Constructor University Bremen Spring Semester 2025/6

Dr. D. Sinden

CTMS-MAT-13: Numerical Methods

Summary from 7 October 2025

Notes for numerical methods course 2026. This can be downloaded from:
`https://djps.github.io/docs/numericalmethods/notes`

[7 October 2025 – Version 0.1.13]

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 $f(x)$ is given by the power series

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

where $f^{(k)} = \dfrac{\mathrm{d}^k f}{\mathrm{d}x^k}$ is the k^{th} derivative.

The idea behind a Taylor series is that a function $f(x)$ can be approximated around some central point c by a polynomial (or, if the function is infinitely-differentiable, a power series).

This is a power series, which is convergent for some radius. The series can be truncated upto n terms.

Theorem 1 (Taylor's Theorem).
For a function $f(x) \in C^{n+1}([a,b])$, i.e. f is $(n+1)$-times continuously differentiable in the interval $[a,b]$, then for some expansion point 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 ξ which depends on x, and exist on $\xi(x) \in [c,x]$. The remainder term

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

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 errors* 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 $\tilde{x}_i$.

3 Number Representations

Definition 3.1 (Base Representation)
Every number $x \in \mathbb{N}_0$ can be written as a unique expansion with respect to base $b \in \mathbb{N}\backslash\{1\}$ 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 $a_i \in \mathbb{N}_0$ 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 an integer 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 $i = n, \ldots, 0$

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

   which at each steps provides an a_i and updates y.

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

where div is integer division, and mod is the remainder operator.

There are two issues: finding n maybe difficult and for large values of b^i division maybe computationally costly. Horner's algorithm seeks to overcome these issues.

Algorithm (Horner)
Horner's algorithm is:

1. Input $(x)_{10}$
2. Set $i = 0$
3. Let $y = x$. Then while $y > 0$

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

   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

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

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**.

Imposing that $a_1 \neq 0$, it makes the representation unique, otherwise b^n can take multiple possible values.

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 $A\boldsymbol{x} = \boldsymbol{b}$, where

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

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

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 & Hermitian 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.

An equivalent definition is that for any non-zero real column vector $\boldsymbol{z}$, then $\boldsymbol{z}^T M \boldsymbol{z}$ is always 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 (i.e. with and without pivoting) will fail due to division by zero, yielding a floating exception error.

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

The residual is $|\boldsymbol{r}|$, formed by taking the norm of the residual vectors, and is a scalar quantity.

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

1: Find maximal absolute values vector $\vec{s}$ with entries

$$s_i = \max_{j=1,\ldots,n} |a_{i,j}|$$

2: **for** $k = 1$ to $n - 1$ **do**
3: **for** $i = k$ to n **do**
4: Compute relative pivot elements $\left|\frac{a_{i,k}}{s_i}\right|$
5: **end for**
6: Find row with largest relative pivot element, denote this as row j
7: Swap rows k and j in the matrix A
8: Swap entries k and j in the vector $\vec{s}$
9: Do forward elimination on row k
10: **end for**

It is important to remember that while the elements of the augmented matrix are rescaled during forward elimination, the elements of the vector $\vec{s}$ are reordered.

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.

If all of the entries on the main diagonal of a lower triangular matrix are also zero, the matrix is called **strictly lower triangular**, and similarly, if all of the entries on the main diagonal of a upper triangular matrix are zero, the matrix is called **strictly upper triangular**.

Note that

- The product of two upper triangular matrices is also upper triangular.
- The inverse of an upper triangular matrix is also upper triangular.

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.

A single pass of Gaussian forward elimination can be written as a matrix operation, where the elementary row operations are expressed as a Gaussian transformation matrix, M. This is a lower triangular matrix. It can be shown that

$$L = M_1^{-1} M_2^{-1} \cdots M_{n-1}^{-1}$$

where each matrix M_i is a matrix which describes the i^{th} pass of forward elimination part of Gaussian elimination. Equivalently, the upper triangular matrix U is given by

$$U = M_{n-1} \cdots M_2 M_1 A.$$

Definition 4.8 (Cholesky-Decomposition)
A symmetric, positive definite matrix can be decomposed as $A = \tilde{L}\tilde{L}^T$, where $\tilde{L} = LD^{1/2}$, where D is a diagonal matrix whose elements d_i are all positive, so that $D^{1/2}$ has elements $\sqrt{d_i}$.

Algorithm (Cholesky-Decomposition)
Given a matrix A, the lower triangular matrix $\tilde{L}$ 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**
    A is not positive definite, so factorisation fails
  else
    $l_{i,i} = \sqrt{y}$
  end if
end for

4.2 Indirect Methods

For a non-singular matrix A, consider an iterative scheme of the form

$$Q\vec{x}_{k+1} = (Q - A)\,\vec{x}_k + \vec{b}.$$

This is equivalent to

$$\vec{x}_{k+1} = \left(I - Q^{-1}A\right)\vec{x}_k + Q^{-1}\vec{b}.$$

Definition 4.9 (Convergent Methods)
A numerical method is said to be **convergent** if $x_k \to x^*$ as $k \to \infty$ where x^* is the exact solution.

Definition 4.10 (Spectral Radius of a Matrix)
The **spectral radius** of a square $n \times n$ matrix A is defined as

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

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.

$$\rho\left(I - Q^{-1}A\right) < 1.$$

Corollary 4.11
A sequence of vectors $\vec{e}_k = M^k\vec{e}_0$ converges to the zero vector if and only if the **spectral radius** of the matrix M is less than one.

Definition 4.12 (Order of Convergence)
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 $q = 3$.

Definition 4.13 (Richardson Iteration)
Let $Q = I$. Then **Richardson iteration** computes the sequence of vectors

$$\vec{x}_{k+1} = (I - A)\,\vec{x}_k + \vec{b}.$$

This may converge, depending on A.

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

$$\vec{x}_{k+1} = \vec{x}_k + \omega\left(\vec{b} - A\vec{x}_k\right).$$

Definition 4.14 (Jacobi Iteration)
The **Jacobi iteration** has $Q = D$, where D is the diagonal matrix of A, so

$$\vec{x}_{k+1} = \left(I - D^{-1}A\right)\vec{x}_k + D^{-1}\vec{b}.$$

Definition 4.15 (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.

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

If the absolute value of the diagonal element is greater than the sum of the magnitudes of all other elements, then the matrix is said to be **strictly diagonally dominant**

Theorem 6 (Convergence of Jacobi Scheme).
If a matrix A is strictly diagonally dominant, then the Jacobi scheme converges for any initial guess $\vec{x}_0$.

Definition 4.16 (Gauss-Seidel Scheme)
Let $Q = L + D$, where L is the lower triangular matrix of A and D is the diagonal matrix of A, then the **Gauss-Seidel** scheme is given by

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

Theorem 7 (Convergence of Gauss-Seidel).
If a matrix A is strictly diagonally dominant, then the Gauss-Seidel scheme converges for any initial guess $\vec{x}_0$.

Definition 4.17 (Successive Over Relaxation)
The scheme uses $Q = L + \frac{1}{\omega}D$, where L is the lower triangular matrix of A and D is the diagonal matrix of A, thus

$$(D + \omega L)\vec{x}_{k+1} = -\left((\omega - 1)\, D + \omega U\right)\vec{x}_k + \omega\vec{b}.$$

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.

4.3 Fundamental Theorem of Numerical Analysis

Definition 4.18 (Stability)
A numerical method is said to be **stable** if and only if any initial error $\vec{e}_0$ is damped during the iterations, i.e. $\|\vec{e}_k\| < \|\vec{e}_0\|$.

Remark
Note that $\|\vec{x}\|$ is a *norm* of a vector $\vec{x} \in \mathbb{R}^n$, for example one such norm is the L^2 (or Euclidean) norm, given by $\|\vec{x}\|_2 = \sqrt{x_0^2 + x_1^2 + \dots x_n^2}$.

Definition 4.19 (Consistency)
A numerical method is said to be **consistent** if any fixed point $\boldsymbol{x}^*$ of the iteration is a solution to the problem being solved.

That is the consistency error $\tau_k = \left\|A\boldsymbol{x}^k - \boldsymbol{b}\right\|$ tends to zero as $k \to \infty$ and $\boldsymbol{x}^k \to \boldsymbol{x}^*$

For a linear system, a fixed point, $\boldsymbol{x}^*$, fulfils

$$\boldsymbol{x}^* = \left(I - Q^{-1}A\right)\boldsymbol{x}^* + Q^{-1}\boldsymbol{b} \Leftrightarrow A\boldsymbol{x}^* = \boldsymbol{b}.$$

Thus, a fixed point of the iterative scheme is a solution of the linear system.

If the method is stable then

$$\vec{e}_k = \left(I - Q^{-1}A\right)^k \vec{e}_0,$$

so then $\left\|I - Q^{-1}A\right\| < 1$ for $\|\vec{e}_k\| < \|\vec{e}_0\|$. Note that $\left\|I - Q^{-1}A\right\|$ is the norm of a matrix, which is induced by a vector norm.

Theorem 9 (Fundamental Theorem of Numerical Analysis).
A numerical method is convergent if and only if it is consistent and stable.

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 $[c, b]$.

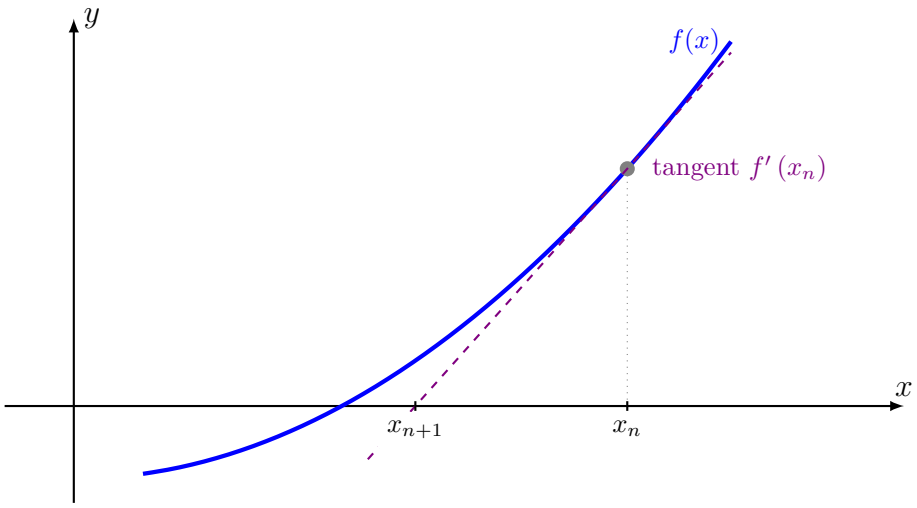
Theorem 10.
The bisection method, when applied in the interval $[a, b]$ to a function $f \in C^0([a, b])$ with $f(a)f(b) < 0$ will compute, after n steps, an approximation c_n of the root r with error

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

5.2 Newton's Method

Definition 5.2 (Newton's Method)
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 11.
When Newton's method converges, it converges to a root, r, of $f(x)$, i.e. $f(r) = 0$.

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, 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 (a,b) 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 iterates in the sequence will satisfy

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

Theorem 13.
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. $f(x) \in C^2([a,b])$.

If x_0 is close enough to the root r, then Newton's method converges quadratically:

$$|r - x_{n+1}| \leq \frac{M}{2m} |r - x_n|^2$$

where $M = |\max_{\xi_n} f''(\xi_n)|$ and $m = |\min x_n f'(x_n)|$

5.3 Secant Method

Definition 5.3 (Secant Method)
The secant method is defined as

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

where the derivative of f is approximated via a Taylor expansion.

Theorem 14.
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 $\alpha = \frac{1+\sqrt{5}}{2}$.

The properties of the three nonlinear solvers can be summarised as follows:

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

5.4 Systems of Nonlinear Equation

Definition 5.4 (Multi-Dimensional Newton Method)
For a vector-valued function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$, which takes as an argument the vector

$$\boldsymbol{x} = (x_1, x_2 \dots x_n) \in \mathbb{R}^n,$$

and returns a vector in $\boldsymbol{f}(x) \in \mathbb{R}^m$, the **Jacobian** matrix is defined as

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & & & \vdots \\ \vdots & & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix} \in \mathbb{R}^{m \times n}$$

where f_1 is the first component of the vector-valued function $\boldsymbol{f}$.

If the derivatives are evaluated at the vector $\boldsymbol{x}$, the Jacobian matrix can be parameterised as $J(\boldsymbol{x})$.

If $m = n$, so that the matrix is square, then Newton's method can then be written as a vector equation,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - J^{-1}(\boldsymbol{x}_k)\, \boldsymbol{f}(\boldsymbol{x}_k)$$

where $J^{-1}(\boldsymbol{x}_k)$ is the inverse of the Jacobian matrix evaluated at the k-iterate of the approximation vector which is denoted by $\boldsymbol{x}_k$.

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

$$J(\boldsymbol{x}_k)(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) = -\boldsymbol{f}(\boldsymbol{x}_k)$$

is solved for the unknown vector $\boldsymbol{x}_{k+1} - \boldsymbol{x}_k$, and then $\boldsymbol{x}_{k+1}$ is found.

6 Interpolation

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

Definition 6.2 (Collocation)
If the interpolating function is a polynomial, it can be written as

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

where $\varphi_i(u)$ are polynomials. Thus for every nodal value j, the polynomial exactly satisfies

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

for the weights α_i. Thus, solving for all the 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**. If there are $n+1$ data points, the collocation matrix is given by

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

i.e. elements of the matrix are given by $\Phi_{i,j} = \varphi_j(u_i)$.

Thus the weights which define the interpolating polynomial are found as $\alpha = \Phi^{-1}p$, if the matrix is square.

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

When the polynomials are given by $\varphi_i(u) = u^i$, so that the collocation matrix has elements $\Phi_{i,j} = u_i^j$ then Φ is called the **Vandermonde matrix**.

Definition 6.3 (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

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

Definition 6.4 (Newton Interpolation)
Newton interpolation interpolates a set of points (x_i, y_i) as

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

using a linear combination of Newton basis polynomials, which are defined as

$$\begin{aligned} n_0(x) = 1, \quad n_i(x) &= (x - x_0)(x - x_1) \cdots (x - x_{i-1}) \\ &= \prod_{j=0}^{j-1} (x - x_j). \end{aligned}$$

By construction, $\alpha_0 = y_0$, and subsequent terms must be solved by evaluating the interpolating polynomial at increasing orders, leading to the formula

$$\alpha_{i+1} = \frac{y_{i+1} - p_i(x_{i+1})}{n_i(x_{i+1})}.$$

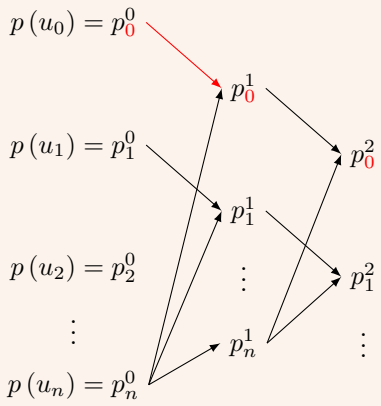
Algorithm (Aitken's Algorithm)
Aitken's algorithm is an iterative process for evaluating Lagrange interpolation polynomials at an arbitrary point, u^*, without explicitly constructing them. If the interpolating polynomial is given by p, and is derived from n data points (u_i, y_i) for $i = 0, \ldots,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

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

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 $p_i^0 = y_i$, i.e.



where the coefficients are evaluated from left to right, until p_0^n is evaluated.

6.1 Piecewise Polynomial Interpolation

Definition 6.5 (Spline Functions)
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

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

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 $y(x)$ is a function of the form

$$y = \sum_{j=0}^{m} \beta_j \varphi_j(x)$$

where φ_j are linearly independent, i.e. there are no non-zero constants c_1 and c_2 such that $c_1\varphi_i + c_2\varphi_j = 0$ for all x. Then, for $n+1$ data points, the least squares problem can be expressed as

$$E(\beta) = \sum_{i=0}^{n} \left(y_i - \sum_{j=0}^{m} \beta_j \varphi_j(x_i) \right)^2.$$

This has a minimum when

$$\frac{\partial E}{\partial \beta_j} = \sum_{i=0}^{n} \left(y_i - \sum_{k=0}^{m} \beta_k \varphi_k(x_i) \right) \varphi_j(x_i) = 0.$$

Thus, the weights β_j, for $j = 0, \ldots, m$, can be determined by solving the **normal equations**, that is, from the linear equation

$$\vec{y} = \Phi\vec{\beta},$$

where $\vec{\beta} \in \mathbb{R}^{m+1}$, $\vec{y} \in \mathbb{R}^{n+1}$ where $\Phi \in \mathbb{R}^{(n+1)\times(m+1)}$ is the **collocation matrix**. Equivalently,

$$\Phi^T\Phi\vec{\beta} = \Phi^T\vec{y},$$

i.e. it is now possible to solve the linear equation through finding the inverse of a matrix

$$\vec{\beta} = \left(\Phi^T\Phi\right)^{-1}\Phi^T\vec{y},$$

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, i.e. $D^+ u = f(x) + \mathcal{O}(h)$. 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

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

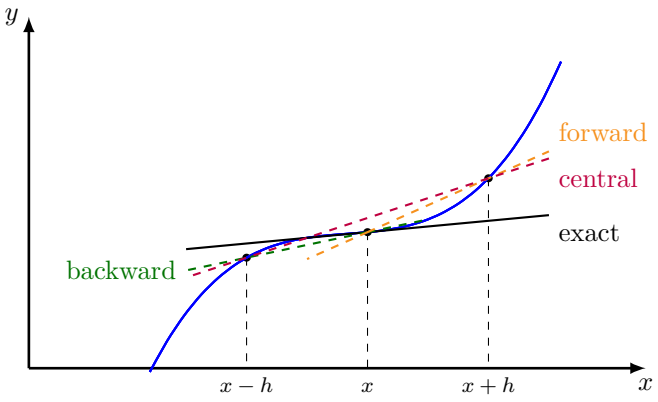
Combining the two terms so that the low order term cancel, i.e. via $f'(x) - 4f'(x)$, then a better approximation can be found as

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

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

Definition 7.3 (Higher Order Derivatives)
From the Taylor expansion of $f(x+h)$ and $f(x-h)$, the second order derivative can be expressed as

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



8 Numerical Integration

Definition 8.1 (Riemann Sum)
Create a partition, p, of the domain of integration: define $n+1$ nodes $a = x_0 < x_1 < \ldots < x_n = b$, so that there are n 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 Riemann 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 bounds for the value of the quadrature can be made

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

Additionally, the **left** and **right Riemann sums** are given by

$$\sum_{i=0}^{n-1} (x_{i+1} - x_i)\, f(x_{i-1}),$$

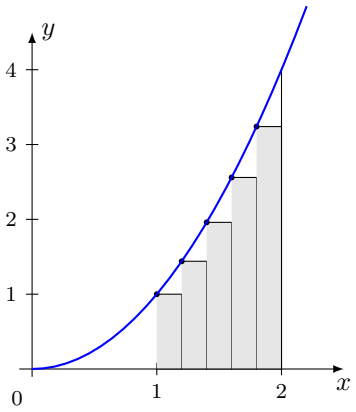
$$\sum_{i=0}^{n-1} (x_{i+1} - x_i)\, f(x_i).$$

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

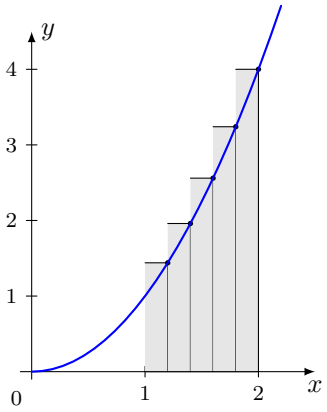
$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=0}^{n-1} (x_{i+1} - x_i)\,\frac{f(x_i) + f(x_{i+1})}{2}.$$

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

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



(a) Lower Riemann Sum



(b) Upper Riemann Sum

Theorem 15 (Error for Trapezoidal Rule).
Let $f \in C^2([a,b])$ and p be equidistant partition of $[a,b]$, with $h = x_{i+1} - x_i$. The error can be shown to have the form:

$$\left| \int_a^b f(x)\, \mathrm{d}x - T(f,p) \right| = a_2 h^2 + a_4 h^4 + \dots$$

that is, the error terms are even powers of the discretization. The error for the trapezium rule is

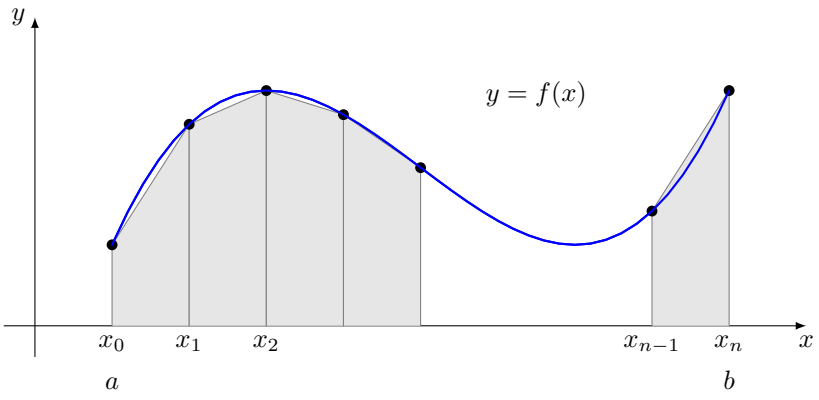
$$\left| \int_a^b f(x)\, \mathrm{d}x - T(f,p) \right| = \frac{1}{12} \left| (b-a)\, h^2 f''(\xi) \right|$$

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

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

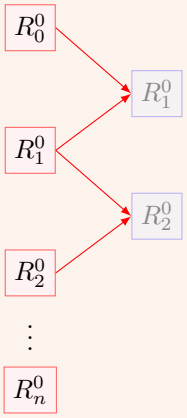
$$\int_a^b f(x)\, \mathrm{d}x \approx \frac{b-a}{6} \left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right).$$

Simpson's rule uses interpolation with quadratic polynomials. 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 $R_i^0 = T_i\left(f, p_i\right)$. Refinements of the integrals can then be produced by Richardson Extrapolation.



Thus, consider the two integrals

$$\int_a^b f\left(x\right)\,\mathrm{d}x = R_{i-1}^0 + a_2 h^2 + a_4 h^4 + \ldots$$

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

Note that there no odd terms in the error. Then, define the next set of refinements as

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

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 16 (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 so that $\langle x^k, q \rangle = 0$.

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*. Thus, 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

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

where m and n are the order of the polynomials, and $P_0 = 1$. Thus, $P_1 = x$, $P_2 = \left(3x^2 - 1\right)/2$ *etc.* 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 $f(x) = 1, x, x^2, \ldots, n$.

The domain of integration can be scaled via the invertible transformation $x = \frac{b-a}{2}t + \frac{a+b}{2}$, so that

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

9 Differential Equations

9.1 Finite Difference Methods for Differential Equations

Solutions to differential equations are functions.

Definition 9.1 (Ordinary Differential Equations)
An **ordinary differential equation** (ODE) is an equation that involves one or more derivatives of a function of a single variable.

For example, with only the first derivative $y'(t) = f(y(t), t)$.

Definition 9.2 (Initial Value Problems)
An **initial value problem** (IVP) is given by an ordinary differential equation of the form $y'(t) = f(y(t), t)$ and initial value $y(a) = y_a$ for the unknown function $y(t)$, with $t \geq a$.

Often $a = 0$, and the initial condition is denoted by $y(0) = y_0$.

Definition 9.3 (One-step methods)
A numerical method for approximating the solution to a differential equation is called a **one-step method** if the solution at time step t_{n+1}, denoted by u_{n+1}, depends only on the previous one, t_n, where $t_{n+1} = t_n + h$, for some small increment $h = \Delta t$.

Definition 9.4 (Forward Euler)
This approximates the derivative through a first-order forward difference approximation of the first-order derivative, i.e. for u_n, the solution u at t_n, the computed solution to a differential equation $\dot{u} = f(u)$, evolves according to

$$u_{n+1} = u_n + hf_n$$

where $f_n = f(u_n, t_n)$ and $h = t_{n+1} - t_n$. The local truncation error is $\mathcal{O}(h^2)$.

Definition 9.5 (Backward Euler Method)
This method uses the **backward finite difference** approximation of the first-order derivative, so that the solution is computed via

$$u_{n+1} = u_n + hf_{n+1}$$

where $f_{n+1} = f(u_{n+1}, t_{n+1})$.

Definition 9.6 (Crank-Nicolson Method)
This method is given by

$$u_{n+1} = u_n + \frac{h}{2}\left(f_n + f_{n+1}\right).$$

Definition 9.7 (Heun's Method)
This method is given by

$$u_{n+1} = u_n + \frac{h}{2}\left(f_n + f\left(u_n + hf_n, t_{n+1}\right)\right).$$

Alternatively, one-step methods can be considered to be integrators

$$y\left(t+h\right) = y\left(t\right) + \int_t^{t+h} f\left(y\left(\tau\right), \tau\right)\, \mathrm{d}\tau.$$

Thus, the forward Euler method is equivalent to the **left Riemann sum**, backward Euler is equivalent to the **right Riemann sum** and the Crank-Nicolson is the **trapezoidal rule**.

Definition 9.8 (Implicit and Explicit Schemes)
A numerical method is said to be **explicit** if an approximation u_{n+1} can be calculated directly from already computed values u_i, $i < n$. Otherwise, the method is said to be **implicit**.

Often, implicit methods require, at each step, the solution of a nonlinear equation for computing u_{n+1}.

Both the Forward Euler and Heun's method are explicit, whereas the Backward Euler and Crank-Nicolson methods are implicit.

Heun's method can be interpreted as the Crank-Nicolson method with the approximation $u_{n+1} \approx u_n + hf_n$ replacing the explicit f_{n+1} term, which depends on u_{n+1}.

9.2 Analysis of One-Step Methods

Any explicit one-step method has the form

$$u_{n+1} = u_n + h\Phi\left(t_n, u_n, f_n, h\right)$$

with Φ the increment function.

Definition 9.9 (Hölder & Lipschitz Continuity)
A function f is **Hölder continuous** if there exists real constants $C > 0$ and $\alpha \geq 0$ such that

$$|f(x) - f(y)| \leq C\|x - y\|^{\alpha}$$

for all x and y. If $\alpha = 1$ the function is said to be **Lipschitz continuous**.

Definition 9.10 (Consistent Schemes)
For the exact solution to the differential equation, $y(t_n) = y_n$, the solution can be written as

$$y_{n+1} = y_n + h\Phi(t_n, y(t_n), f_n, h) + \varepsilon_n$$

so that

$$\tau_n = \frac{y_{n+1} - y_n}{h} - \Phi(t_n, y(t_n), f_n, h)$$

where $\varepsilon_n = h\tau_n$ for a $\tau_n = \tau_n(h)$ is defined as the **local truncation error** at step n.

The **consistency error** is given by $\tau = \max_n |\tau_n|$.

A method is said to be **consistent** if

$$\lim_{h \to 0} \Phi = f.$$

This means the increment function is a good approximation to the differential equation as the step size tends to zero.

Definition 9.11 (Order of One Step Methods)
A one-step method is of order $p \in \mathbb{N}$, if for all $t \in [0, T]$, the solution satisfies the condition that $\tau(h) = \mathcal{O}(h^p)$ as $h \to 0$.

Definition 9.12 (Zero Stable Methods)
A method of the form

$$u_{n+1} = u_n + h\Phi\left(t_n, u_n, f_n, h\right)$$

is called **zero-stable** if there exists both a maximal step size, $h_{\max}$ and a constant, C, such that for all $h \in [0, h_{\max}]$ and for $\varepsilon > 0$, then if, for all time-steps $0 \leq n \leq N$, there exists a $\delta_n \leq \varepsilon$ and

$$z_{n+1} = z_n + h\Phi\left(t_n, z_n, f_n\left(z_n, t_n\right), h\right) + \delta_{n+1}$$

and $z_0 = y_0 + \delta_0$, then

$$|z_n - u_n| \leq C\varepsilon \quad \text{for} \quad 0 \leq n \leq N.$$

Zero stability means that small perturbations in the computations lead to small perturbations in the approximations.

Theorem 17.
If the increment function is Lipschitz continuous for y_n for any h and t_n, then the one-step method is zero-stable.

Theorem 18.
If the increment function Φ is

(i) Lipschitz continuous for u_n for any h and t_{n+1}

and

(ii) the method is consistent

then

$$\lim_{h \to 0} |y_n - u_n| = 0.$$

Also, if the method is of order p and if $|y_0 - u_0| = \mathcal{O}\left(h^p\right)$ as $h \to 0$, the convergence is of order p.

Definition 9.13 (Absolute Stability)
A numerical scheme for approximating the solution to the linear differential equation $y'(t) = \lambda y(t)$ with $\lambda \in \mathbb{C}$ and initial condition $y_0 = 1$ is said to be **absolutely stable** if $|u_n| \to 0$ as $n \to \infty$, when $\mathrm{Re}(\lambda) < 0$, for a fixed value of h.

Definition 9.14 (Well-posed)
A differential equation is said to be **well-posed** if

- a unique solution exists for any initial conditions and
- the solution's behaviour changes continuously with the initial conditions.

A differential equation which does not have these properties is said to be **ill-posed**.

Theorem 19 (Lax Equivalence Theorem).
The Lax Equivalence theorem or Lax–Richtmyer theorem is the equivalent form of the Fundamental Theorem of Numerical Analysis for differential equations, which states that for a *consistent* finite difference method for a well-posed linear initial value problem, the method is *convergent* if and only if it is *stable*.

Runge-Kutta Schemes And Multi-Step Schemes

Definition 9.15 (Runge-Kutta Methods)
If an ordinary differential equation is given by $\dot{y} = f(y, t)$, then a Runge-Kutta scheme takes the form

$$u_{n+1} = u_n + hF(t_n, u_n, h; f)$$

where F is an increment function given by

$$F(t_n, u_n, h; f) = \sum_{i=1}^{s} b_i k_i,$$

with k_i defined as

$$k_i = f\left(u_n + h\sum_{j=1}^{s} a_{i,j}k_j, t_n + c_i h\right) \quad \text{for} \quad i = 1, \ldots, s$$

where s is referred to as the number of stages of the method.

Thus, an s-stage scheme is characterised by coefficients b_i, c_i and $a_{i,j}$. If the matrix defined by the elements $a_{i,j}$ is lower triangular, i.e. $a_{i,j} = 0$ for all $i \leq j$, then each k_i can be computed explicitly in terms of the previous coefficients $k_1, \ldots, k_{i-1}$. Thus, such schemes are called **explicit**, otherwise they are said to be **implicit**.

The **local truncation error** is defined as

$$h\tau_{n+1}(h) = u(t_{n+1}) - u(t_n) - hF(t_n, u_n, h; f).$$

It can be shown that $\tau(h) = \max|\tau_{n+1}(h)| \to 0$ as $h \to 0$ if and only if $\sum_{i=1}^{s} b_i = 1$.

A Runge-Kutta method is of order $p \geq 1$ if $\tau(h) = \mathcal{O}(h^p)$ as $h \to 0$.

The components of a Runge-Kutta scheme are expressed in a **Butcher array**

$$\begin{array}{c|ccc} c_1 & a_{1,1} & \dots & a_{1,s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s,1} & \dots & a_{s,s} \\ \hline & b_1 & \dots & b_s \end{array}$$

The order of an s-stage explicit Runge-Kutta method cannot be greater than s. Additionally, there does not exist a s-stage explicit Runge-Kutta method with order s if $s \geq 5$.

The order of an s-stage implicit Runge-Kutta method cannot be greater than $2s$.

The most common form of the Runge-Kutta method is the fourth order Runge-Kutta method (RK4). It takes the form:

$$u_{n+1} = u_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

where

$$\begin{aligned} k_1 &= f(u_n, t_n), \\ k_2 &= f\left(u_n + \frac{h}{2}k_1, t_n + \frac{h}{2}\right), \\ k_3 &= f\left(u_n + \frac{h}{2}k_2, t_n + \frac{h}{2}\right), \\ k_4 &= f(u_n + hk_3, t_n + h). \end{aligned}$$

9.3 Partial Differential Equations

Definition 9.16 (Partial Differential Equations)
A partial differential equation is a relation involving an unknown function of several free variables and partial derivatives with respect to these variables.

A partial differential equation is said to be linear if it only contains linear terms of the unknown and its derivatives. For example, a second-order linear partial differential equation for an unknown function $u(x, t)$ has the form

$$a_1 u_{xx} + a_2 u_{xt} + a_3 u_{tt} + a_4 u_x + a_5 u_t + a_6 u = f(x,t).$$

where $u_{xx} = \dfrac{\partial^2 u}{\partial x^2}$ is the second-order partial derivative with respect to the variable x and, assuming u is $\mathcal{C}^2$ continuous

$$u_{xt} = \frac{\partial}{\partial t}\frac{\partial u}{\partial x} = \frac{\partial}{\partial x}\frac{\partial u}{\partial t} = u_{tx}$$

For finite-difference schemes, all partial derivatives must be approximated by discretized operators.

Index