm{d}x = 0 \quad \text{for} \quad n \neq m.$$

and $P_0 = 1$. Thus, $P_1 = x$, $P_2 = \left(3x^2 - 1\right)/2$. The Legendre polynomials obey a recursive formula:

$$P_n = \frac{2n-1}{n} x P_{n-1}(x) - \frac{n-1}{n} P_{n-2}(x), \quad \text{for} \quad n \geq 2.$$

Gauss-Legendre quadrature uses the roots of the Legendre polynomials as the nodes for integration, and weights found by equating the quadrature expressions with the exact integrals for $f = 1, x, x^2$.
The domain of integration can be scaled via the invertible transformation $x = \frac{b-a}{2}t + \frac{a+b}{2}$, so that

$$\int_a^b f(x)\,\mathrm{d}x = \frac{b-a}{2} \int_{-1}^{1} f\left(\frac{b-a}{2}t + \frac{a+b}{2}\right)\,\mathrm{d}t.$$