MATH 464 Notes

Contents

1 Rounding Errors and Floating Point Arithmetic

1.1 Number Representation

Let β be an integer greater than 1. β is called the base of a number system. Let a_k, b_k be integers such that $0 \leq a_k, b_k < \beta$. Then $(a_n\ a_{n-1}\ ...\ a_1\ a_0\ .\ b_1\ b_2\ ...)_\beta$ represents the real number

$$\begin{aligned} x &= a_n\beta^n + a_{n-1}\beta^{n-1} + ... + a_1\beta^1 + a_0\beta^0 + b_1\beta^{-1} + b_2\beta^{-2} + ... \\ &= \underbrace{\sum_{k=0}^{n} a_k\beta^k}_{\text{integral part}} + \underbrace{\sum_{k=1}^{\infty} b_k\beta^{-k}}_{\text{fractional part}} \end{aligned}$$

- If there exists a value K such that $b_k = 0$ for all $k \geq K$, then the expression is said to terminate

1.2 Scientific Notation

$x = a \times \beta^b$ where β is the base of the number system

- a is the mantissa
- b is the exponent

1.3 Normalized Scientific Notation in Base β

Let β be an integer greater than 1

- For any real number $x > 0$, there is a unique integer c and unique number $r \in [\frac{1}{\beta}, 1)$ such that $\beta^{c-1} \leq r \times \beta^c < \beta^c$
  - $r \in [\frac{1}{\beta}, 1)$ is equivalent to $\frac{1}{\beta} \leq r < 1$
- If $r \in [\frac{1}{\beta}, 1)$, then r can be expressed as an expansion in base β, i.e. $r = (\ .\ d_1\ d_2\ d_3\ ...)_\beta$
- Let $x \neq 0$ be any real number. Then x has an expansion $x = \pm(\ .\ d_1\ d_2\ d_3\ ...)_\beta \times \beta^c$ where $d_1 \neq 0$ in base β

This representation of x is called the normalized scientific notation for x in base β

- It is unique, except for real numbers x with terminating expansions, which have two representations
  - i.e. The number 1 can be represented as $0.999...$ and $1.000...$
- Normalized means $d_1 \neq 0$
  - i.e. 0.4839 is normalized but 0.0371 is not normalized

1.4 Floating Point Numbers

An m-digit floating point number in base β has the form $x = \pm(\ .\ d_1\ d_2\ ...\ d_m)_\beta \times \beta^c$

- If $d_1 \neq 0$ or $x = 0$, then x is a normalized floating point number
- The exponent c has limited range, where $\mu \leq c \leq M$

1.5 Floating Point Approximation

Let $fl(x)$ be the function which assigns a floating point number to a real number x

- Domain of $fl(x)$ is $\{x \mid \beta^{\mu-1} \leq |x| < \beta^M\}$
- If $|x| < \beta^{\mu-1}$, then an underflow error occurs
- If $|x| \geq \beta^M$, then an overflow error occurs

Rounding functions

- $fl(x)_{\text{round}}$ is the normalized floating point number that is closest to x
- May cause overflow errors (i.e. 0.995 rounds to 1.00 which cannot be represented)

Symmetric rounding functions

- If there is a tie between round-offs (i.e. 0.15 can be approximated as 0.1 or 0.2), then round to the even digit (i.e. 0.2)
- Preserves mean of data

Chopping functions

- $fl(x)_{\text{chop}}$ is the nearest normalized floating point number between x and 0

Examples

- $\frac{2}{3} = \begin{cases} (0.67)_{10} \times 10^0 & \text{rounding} \\ (0.66)_{10} \times 10^0 & \text{chopping} \end{cases}$
- $0.995 = \begin{cases} \text{overflow error} & \text{rounding} \\ (0.99)_{10} \times 10^0 & \text{chopping} \end{cases}$
- $\left.\begin{array}{l} 145 = (0.14)_{10} \times 10^3 \\ 155 = (0.16)_{10} \times 10^3 \end{array}\right\}$ symmetric rounding

1.6 Precise Definition of $fl(x)$

- If $x = 0$, then $fl(x) = (\ .\ 0\ 0\ ...\ 0)_\beta \times \beta^c$
- If $0 < |x| < \beta^{\mu-1}$, then an underflow error occurs
- If $|x| \geq \beta^M$, then an overflow error occurs
- If $\beta^{\mu-1} \leq |x| < \beta^M$, then x is in the usual domain for $fl(x)$

Given a value x in base β where $x = \pm(\ .\ d_1\ d_2\ ...\ d_m\ d_{m+1}\ ...)_\beta \times \beta^c$ where $\mu \leq c \leq M$ and $d_1 \neq 0$

- Definition of rounding functions
  - If $(\ .\ d_{m+1}\ d_{m+2}\ ...)_\beta < \frac{1}{2}$, then $fl(x)_{\text{round}} = \pm(\ .\ d_1\ d_2\ ...\ d_m) \times \beta^c$
  - If $(\ .\ d_{m+1}\ d_{m+2}\ ...)_\beta > \frac{1}{2}$, then $fl(x)_{\text{round}} = \pm(\ .\ d_1\ d_2\ ...\ d_m +\ .\ 0_1\ 0_2\ ...\ 1_m) \times \beta^c$
  - If $(\ .\ d_{m+1}\ d_{m+2}\ ...)_\beta = \frac{1}{2}$, then round down if d_m is even and round up if d_m is odd
- Definition of chopping functions
  - $fl(x)_{\text{chop}} = \pm(\ .\ d_1\ d_2\ ...\ d_m)_\beta \times \beta^c$

1.7 Absolute and Relative Errors

Suppose x^* is an approximation to a real number x

- The absolute error in x^* is $x - x^*$
- The relative error in x^* is $\frac{x-x^*}{x}$

1.8 Relative Error Bounds

Suppose $\beta^{\mu-1} \leq |x| < \beta^M$. Define $\delta = \frac{fl(x)-x}{x}$

- In rounding, $|\delta| \leq \frac{1}{2}\beta^{1-m}$
- In chopping, $-\beta^{1-m} < \delta \leq 0$

The maximum possible value for $|\delta|$ is called the unit roundoff, denoted as u

- In rounding, $u = \frac{1}{2}\beta^{1-m}$
- In chopping, $u = \beta^{1-m}$

1.9 Alternate Representation of Relative Errors

- If $\delta = \frac{fl(x)-x}{x}$, then $fl(x) = x(1+\delta)$
- If $\delta = \frac{fl(x)-x}{fl(x)}$, then $fl(x) = \frac{x}{1+\delta}$

1.10 Floating Point Arithmetic

Floating point numbers have the usual arithmetic operations, $+, -, \times, \div$

- If β, m are fixed, the set of floating point numbers is not closed under arithmetic operations
  - i.e. if $x = (.201) \times 10^1$ and $y = (.304) \times 10^0$, then $x + y = (.2314) \times 10^1$
  - The mantissa is too short to represent the resulting number
  - Arithmetic operations introduce a relative error due to the need to approximate results
- Arithmetic operations on floating point numbers may not be associative/distributive

1.11 Machine Floating Point Arithmetic

In machines, $x\ \omega^*\ y = fl(x\ \omega\ y)(1+\delta)$ or $fl(x\ \omega\ y)/(1+\delta)$ where ω is the true operation, ω^* is the machine / floating point operation, and $(1+\delta)$, $/(1+\delta)$ is the relative error caused by operation

- i.e. Suppose x_1, x_2, x_3, x_4 are positive floating point numbers. Let $+$ represent true addition and $\oplus$ represent machine addition

$$x_1 \oplus x_2 = (x_1 + x_2)(1+\delta_1)$$

$$\begin{aligned}(x_1 \oplus x_2) \oplus x_3 &= ((x_1+x_2)(1+\delta_1) + x_3)(1+\delta_2) \\ &= (x_1+x_2)(1+\delta_1)(1+\delta_2) + x_3(1+\delta_2)\end{aligned}$$

- i.e. Suppose p, q are real numbers. Let $\cdot$ represent true multiplication and $*$ represent machine multiplication

$$fl(p) = p(1+\delta_1)$$

$$fl(q) = q(1+\delta_2)$$

$$\begin{aligned}fl(p) * fl(q) &= (fl(p) \cdot fl(q))(1+\delta_3) \\ &= pq(1+\delta_1)(1+\delta_2)(1+\delta_3)\end{aligned}$$

- i.e. Suppose x, y, w, z are floating point numbers. Let $\frac{*}{*}$ represent true division and $/$ represent machine division

$$x * y = xy(1+\delta_1)$$

$$w * z = \frac{wz}{1+\delta_2}$$

$$\begin{aligned}(x*y)/(w*z) &= \frac{x*y}{w*z}(1+\delta_3) \\ &= \frac{xy(1+\delta_1)}{\frac{wz}{1+\delta_2}}(1+\delta_3) \\ &= \frac{xy}{wz}(1+\delta_1)(1+\delta_2)(1+\delta_3)\end{aligned}$$

- Suppose $0 < u < 1$ and $|\delta_j| \le u$ for $j = 1, 2, ..., r$. Then there exists δ where $|\delta| \le u$ such that $(1+\delta_1)(1+\delta_2)...(1+\delta_r) = (1+\delta)^r$
  - Simplifies representation of combined relative errors
    - i.e. $(x_1+x_2)(1+\delta_1)(1+\delta_2) + x_3(1+\delta_2) = (x_1+x_2+x_3)(1+\delta)^2$ for some δ
    - i.e. $pq(1+\delta_1)(1+\delta_2)(1+\delta_3) = pq(1+\delta)^3$ for some δ
    - i.e. $\frac{xy}{wz}(1+\delta_1)(1+\delta_2)(1+\delta_3) = \frac{xy}{wz}(1+\delta)^3$ for some δ
  - If $r \cdot u << 1$ and $|\delta| \le u$, then $(1+\delta)^r \approx 1 + r\delta$
    - Easy way to calculate relative error after performing r operations

1.12 Significant Digits

Suppose $x = (\ .\ d_1\ d_2\ d_3\ ...)_\beta \times \beta^c$ with $d_1 \ne 0$ and approximation x^*. The digit d_j is called the j^{th} significant digit of x

- If the absolute error $|x - x^*| \le \frac{1}{2}\beta^{c-r}$, then x^* approximates x to r significant digits

- The number of significant figures in x^* is very approximately $-\log_\beta \left|\frac{x-x^*}{x}\right|$

1.13 Cancellation Errors

Cancellation error is the loss of significant digits caused by subtraction. Very approximately, if x and y have t significant digits, have the same sign, and agree to s significant digits, then the computed value of $x - y$ will have only $t - s$ significant digits

- i.e. Compute $\frac{22}{7} - \pi$

  $x^* = fl(\frac{22}{7}) = (.31429) \times 10^1$

  $y^* = fl(\pi) = (.31416) \times 10^1$

  $x^* - y^* = (0.00013) \times 10^1 = (.13000) \times 10^{-2}$

  x^* and y^* have 5 significant digits and agree to 3 significant digits, such that $x^* - y^*$ has 2 significant digits

1.14 Avoiding Cancellation Errors

- In the quadratic formula, where $ax^2 + bx + c = 0$ and $a, b, c \neq 0$ and $b^2 - 4ac > 0$
  - When $b > 0$, $x = \frac{2c}{-b-\sqrt{b^2-4ac}}, \frac{-b-\sqrt{b^2-4ac}}{2a}$
  - When $b < 0$, $x = \frac{2c}{-b+\sqrt{b^2-4ac}}, \frac{-b+\sqrt{b^2-4ac}}{2a}$

1.15 Machine Epsilon

The machine epsilon ε is the supremum of the set $\{y > 0 \mid fl(1 + y) = 1\}$

- The supremum of a set is the smallest element of the set that is greater than or equal to all other elements in the set
- The machine epsilon of $fl(x)_{\text{round}}$ is equal to $u = \frac{1}{2}\beta^{1-m}$
- The machine epsilon of $fl(x)_{\text{chop}}$ is equal to $u = \beta^{1-m}$
- Machine epsilon and approximation errors are two definitions of the same number

1.16 Integer Base Conversion Algorithms

Suppose we want to convert $N = (a_n\, a_{n-1}\, ...\, a_0)_\alpha$ from base α to base β

- Algorithm 1: Using base β arithmetic. Best if $\alpha < \beta$

  Express $\alpha, a_0, a_1, ..., a_n$ in base β and perform the following calculation in base β arithmetic

  $$N = (((a_n \cdot \alpha + a_{n-1}) \cdot \alpha + ...) \cdot \alpha + a_1) \cdot \alpha + a_0 \leftarrow \text{nested multiplication, more efficient}$$
  $$= a_n \cdot \alpha^n + a_{n-1} \cdot \alpha^{n-1} + ... + a_0 \leftarrow \text{polynomial multiplication, more readable}$$

- Algorithm 2: Using base α arithmetic. Best if $\alpha > \beta$

  Suppose $N = (c_m\, c_{m-1}\, ...\, c_0)_\beta$

  $$N = c_0 + \beta \cdot (c_1 + \beta \cdot (c_2 + ...))$$

  1. c_0 is the remainder and $c_1 + \beta \cdot (c_2 + ...)$ is the quotient when N is divided by β in base α arithmetic
  2. c_1 is the remainder and $c_2 + ...$ is the quotient when the quotient in the previous step is divided by β in α arithmetic

     $\vdots$
  3. c_m is the remainder and 0 is the quotient when the quotient in the previous step is divided by β in α arithmetic

1.17 Fraction Base Conversion Algorithms

Suppose we want to convert $x = (\,.\, b_1\, b_2\, ...\, b_m)_\alpha$ from base α to base β

- Algorithm 1: Using base β arithmetic. Best if $\frac{1}{\alpha}$ has a terminating expansion in base β

  Express $\alpha, b_1, b_2, ..., b_m$ in base β and perform the following calculation in base β arithmetic

  $$x = ((((b_m)/\alpha + b_{m-1})/\alpha + ...)/\alpha + b_1)/\alpha + b_0 \leftarrow \text{nested multiplication, more efficient}$$
  $$= b_m \cdot \alpha^{-m} + b_{m-1} \cdot \alpha^{-m+1} + ... + b_0 \leftarrow \text{polynomial multiplication, more readable}$$

- Algorithm 2: Using base α arithmetic. Best if $\frac{1}{\alpha}$ does not have a terminating expansion in base β

  Suppose $x = (\,.\, c_1\, c_2\, c_3\, ...)_\beta$

  1. $\beta x = (c_1\, .\, c_2\, c_3\, ...)_\beta$, such that $c_1 = (\beta x)_I$
  2. $(\beta x)_F = (\,.\, c_2\, c_3\, ...)_\beta$
  3. $\beta(\beta x)_F = (c_2\, .\, c_3\, ...)_\beta$, such that $c_2 = (\beta(\beta x)_F)_I$

     $\vdots$

1.18 Base 2 to Base 8 Conversion Algorithm

Since $8 = 2^3$, we arrange numbers in groups of 3

$$a_5\, a_4\, a_3\, a_2\, a_1\, a_0\, .\, b_1\, b_2\, b_3 = (a_5 \cdot 4 + a_4 \cdot 2 + a_3) \cdot 8 + (a_2 \cdot 4 + a_1 \cdot 2 + a_0) + (b_1 \cdot 4 + b_2 \cdot 2 + b_3) \cdot 8^{-1}$$

2 Linear Algebra

2.1 Matrices

A matrix is a rectangular array of numbers arranged in rows and columns

$$A^{m\times n} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

2.2 Matrix Multiplication

If A is an $m \times n$ matrix and B is an $n \times p$ matrix, then the product $C = AB$ is an $m \times p$ matrix

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

2.3 Diagonal and Triangular Matrices

Suppose A is a square matrix $n \times n$

- $a_{11}, a_{22}, ..., a_{nn}$ are called the diagonal entries of A
- a_{ij} with $i \neq j$ are called the off-diagonal entries of A
- a_{ij} with $i < j$ are called the super-diagonal entries of A
- a_{ij} with $i > j$ are called the sub-diagonal entries of A
- If $a_{ij} = 0$ for $i \neq j$, then A is called a diagonal matrix
- If $a_{ij} = 0$ for $i > j$, then A is called upper-triangular
- If $a_{ij} = 0$ for $i < j$, then A is called lower-triangular

2.4 Invertible Matrices

An $n \times n$ matrix A is invertible if there is a matrix B such that $AB = I_n$. B is the inverse of A and denoted as A^{-1}, where $AA^{-1} = A^{-1}A = I_n$

- Only square matrices are invertible
- The inverse of an inverted matrix is the matrix itself
- Zero matrices have no inverse
- An invertible matrix is called non-singular
- A non-invertible matrix is called singular

Assuming A and B are $n \times n$ invertible matrices

- $(A^{-1})^{-1} = A$
- $(AB)^{-1} = B^{-1}A^{-1}$
- $AC = 0$ implies $C = 0$
- $AC = AD$ implies $C = D$, when A is invertible
- $(kA)^{-1} = \frac{1}{k}(A)^{-1}$
- $(A^T)^{-1} = (A^{-1})^t$

2.5 Matrix Addition

Given that A and B have the same order, $A + B = C$ where $c_{ij} = a_{ij} + b_{ij}$

2.6 Scalar Multiplication

Given a matrix A and a scalar α, $(\alpha \cdot A)_{ij} = \alpha \cdot a_{ij}$

2.7 Properties of Matrix Addition and Scalar Multiplication

- $A + B = B + A$
- $(A + B) + C = A + (B + C)$
- $\alpha(A + B) = \alpha A + \alpha B$
- $(\alpha + \beta)A = \alpha A + \beta A$
- If A is invertible and $\alpha \neq 0$, then αA is invertible and $(\alpha A)^{-1} = \frac{1}{\alpha}A^{-1}$

2.8 Null Matrix

The null matrix is the matrix with all zero entries

2.9 Linear Combinations

Suppose we have n-vectors $x_1, x_2, ..., x_k$ and scalars $b_1, b_2, ..., b_k$. Then $b_1x_1 + b_2x_2 + ... + b_kx_k$ is called a linear combination

- If A is an $n \times m$ matrix, then a_j is the m-vector representing the j^{th} column of A
- If A is an $n \times m$ matrix and x is an n-vector, then $Ax = x_1a_1 + x_2a_2 + ... + x_na_n$ is a linear combination of the columns of A

2.10 Unit Vectors

Let i_j be the j^{th} column of I_n. Then

$$i_j = \begin{bmatrix} 0_1 \\ \vdots \\ 1_j \\ \vdots \\ 0_n \end{bmatrix}$$

Where i_j is the j^{th} unit vector

2.11 Existence and Uniqueness of Solutions to $Ax = b$

Assuming A is an $m \times n$ matrix, b is an m-vector, and x is an n-vector

- If x_1 is a solution of $Ax = b$, then any other solution x_2 is of the form $x_2 = x_1 + y$ where y is a solution of the homogeneous system $Ay = 0$
- $Ax = b$ has at most one solution if and only if $Ay = 0$ only has the trivial solution $y = 0$
- Any homogeneous solution linear system with fewer equations than unknowns has non-trivial solutions
- If $Ax = b$ has a solution for every m-vector b, then there exists an $n \times m$ matrix C such that $AC = I_m$
- If $BA = I$, then $Ax = 0$ has only the trivial solution
- If $Ax = b$ has a solution for every m-vector b, then $m \leq n$
- Let A be a square $n \times n$ matrix. The following are equivalent
  - $Ax = 0$ only has the trivial solution $x = 0$
  - $Ax = b$ has a solution for every n-vector b
  - A is invertible

2.12 Linear Independence

Suppose we have m-vectors $a_1, a_2, ..., a_n$ and scalars $x_1, x_2, ..., x_n$. Then $a_1, a_2, ..., a_n$ are linearly independent if $x_1a_1 + x_2a_2 + ... + x_na_n = 0$ implies $x_1 = x_2 = ... = x_n = 0$

- If $a_1, a_2, ..., a_n$ are not linearly independent, then they are linearly dependent
- Let A be the $m \times n$ matrix whose columns are $a_1, a_2, ..., a_n$. Then $a_1, a_2, ..., a_n$ are linearly independent if and only if $Ax = 0$ only has the trivial solution
- Any set of more than m m-vectors is linearly dependent

2.13 Basis

If every m-vector b can be written as a linear combination of independent vectors $a_1, a_2, ..., a_n$, then $a_1, a_2, ..., a_n$ form a basis

- Let A be the $m \times n$ matrix whose columns are $a_1, a_2, ..., a_n$. Then $a_1, a_2, ..., a_n$ form a basis if and only if $Ax = b$ has a unique solution for each b
- If the m-vectors $a_1, a_2, ..., a_n$ form a basis, then $m = n$

2.14 Transpose of a Matrix

If $A = (a_{ij})$ is a $m \times n$ matrix, then its transpose $A^T = (a_{ji})$ is a $n \times m$ matrix. The elements of the matrix are mirrored along the diagonal

- If $A^T = A$, then A is called symmetric
- $(AB)^T = B^TA^T$
- $(A^T)^T = A$
- If A is invertible, then $(A^T)^{-1} = (A^{-1})^T$

2.15 Conjugate/Hermitian Transpose

Given the complex matrix $(A^H)_{ij} = \bar{a}_{ji}$, where $\bar{a}_{ji}$ is the conjugate of a_{ji}

- If A is real, then $A^H = A^T$
- If $A^H = A$, then A is called Hermitian
  - Complex extension of a symmetric matrix

2.16 Scalar Product

If a, b are real n-vectors, the scalar product $b \cdot a = b^Ta = b_1a_1 + b_2a_2 + ... + b_na_n$

- This is the dot product of real vectors

If a, b are complex n-vectors, the scalar product $b \cdot a = b^Ha = \bar{b_1}a_1 + \bar{b_2}a_2 + ... + \bar{b_n}a_n$

- This is the dot product of complex vectors

2.17 Permutations

A permutation $p : \{1, 2, ..., n\} \to \{1, 2, ..., n\}$ of degree n is a bijective mapping from the first n integers into itself

- There are $n!$ such permutations, since there are $n!$ ways to construct a bijective mapping between the two sets
- A permutation is even if it takes an even number of interchanges to restore the mapping to its ordered state (i.e. $1 \to 1$, $2 \to 2$, ..., $n \to n$)
- A permutation is odd if it takes an odd number of interchanges to restore the mapping to its ordered state (i.e. $1 \to 1$, $2 \to 2$, ..., $n \to n$)

2.18 Determinants

Let S_n denote the symmetric group of degree n, which is the set of all permutations p of degree n

Let us define the sign of p as $\operatorname{sgn}(p) = \begin{cases} 1 & \text{if } p \text{ is an even permutation} \\ -1 & \text{if } p \text{ is an odd permutation} \end{cases}$

Let A be an $n \times n$ matrix. Then the determinant of A is

$$\det(A) = \sum_{p \in S_n} \operatorname{sgn}(p)\ a_{1,p(1)} a_{2,p(2)} ... a_{n,p(n)}$$

Theorems on determinants

- If A is diagonal or upper or lower triangular, then $\det(A) = a_{11}a_{22}...a_{nn}$
- If A and B are $n \times n$ matrices, $\det(AB) = \det(A) \cdot \det(B)$
- A is invertible if and only if $\det(A) \neq 0$

3 Solution of Linear Systems

3.1 Invertible Square Linear Systems

Invertible square linear systems are of the form

$$Ax = b$$

Where A is an $n \times n$ invertible matrix, b is an n-vector, and x is an unknown n-vector

3.2 Invertible Matrices Theorem

- An upper triangular matrix A is invertible if and only if all diagonal entries are non-zero

3.3 Gaussian Elimination

Gaussian elimination generates a sequence of equivalent linear systems

$$A^{(k)}x = b^{(k)} \text{ for } 0 \leq k \leq n-1$$

where $A^{(0)} = A$, $b^{(0)} = b$, and $A^{(n-1)}$ is upper triangular

$$\begin{bmatrix} {a_{11}}^{(k-1)} & \dots & \dots & \dots & {a_{1n}}^{(k-1)} \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & 0 & {a_{kk}}^{(k-1)} & \dots & {a_{kn}}^{(k-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & {a_{nk}}^{(k-1)} & \dots & {a_{nn}}^{(k-1)} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_k \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} {b_1}^{(k-1)} \\ \vdots \\ {b_k}^{(k-1)} \\ \vdots \\ {b_n}^{(k-1)} \end{bmatrix}$$

1. Assume by induction that the $(k-1)^{\text{th}}$ system has the above form
2. If ${a_{kk}}^{(k-1)} \neq 0$, add a multiple $-m_{ik} = -\frac{{a_{ik}}^{(k-1)}}{{a_{kk}}^{(k-1)}}$ of the k^{th} row to the i^{th} row for $i = k+1, k+2, ..., n$
3. Then ${a_{ik}}^{(k)} = 0$ for $i = k+1, k+2, ..., n$

This can be represented in matrix form as

$$M_1 = \begin{bmatrix} 1 & 0 & \dots & 0 \\ -m_{21} & 1 & \ddots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -m_{n1} & 0 & \dots & 1 \end{bmatrix} \qquad M_k = \begin{bmatrix} 1 & 0 & \dots & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots & \ddots & \vdots \\ \vdots & \ddots & 1 & 0 & \ddots & \vdots \\ \vdots & \ddots & -m_{k+1,k} & 1 & \ddots & \vdots \\ \vdots & \ddots & \vdots & \vdots & \ddots & 0 \\ 0 & \dots & -m_{n,k} & 0 & \dots & 1 \end{bmatrix}$$

where $A^{(1)} = M_1 A^{(0)},\ A^{(2)} = M_2 A^{(1)},\ ...,\ A^{(k)} = M_k A^{(k-1)}$

3.4 Back Substitution

If A is an $n \times n$ upper triangular matrix with all diagonal entries non-zero

$$\begin{aligned} x_n &= \frac{b_n}{a_{n,n}} \\ x_{n-1} &= \frac{b_{n-1} - a_{n-1,n} x_n}{a_{n-1,n-1}} \\ x_k &= \frac{b_k - \sum_{j=k+1}^{n} a_{k,j} x_j}{a_{k,k}} \end{aligned}$$

3.5 Matrix Form of Row Operations

Representing row operations on an $n \times n$ matrix A as a matrix product RA

- Multiplying the i^{th} row of A by α, where $\alpha \neq 0$

$$R = \begin{array}{cc} \begin{bmatrix} 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \alpha & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 & 0 \\ 0 & \dots & \dots & 0 & 1 \end{bmatrix} & \begin{array}{c} \\ \\ \leftarrow i \\ \\ \\ \end{array} \\ \begin{array}{ccccc} & & \uparrow & & \\ & & i & & \end{array} & \end{array}$$

- Adds α times the j^{th} row of A to the i^{th} row of A, where $i \neq j$

$$R = \begin{array}{cc} \begin{bmatrix} 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & \ddots & \ddots & \vdots \\ \vdots & \alpha & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 & 0 \\ 0 & \dots & \dots & 0 & 1 \end{bmatrix} & \begin{array}{c} \\ \\ \leftarrow i \\ \\ \\ \end{array} \\ \begin{array}{ccccc} & \uparrow & & & \\ & j & & & \end{array} & \end{array}$$

- Interchanges the i^{th} and j^{th} rows of A, where $i \neq j$

$$R = \begin{array}{cc} \begin{bmatrix} 1 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 0 & \dots & 1 & \ddots & \vdots \\ \vdots & \ddots & \vdots & 1 & \vdots & \ddots & \vdots \\ \vdots & \ddots & 1 & \dots & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 1 & 0 \\ 0 & \dots & \dots & \dots & \dots & 0 & 1 \end{bmatrix} & \begin{array}{c} \\ \\ \leftarrow i \\ \\ \leftarrow j \\ \\ \\ \end{array} \\ \begin{array}{ccccccc} & & \uparrow & & \uparrow & & \\ & & i & & j & & \end{array} & \end{array}$$

3.6 Triangular/LU Factorization

Assume that no pivoting is necessary. Then

$$A^{(n-1)} = M_{n-1}M_{n-2}...M_1A^{(0)}$$

Let $U = A^{n-1}$ such that U is an upper triangular matrix with non-zero diagonal elements. Then

$$A^{(0)} = {M_1}^{-1}{M_2}^{-1}...\ {M_{n-1}}^{-1}U$$

Since M_k are lower triangular matrices for all $0 \le k \le n-1$, $L = {M_1}^{-1}{M_2}^{-1}...\ {M_{n-1}}^{-1}$ is a lower triangular matrix with non-zero diagonal elements. Then

$$A^{(0)} = LU$$

Since $A = A^{(0)}$, we can write $A = LU$

- An invertible matrix A has an LU factorization if and only if each of the upper left hand submatrices $\begin{bmatrix} a_{11} & \dots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{k1} & \dots & a_{kk} \end{bmatrix}$ are invertible for $1 \le k \le n$

3.7 Solution By Triangular/LU Factorization

Let y be the solution of $Ly = b$. Since $L = {M_1}^{-1}{M_2}^{-1}...\ {M_{n-1}}^{-1}$, then $y = M_{n-1}M_{n-2}...M_1b$. Use forward substitution to solve for y, and then use back substitution to solve for x

- Once L and U is calculated for a matrix A, we can easily solve for x in $Ax = b$ for any b

3.8 Pivoting

At the k^{th} step of Gaussian elimination, the current matrix is of the form

$$A^{(k-1)} = \begin{bmatrix} {a_{11}}^{(k-1)} & \dots & \dots & \dots & {a_{1n}}^{(k-1)} \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & 0 & {a_{kk}}^{(k-1)} & \dots & {a_{kn}}^{(k-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & {a_{nk}}^{(k-1)} & \dots & {a_{nn}}^{(k-1)} \end{bmatrix}$$

In order to avoid errors with small numbers, we must select an appropriate element ${a_{kk}}^{(k-1)}, ..., {a_{nk}}^{(k-1)}$ to use as the k^{th} pivot element

- Every invertible matrix A can be written as PLU where P is a permutation matrix, L is a lower triangular matrix, and U is an invertible upper triangular matrix

3.9 Simple Pivoting

Choose as the pivot row the smallest $I \geq k$ for which ${a_{Ik}}^{(k-1)} \neq 0$. Interchange the k^{th} row and the I^{th} row

3.10 Partial Pivoting

Choose as the pivot row the smallest $I \geq k$ for which $|{a_{Ik}}^{(k-1)}|$ is the maximum of $|{a_{kk}}^{(k-1)}|, ..., |{a_{nk}}^{(k-1)}|$. Interchange the k^{th} row and the I^{th} row

3.11 Scaled Partial Pivoting Without Updating Scale Factors

Before starting the elimination procedure, store the scale factors

$$d_i = \max_{1 \leq j \leq n} |a_{ij}| \text{ for } i = 1, .., n$$

When rows are interchanged, their scale factors d_i are also interchanged. We do not recompute new scale factors

At the k^{th} step, choose as the pivot row the smallest $I \geq k$ for which $\dfrac{|{a_{Ik}}^{(k-1)}|}{d_I}$ is the maximum of $\dfrac{|{a_{kk}}^{(k-1)}|}{d_k}, ..., \dfrac{|{a_{nk}}^{(k-1)}|}{d_n}$. Interchange the k^{th} row and the I^{th} row

3.12 Scaled Partial Pivoting With Updating Scale Factors

At the k^{th} step, first update the scale factors. Let

$${d_i}^{(k-1)} = \max_{k \leq j \leq n} |{a_{ij}}^{(k-1)}| \text{ for } i = k, ..., n$$

Choose as the pivot row the smallest $I \geq k$ for which $\dfrac{|{a_{Ik}}^{(k-1)}|}{{d_I}^{(k-1)}}$ is the maximum of $\dfrac{|{a_{kk}}^{(k-1)}|}{{d_k}^{(k-1)}}, ..., \dfrac{|{a_{nk}}^{(k-1)}|}{{d_n}^{(k-1)}}$. Interchange the k^{th} row and the I^{th} row

3.13 Total Pivoting

In total pivoting, both rows and columns are interchanged. At the k^{th} step, choose $I \geq k$ and $J \geq k$ for which $|{a_{IJ}}^{(k-1)}|$ is the maximum of $|a_{ij}|$ for $i = k, ..., n$ and $j = k, ..., n$. Interchange the k^{th} row and I^{th} row. Interchange the k^{th} column and J^{th} column

3.14 Matrix Form of Interchanges

Let A be an invertible matrix A with factorization $A = PLU$. Then

$$U = (M_{n-1}P_{n-1})(M_{n-2}P_{n-2})...(M_1P_1)$$

where P_k is either the identity (if no pivot at k^{th} step) or P_k is a permutation matrix that interchanges the k^{th} row and the I^{th} row

$$P_k = \begin{array}{c@{}l} \begin{bmatrix} 1 & 0 & \dots & \dots & \dots & 0 \\ 0 & 1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & 1 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 1 & 0 \\ 0 & \dots & \dots & \dots & 0 & 1 \end{bmatrix} & \begin{array}{l} \\ \\ \leftarrow k \\ \leftarrow I \\ \\ \\ \end{array} \\ \begin{array}{cccccc} & & \uparrow & \uparrow & & \\ & & k & I & & \end{array} & \end{array}$$

Suppose $k > l$ and P_k interchanges the k^{th} row and i^{th} row where $I > k$. Then $P_k M_l = \widetilde{M_l} P_k$, where $\widetilde{M_l}$ is the same as M_l except that the multipliers m_{kl} and m_{Il} have been interchanged

$$P_k M_l = \widetilde{M_l} P_k = \begin{array}{c@{}l} \begin{bmatrix} 1 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & m_{Il} & \ddots & 0 & 1 & \ddots & \vdots \\ \vdots & m_{kl} & \ddots & 1 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 1 & 0 \\ 0 & \dots & \dots & \dots & \dots & 0 & 1 \end{bmatrix} & \begin{array}{l} \\ \leftarrow l \\ \\ \leftarrow k \\ \leftarrow I \\ \\ \\ \end{array} \\ \begin{array}{ccccccc} & \uparrow & & \uparrow & \uparrow & & \\ & l & & k & I & & \end{array} & \end{array}$$

4 Norms

4.1 Vector Norms on $\mathbb{R}^n$ and $\mathbb{C}^n$

The norm $||a||$ is a number assigned to each real or complex n-vector a. Vector norms satisfy the following properties

- For all vectors a, $||a|| \geq 0$ and $||a|| = 0$ if and only if $a = 0$
  - The only vector with zero length is the zero vector
- For vectors a and all scalars $\alpha \in \mathbb{R}$ or $\mathbb{C}$, $||\alpha a|| = |\alpha| \cdot ||a||$
  - Scaling a vector also scales its norm
- For all vectors a, b, $||a + b|| \leq ||a|| + ||b||$
  - In a triangle, the sum of lengths of two sides is greater than or equal to the length of the remaining side

Common vector norms

- $||a||_1 = \sum_{j=1}^{n} |a_j|$
  - Referred to as the 'one norm'
  - This is the absolute vector sum
- $||a||_2 = \left(\sum_{j=1}^{n} |a_j|^2\right)^{\frac{1}{2}}$
  - Referred to as the 'two/Euclidean norm'
  - This is the root of the absolute square vector sum
- $||a||_\infty = \max_{1 \leq j \leq n} |a_j|$
  - Referred to as the 'infinity/max norm'
  - This is the maximum absolute element

4.2 Matrix Norms

The set of real or complex $n \times n$ matrices is itself a vector space. Matrix norms satisfy the following properties

- For all matrices A, $||A|| \geq 0$ and $||A|| = 0$ if and only if $A = 0$
- For matrices A and all scalars $\alpha \in \mathbb{R}$ or $\mathbb{C}$, $||\alpha A|| = |\alpha| \cdot ||A||$
- For all matrices A, B, $||A + B|| \leq ||A|| + ||B||$
- For all matrices A, B, $||AB|| \leq ||A|| \cdot ||B||$

4.3 Operator Norms

Given a vector norm on $\mathbb{R}^n$ or $\mathbb{C}^n$, the operator norm of an $n \times n$ matrix is

$$||A|| = \sup_{x \neq 0} \left(\frac{||Ax||}{||x||} \right)$$

Operator norms satisfy the following properties

- $||A|| < \infty$
- For all vectors x, $||Ax|| \leq ||A|| \cdot ||x||$
- There exists $x \neq 0$ such that $||Ax|| = ||A|| \cdot ||x||$
- $||A|| = \max\limits_{||x||=1} ||Ax||$
- Operator norms are matrix norms

4.4 Operator Norms Induced by $|| \cdot ||_1$ and $|| \cdot ||_\infty$

The operator norms induced by $|| \cdot ||_p$ for $p = 1, 2, \infty$ are denoted as $||A||_p$

- $||A||_1 = \max\limits_{1 \leq j \leq n} \sum\limits_{i=1}^{n} |a_{ij}|$
  - Analogous to the vector 'one norm'
  - This is the maximum absolute column sum
- $||A||_\infty = \max\limits_{1 \leq i \leq n} \sum\limits_{j=1}^{n} |a_{ij}|$
  - Analogous to the vector 'infinity/max norm'
  - This is the maximum absolute row sum

4.5 Compatible Matrix Norms

A matrix norm $|| \cdot ||_m$ is compatible with a vector norm $|| \cdot ||_v$ if for every $(n \times n)$ matrix A and n-vector x, $||Ax||_v \leq ||A||_m \cdot ||x||_v$

4.6 Frobenius Norm

The Frobenius norm of A is $||A||_F = \left(\sum\limits_{i=1}^{n} \sum\limits_{j=1}^{n} |a_{ij}|^2 \right)^{\frac{1}{2}}$

- The Frobenius norm is compatible with $|| \cdot ||_2$

5 Error Analysis

5.1 Condition Number

The condition number of a matrix A is defined as $\kappa(A) = ||A|| \cdot ||A^{-1}||$

- If $\kappa(A)$ is small, A is well-conditioned
- If $\kappa(A)$ is large, A is ill-conditioned
  - For a small change in the value of A, there is a large change in the solution x of $Ax = b$

5.2 Invertible Conditions

- $\dfrac{1}{\kappa(A)} = \min \left\{ \dfrac{||A - B||}{||A||} \mid B \text{ is not invertible} \right\}$
- If A is invertible and $||A - B|| < \dfrac{1}{||A^{-1}||}$, then B is invertible

5.3 Error and Residual Vectors

Consider the linear system $Ax = b$. Let x be the true solution and let $\widehat{x}$ be the approximate solution. Then the error vector $e = x - \widehat{x}$ and residual vector $r = b - A\widehat{x} = Ax - A\widehat{x} = Ae$

5.4 Estimating Error from Residual Vector

The size of the residual r is often a good indicator on the size of the error e

- $\dfrac{||r||}{||A||} \leq ||e|| \leq ||A^{-1}|| \cdot ||r||$
- $\dfrac{1}{\kappa(A)} \dfrac{||r||}{||b||} \leq \dfrac{||e||}{||x||} \leq \kappa(A) \dfrac{||r||}{||b||}$

5.5 Iterative Improvement

We want to find x in $Ax = b$, given A, b, and an approximate solution $\widehat{x}$

1. Calculate $r = b - A\widehat{x}$ using double precision
2. Solve $Ae = r$ and call the solution e^*

3a. If $||e^*||/||\widehat{x}||$ is small enough, let $x = \widehat{x} + e^*$

3b. If $||e^*||/||\widehat{x}||$ is not small enough, let $\widehat{x} = \widehat{x} + e^*$ and repeat

5.6 Backward Error Analysis

We want to find the relative error $\frac{||x-\widehat{x}||}{||\widehat{x}||}$ in $Ax = b$, given A, b, and an approximate solution $\widehat{x}$

1. Express the approximate solution $\widehat{x}$ as the exact solution of an equation $\widehat{A}\widehat{x} = b$
2. Define $E = A - \widehat{A}$
3. Then $\dfrac{||x-\widehat{x}||}{||\widehat{x}||}$ is bounded by $\dfrac{||x-\widehat{x}||}{||\widehat{x}||} \leq \kappa(A) \cdot \dfrac{||E||}{||A||}$

If $||E|| < \dfrac{1}{||A^{-1}||}$, then $\widehat{A} = A - E$ is invertible

5.7 PLU Error Analysis

We want to find the relative error $\frac{||x-\widehat{x}||}{||\widehat{x}||}$ in $Ax = b$, given A, b, and the PLU solution with scaled partial pivoting $\widehat{x}$

1. Let the n-vector $u_n = 1.01 \cdot n \cdot u$, where u is the unit roundoff
2. Define the matrix E as $|e_{ij}| \leq u_n|(P^T A)_{ij}| + u_n(3 + u_n)\displaystyle\sum_{k=1}^{n} |\widehat{l}_{ik}| \cdot |\widehat{u}_{kj}|$
3. Then E is approximately bounded by $||E|| \leq ||A|| \cdot n \cdot u$
4. Express the approximate solution $\widehat{x}$ as the exact solution of an equation $(A + PE)\widehat{x} = b$
5. Then $\dfrac{||x-\widehat{x}||}{||\widehat{x}||}$ is bounded by $\dfrac{||x-\widehat{x}||}{||\widehat{x}||} \leq \kappa(A) \cdot n \cdot u$

Let $s = -\log_{10}(\kappa(A) \cdot n \cdot u)$. Then $\widehat{x}$ is correct up to about s significant digits. Using iterative improvement, $\widehat{x}$ is correct up to s more significant digits with each iteration

6 Iterative Methods for Linear Systems

6.1 Diagonally Dominant Matrices

A real $(n \times n)$ matrix A is strictly row diagonally dominant if

$$|a_{ii}| > \sum_{i \neq j} |a_{ij}| \qquad \text{for } 1 \leq i \leq n$$

- The diagonal entry is strictly greater than the sum of all other entries in the row

6.2 Symmetric Positive Definite Matrices

A real $(n \times n)$ matrix A is symmetric positive definite if

- A is symmetric (i.e. $A^T = A$)
- For all $x \neq 0$, $x^T Ax > 0$

A real symmetric $(n \times n)$ matrix is positive definite if and only if all of its eigenvalues are positive

6.3 General Iterative Methods

Let M be a real $(n \times n)$ matrix and let $x^{(0)}$ be a vector in $\mathbb{R}^n$. Generate a sequence of vectors $x^{(0)}, x^{(1)}, x^{(2)}, ...$ by the iteration

$$x^{(k+1)} = Mx^{(k)} + g \qquad \text{for } k = 0, 1, 2, ...$$

where g is a given fixed vector in $\mathbb{R}^n$

- If $x^{(k)} \to x^*$ as $k \to \infty$, then $x^* = Mx^* + g$ such that x^* is a solution of $(I - M)x^* = g$

Let $|| \cdot ||$ be a vector norm on $\mathbb{R}^n$ and let $\alpha = ||M||$, the matrix norm of M induced by the vector norm $|| \cdot ||$. Suppose $\alpha < 1$. Then

- $I - M$ is invertible such that the linear system $(I - M)x = g$ has a unique solution x^*
- For any choice of $x^{(0)}$, the sequence $x^{(k)}$ generated by $x^{(k+1)} = Mx^{(k)} + g$ converges to x^*
- If $e^{(k)} = x^{(k)} - x^*$, then $||e^{(k)}|| \leq \alpha^k ||e^{(0)}||$

6.4 General Splitting Methods

Choose matrices N and P for which $A = N - P$ and consider the iteration

$$Nx^{(k+1)} = Px^{(k)} + b \qquad \text{for } k = 0, 1, 2, ...$$

The iteration converges if N and P satisfies the following properties

- N is invertible
- $Nx = b$ is easy to solve (i.e. N is a diagonal or triangular matrix)
- $||N^{-1}P|| < 1$ in some norm

Analytically, this iteration is the same as $x^{(k+1)} = Mx^{(k)} + g$ where $M = N^{-1}P$ and $g = N^{-1}b$

- The smaller the norm $||M||$ of the matrix $M = N^{-1}P$, the faster the method converges

6.5 Jacobi Method

Given a real $(n \times n)$ matrix A, let

$$L = \begin{bmatrix} 0 & \dots & \dots & 0 \\ a_{2,1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_{n,1} & \dots & a_{n,n-1} & 0 \end{bmatrix} \qquad D = \begin{bmatrix} a_{1,1} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & a_{n,n} \end{bmatrix} \qquad U = \begin{bmatrix} 0 & a_{1,2} & \dots & a_{1,n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n} \\ 0 & \dots & \dots & 0 \end{bmatrix}$$

Choose $N = D$ and $P = -(L + U)$ to get the Jacobi method:

$$Dx^{(k+1)} = -(L+U)x^{(k)} + b$$

which is equivalent to

$$x^{(k+1)}{}_i = \left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j{}^{(k)} - \sum_{j=i+1}^{n} a_{i,j} x_j{}^{(k)} \right) \div a_{i,i} \qquad \text{for } 1 \le i \le n \text{ and } k = 0, 1, 2, ...$$

- The Jacobi method is guaranteed to work with a diagonally dominant matrix

6.6 Gauss-Seidel Method

Given a real $(n \times n)$ matrix A, let

$$L = \begin{bmatrix} 0 & \dots & \dots & 0 \\ a_{2,1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_{n,1} & \dots & a_{n,n-1} & 0 \end{bmatrix} \qquad D = \begin{bmatrix} a_{1,1} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & a_{n,n} \end{bmatrix} \qquad U = \begin{bmatrix} 0 & a_{1,2} & \dots & a_{1,n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n} \\ 0 & \dots & \dots & 0 \end{bmatrix}$$

Choose $N = D + L$ and $P = -U$ to get the Gauss-Seidel method

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

which is equivalent to

$$x^{(k+1)}{}_i = \left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j{}^{(k+1)} - \sum_{j=i+1}^{n} a_{i,j} x_j{}^{(k)} \right) \div a_{i,i} \qquad \text{for } 1 \le i \le n \text{ and } k = 0, 1, 2, ...$$

- The Gauss-Seidel method is guaranteed to work with a diagonally dominant matrix
- The Gauss-Seidel method is guaranteed to work with a symmetric positive definite matrix

6.7 Successive Over-Relaxation

Given a real $(n \times n)$ matrix A, let

$$L = \begin{bmatrix} 0 & \dots & \dots & 0 \\ a_{2,1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_{n,1} & \dots & a_{n,n-1} & 0 \end{bmatrix} \qquad D = \begin{bmatrix} a_{1,1} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & a_{n,n} \end{bmatrix} \qquad U = \begin{bmatrix} 0 & a_{1,2} & \dots & a_{1,n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n} \\ 0 & \dots & \dots & 0 \end{bmatrix}$$

Rewrite the Gauss-Seidel method as

$$x^{(k+1)}{}_i = x^{(k)}{}_i + \underbrace{\left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j{}^{(k+1)} - \sum_{j=i}^{n} a_{i,j} x_j{}^{(k)} \right) \div a_{i,i}}_{\text{this is the 'correction term' from } x^{(k)} \text{ to } x^{(k+1)}} \qquad \text{for } 1 \le i \le n \text{ and } k = 0, 1, 2, \ldots$$

If the correction term gets us closer to the limit, maybe taking a multiple of it gives us better results

Given the multiple ω where $0 < \omega < 2$, the SOR iteration is

$$x^{(k+1)} = (D + \omega L)^{-1}((1 - \omega)D - \omega U)x^{(k)} + \omega(D + \omega L)^{-1} b$$

which is equivalent to

$$x^{(k+1)}{}_i = x^{(k)}{}_i + \omega \left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j{}^{(k+1)} - \sum_{j=i}^{n} a_{i,j} x_j{}^{(k)} \right) \div a_{i,i} \qquad \text{for } 1 \le i \le n \text{ and } k = 0, 1, 2, \ldots$$

- When $0 < \omega < 1$, it is called under-relaxation
  - Taking smaller steps towards the limit allows us to get very close to the limit
- When $\omega = 1$, it is equivalent to the Gauss-Seidel method
- When $1 < \omega < 2$, it is called over-relaxation
  - Taking larger steps towards the limit allows us to get close to the limit faster

6.8 Iterative Improvement Viewed as an Iterative Method

Let $\widehat{L}$ and $\widehat{U}$ be the computed LU-factorization of A. Then $\widehat{L}\widehat{U} \approx A$ such that $(\widehat{L}\widehat{U})^{-1} A \approx I$. Take the initial guess $x^{(0)} = (\widehat{L}\widehat{U})^{-1} b$. The sequence of approximations is obtained by

$$x^{(k+1)} = x^{(k)} + (\widehat{L}\widehat{U})^{-1}(b - Ax^{(k)})$$

7 Linear Least Squares

7.1 Linear Least Squares

We want to find the unknown n-vector x in $Ax = b$, given the real $(m \times n)$ matrix A where $m > n$ and known m-vector B. Since there are more equations than unknowns, there is no exact solution x. Instead, we try to find a vector $x \in \mathbb{R}^n$ that minimizes the error

$$||Ax - b||_2{}^2 = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij}x_j - b_i \right)^2$$

- Linear least squares is analogous to finding the line of best fit

7.2 Range of A

Let $Y = \{y \in \mathbb{R}^m \mid \exists x \in \mathbb{R}^n \text{ such that } Ax = y\}$ be a subspace of $\mathbb{R}^m$ and let $b \in \mathbb{R}^m$. Then there is a unique closest element $y^* \in Y$ to b in the two norm $|| \cdot ||_2$

- For all $y \in Y$, $||b - y^*||_2 \leq ||b - y||_2$
- For all $y \in Y$ where $y \neq y^*$, $||b - y^*||_2 < ||b - y||_2$

7.3 Normal Equations

Given a real $(m \times n)$ matrix A and $ab \in \mathbb{R}^m$, $x \in \mathbb{R}^n$ minimizes $||Ax - b||_2{}^2$ if and only if x is a solution of the normal equations $A^T Ax = A^T b$

- $A^T A$ is an $(n \times n)$ matrix
- x is an n-vector
- $A^T b$ is an n-vector

Remarks on the normal equations

- We can solve linear least squared problems by solving the corresponding normal equations
  - Allows us to treat least squared problems as a standard linear $(n \times n)$ system
- The normal equations are often very ill-conditioned
  - In the two norm, $\kappa(A^T A) = \kappa(A)^2$

8 Taylor Polynomials

8.1 Taylor Polynomial Approximation

Suppose $f(x)$ and its first $n+1$ derivatives are continuous on $[a,b]$ and $c \in [a,b]$. Then the n^{th} degree polynomial of f centered at c is given by

$$\begin{aligned} p_n(x) &= f(c) + \frac{f'(c)}{1!}(x-c) + \frac{f''(c)}{2!}(x-c)^2 + ... + \frac{f^{(n)}(c)}{n!}(x-c)^n \\ &= \sum_{i=0}^{n} \frac{f^{(i)}}{i!}(x-c)^i \end{aligned}$$

This polynomial is a good approximation of f for values of x close to c

8.2 Taylor's Theorem With Remainder

Suppose $f(x)$ and its first $n+1$ derivatives are continuous on $[a,b]$ and $c \in [a,b]$. Then Taylor's Theorem with Remainder states that for any $x \in [a,b]$, there exists an ξ between x and c such that

$$\begin{aligned} f(x) &= f(c) + \frac{f'(c)}{1!}(x-c) + \frac{f''(c)}{2!}(x-c)^2 + ... + \frac{f^{(n)}(c)}{n!}(x-c)^n + \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-c)^{n+1} \\ &= \underbrace{\sum_{i=0}^{n} \frac{f^{(i)}}{i!}(x-c)^i}_{\text{polynomial approximation}} + \underbrace{\frac{f^{(n+1)}(\xi)}{(n+1)!}(x-c)^{n+1}}_{\text{error value}} \end{aligned}$$

8.3 Taylor Polynomial Approximation Error

The error equation for the Taylor polynomial $p_n(x)$ centered at c is given by

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

such that the upper bound to this error equation is given by

$$||f(x) - p_n(x)||_\infty = \max_{a \le x \le b} |f(x) - p_n(x)|$$

Let $M_{n+1} = \max\limits_{a \le x \le b} |f^{(n+1)}(x)|$. Then for any $x \in [a,b]$ the point-wise upper bound for the error function $f(x) - p_n(x)$ is given by

$$|f(x) - p_n(x)| \le \frac{M_{n+1}}{(n+1)!}|(x-c)^{n+1}|$$

Let $d = \max(c-a, b-c)$. Then d is the largest distance $|x-c|$ from a point $x \in [a,b]$ to c such that

$$||f(x) - p_n(x)||_\infty = \max_{a \le x \le b} |f(x) - p_n(x)| \le \frac{M_{n+1}}{(n+1)!}|d^{n+1}|$$

9 Solution of Nonlinear Equations

9.1 Bracketing Methods

At each step of the iteration, we have an interval $[a, b]$ in which we are sure that f has a zero. If f is continuous on $[a, b]$ and $f(a) \cdot f(b) < 0$, then there exists $s \in (a, b)$ for which $f(s) = 0$

9.2 Bisection Method

Start with an interval $[a_0, b_0]$ that brackets a zero of f. At each step, shrink the length of the interval by a factor of 2, while still bracketing a zero of f

```
def f(x):
    pass # returns value of f(x)

n = <PRIORI CONDITION>

for j in range(0, n):
    c = (a + b) / 2

    if f(c) == 0 or <ITERATIVE CONDITION>:
        print(c)
        exit()

    if f(a) * f(b) < 0:
        b = c
    else:
        a = c

 c = (a + b) / 2
 print(c)
```

Suppose that we stop after n steps such that the final interval is $[a_n, b_n]$ with $f(a_n)f(b_n) < 0$. Then the final best guess for a zero of $f(x)$ is $c_n = \frac{a_n+b_n}{2}$. Given that s is a zero of f, the error of this guess is $|c_n - s| < \frac{1}{2^{n+1}}(b_0 - a_0)$

Suppose we have an error tolerance $\varepsilon > 0$ and we want to stop when $|c_n - s| < \varepsilon$. There are two ways to approach this

- Interactive Condition
  - Check the state of the program at each iteration
  - Stop the program when $\frac{b_n - a_n}{2} \leq \varepsilon$
- Priori Condition
  - Figure out beforehand how many iterations n are needed
  - Take $n \geq \frac{\log(b_0 - a_0) - \log(2\varepsilon)}{\log(2)}$

9.3 Functional Iterative Methods

At each step of the iteration, the next guess x_{n+1} is a specific function $g(x_n)$ of the previous guess

9.4 Newton's Method

Start with an approximation x_n to a zero of f. At each step, find the zero of the tangent line to the graph of f at $(x_n, f(x_n))$ to get x_{n+1}

```
def f(x):
    pass # returns value of f(x)

def df(x):
    pass # returns value of derivative of f(x)

for j in range(0, n):
    x = x - (f(x) / df(x))
```

This is equivalent to the iterative equation

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad \text{for } n = 0, 1, 2, ...$$

9.5 Secant Method

Start with two approximations x_{n-1} and x_n to a zero of f. At each step, find the zero of the secant line joining the two points $(x_{n-1}, f(x_{n-1}))$, $(x_n, f(x_n))$ to get x_{n+1}

```
def f(x):
    pass # returns value of f(x)

for j in range(0, n):
    c_x = x
    x = x - f(x) * ((x - p_x) / (f(x) - f(p_x)))
    p_x = c_x
```

This is equivalent to the iterative equation

$$x_{n+1} = x_n - f(x_n) \cdot \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \qquad \text{for } n = 0, 1, 2, ...$$

9.6 Fixed Point Iteration

If $x_{n+1} = g(x_n)$ where g is continuous and x_n converges to a number ξ in the domain of g, then $g(\xi) = \xi$

9.7 Fixed Point Theorem

Let $g(x)$ be continuous on a closed, bounded interval $I = [a, b]$ and suppose that $g(x) \in I$ for all $x \in I$. Then g has at least one fixed point ξ in I

9.8 Contraction Mapping Fixed Point Theorem

Suppose $g(x)$ is differentiable on a closed, bounded interval $I = [a, b]$ and $g(x) \in I$ for all $x \in I$. Also suppose that for some constant $L < 1$, $|g'(x)| \leq L < 1$ for all $x \in I$. Then

- g has a unique fixed point ξ in I
- Functional iteration $x_{n+1} = g(x_n)$ with any starting x_0 generates a sequence (x_n) that converges to ξ
- Given error e_n, if $e_n = x_n - \xi$, then $|e_n| \leq \dfrac{L^n}{1-L}|x_1 - x_0|$

9.9 Local Convergence Theorem

Let $g(x)$ be continuously differentiable (i.e. g and g' are continuous) on an open interval containing a fixed point ξ and suppose that $|g'(\xi)| < 1$. Then there exists $\varepsilon > 0$ such that if $|x_0 - \xi| \leq \varepsilon$, then functional iteration $x_{n+1} = g(x_n)$ is guaranteed to yield a sequence (x_n) that converges to ξ

9.10 Order of Convergence

Let (x_n) be a sequence that converges to ξ such that the error e_n is given by $e_n = x_n - \xi$. Then if there is a number $p \geq 1$ and a constant $c \neq 0$ for which

$$\lim_{n \to \infty} \frac{|e_{n+1}|}{|e_n|^p} = C$$

then p is called the order of convergence of the sequence, and C is called the asymptotic error constant

- If $p = 1$ and $C = 1$, then the convergence is sub-linear
- If $p = 1$ and $0 < C < 1$, then the convergence is linear
- If $\displaystyle\lim_{n \to \infty} \frac{|e_{n+1}|}{|e_n|} = 0$, then the convergence is super-linear
- If $p = 2$, then the convergence is quadratic
- If $p = 3$, then the convergence is cubic

9.11 Uses of Asymptotic Expressions for Error

- Indicates how fast the sequence will converge
- If p and C are known or estimated, then the asymptotic expression can be used to improve the convergence

9.12 Definition of $f \in C^k[a, b]$

If $f \in C^k[a, b]$, then $f, f', f'', ..., f^{(k)}$ are all well-defined and continuous on $[a, b]$. This is often written simply as $f \in C^k$

9.13 Order of Convergence of Fixed Point Iteration

Let $g \in C^{k+1}$, $g(s) = s$, $g'(s) = g''(s) = ... = g^{(k)}(s) = 0$, and $g^{(k+1)}(s) \neq 0$. Suppose that (x_n) is generated by $x_{n+1} = g(x_n)$ and (x_n) converges to s. Then x_n converges to s to order $k+1$ with asymptotic error constant $\frac{|g^{k+1}(s)|}{(k+1)!}$

9.14 Quadratic Convergence of Newton's Method

Suppose $f \in C^2$, $f(s) = 0$, $f'(s) \neq 0$, and (x_n) is generated by Newton's method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad \text{for } n = 0, 1, 2, ...$$

Let $e_n = x_n - s$. Then

- Whenever x_n is well-defined such that $f'(x_n) \neq 0$, and f is C^2 between x_n and s inclusive, then x_{n+1} is well-defined
- If $x_n \to s$, then the convergence is at least quadratic and

  $$\lim_{n\to\infty} \frac{e_{n+1}}{{e_n}^2} = \frac{f''(s)}{2f'(s)}$$

  unless some $e_n = 0$, in which case there exists an N such that $e_m = 0$ for all $m \geq N$
- If $|x_0 - s| \leq \delta_0$ for some constant $\delta_0 > 0$, then
  - The sequence (x_n) is well-defined and $|x_n - s| \leq \delta_0$ for all n
  - $x_n \to s$ as $n \to \infty$
  - There exists some constant c_0 such that $|e_{n+1}| \leq c_0|e_n|^2$ for all n

10 Polynomials

10.1 Polynomials and Newton's Method

Given polynomial $p(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0$, we can evaluate it at a point z as

$$p(z) = (((a_n z + a_{n-1}) \cdot z + ...) \cdot z + a_1) \cdot z + a_0$$

A recursive algorithm can be written as follows

$$\begin{aligned} {a_n}' &= a_n \\ {a_{n-1}}' &= a_{n-1} + z \cdot {a_n}' \\ &\vdots \\ p(z) &= {a_0}' \end{aligned}$$

Suppose we have evaluated $p(x)$ at a point z and we stored the intermediate values ${a_n}', {a_{n-1}}', ..., {a_1}', {a_0}'$ for that point z. Then for any x

$$p(x) = {a_0}' + (x - z)({a_n}' x^{n-1} + ... + {a_1}')$$

Let $q(x, z) = {a_n}' x^{n-1} + ... + {a_1}'$. Then $p(x) = {a_0}' + (x - z)q(x, z)$ such that

$$q(x, z) = \frac{p(x) - p(z)}{x - z}$$

Then the first derivative of $p(z)$ is given by

$$\begin{aligned} p'(z) &= \lim_{x \to z} \frac{p(x) - p(z)}{x - z} \\ &= \lim_{x \to z} q(x, z) \\ &= {a_n}' z^{n-1} + ... + {a_1}' \end{aligned}$$

This means we can use the intermediate values ${a_n}', {a_{n-1}}', ..., {a_1}', {a_0}'$ at the point z to compute $p'(z)$ as

$$p'(z) = ((a_n z + a_{n-1}) \cdot z + ...) \cdot z + a_1$$

A recursive algorithm can be written as follows

$$\begin{aligned} {a_n}'' &= {a_n}' \\ {a_{n-1}}'' &= {a_{n-1}}' + z \cdot {a_n}'' \\ &\vdots \\ p(z) &= {a_1}'' \end{aligned}$$

Then Newton's method for polynomials is defined by the iterative equation

$$\begin{aligned} x_{m+1} &= x_m - \frac{p(x_m)}{p'(x_m)} \qquad \text{for } n = 0, 1, 2, ... \\ &= x_m - \frac{{a_0}'}{{a_1}''} \qquad \text{for } n = 0, 1, 2, ... \end{aligned}$$

10.2 Polynomials

A real polynomial is a function $p : \mathbb{R} \to \mathbb{R}$ of the form

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$$

- If $a_n \neq 0$, then the degree of $p(x)$ is n
- If $p(x) \equiv 0$, $\deg(p) = -\infty$

If $p(x)$ and $q(x)$ are polynomials, then $p(x)q(x)$ is also a polynomial

- $\deg(pq) = \deg(p) + \deg(q)$

10.3 Euclidean Algorithm

Suppose $p(x)$ and $d(x)$ are polynomials and $\deg(p) \geq 0$ and $\deg(d) \geq 0$. Then there exists polynomials $q(x)$ and $r(x)$ such that

$$p(x) = q(x)d(x) + r(x)$$

where $\deg(r) < \deg(d)$

10.4 Corollary of the Euclidean Algorithm

If $\deg(p) \geq 1$ and $p(x_1) = 0$, then there exists a polynomial $q(x)$ such that

$$p(x) = q(x)(x - x_1)$$

where $\deg(q) = \deg(p) - 1$

10.5 Zeros and Multiplicity

x_1 is called a zero of p of multiplicity m if

$$p(x_1) = p'(x_1) = ... = p^{(m-1)}(x_1) = 0 \neq p^{(m)}(x_1)$$

This means that x_1 occurs m times in the factored form of p

- If x_1 is a zero of p of multiplicity m, then there exists a polynomial $q(x)$ such that $p(x) = q(x)(x - x_1)^m$ where $q(x_1) \neq 0$
- If $x_1, ..., x_k$ are zeros of p of multiplicity $m_1, ..., m_k$, then there exists a polynomial $q(x)$ such that $p(x) = q(x)(x - x_1)^{m_1} ... (x - x_k)^{m_k}$
- If $p(x)$ is a polynomial where $\deg(p) \leq n$ and $p(x)$ has at least $n + 1$ zeros (including multiplicities), then $p \equiv 0$

10.6 Interpolation

Given a table of values

x	x_0	x_1	...	x_n
y	y_0	y_1	...	y_n

a function $p(x)$ is an interpolant if $p(x_i) = y_i$ for all $i = 0, ..., n$

- $p(x)$ belongs to some given class of functions
  - Used to specify what type of interpolation is being used (i.e. linear, exponential, ...)
- Often $y_i = f(x_i)$ for some unknown function $f(x)$
- The interpolant $p(x)$ is used as an approximation to $f(x)$

10.7 Polynomial Interpolation Theorem

If $x_0, x_1, ..., x_n$ are distinct, then for arbitrary real values $y_0, y_1, ..., y_n$, there exists a unique polynomial $p(x)$ where $\deg(p) \leq n$ such that $p(x_i) = y_i$ for $i = 0, ..., n$

10.8 Lagrange Form of Polynomial Interpolation

The interpolation polynomial in Lagrange form is the linear combination

$$p(x) = \sum_{k=0}^{n} y_k \cdot \ell_k(x)$$

of Lagrange basis polynomials

$$\ell_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i}$$

for $k = 0, ..., n$ such that $\deg(\ell_k) = n$

10.9 Undetermined Coefficients Form of Polynomial Interpolation

Suppose the interpolant is the polynomial $p(x) = a_0 + a_1 x + ... + a_n x^n$ where $\deg(p) \leq n$. We want to determine the unknown coefficients $a_0, a_1, ..., a_n$. The table of values gives us $n + 1$ equations of the form $p(x_i) = y_i$ for $i = 0, ..., n$, which can be expressed as the system of equations

$$\underbrace{\begin{bmatrix} y_0 \\ \vdots \\ y_n \end{bmatrix}}_{y} = \begin{bmatrix} p(x_0) \\ \vdots \\ p(x_n) \end{bmatrix} = \begin{bmatrix} a_0 + a_1 x_0 + ... + a_n {x_0}^n \\ \vdots \\ a_0 + a_1 x_n + ... + a_n {x_n}^n \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & x_0 & ... & {x_0}^n \\ \vdots & \vdots & & \vdots \\ 1 & x_n & ... & {x_n}^n \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix}}_{a}$$

The system of equations can be written as $Ma = y$

- The matrix M is called a Vandermonde matrix where $\det(M) = \prod_{i=0}^{n-1} \prod_{j=i+1}^{n} (x_j - x_i)$

10.10 Newton's Form of Polynomial Interpolation

Let $a_k = \dfrac{y_k - p_{k-1}(x_k)}{\prod_{i=0}^{k-1}(x_k - x_i)}$. Then Newton's form of polynomial interpolation is given by

$$p_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + ... + a_n(x - x_0)...(x - x_{n-1})$$

An inductive algorithm can be written as follows

$$\begin{aligned}
p_0(x) &= a_0 \\
p_1(x) &= p_0 + a_1(x - x_0) \\
&\vdots \\
p_k(x) &= p_{k-1}(x) + a_k(x - x_0)...(x - x_{k-1}) \\
&\vdots \\
p_n(x) &= p_{n-1}(x) + a_k(x - x_0)...(x - x_{n-1})
\end{aligned}$$

where $\deg(p_k) \le k$ for $k = 1, ..., n$ and $p_k(x_i) = y_i$ for $i = 0, ..., k$

- If $x_i = 0$ for all i, then $p(x)$ is the usual power form
  - $p(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n$
- If $x_i = c$ for all i, then $p(x)$ is a shifted power form
  - $p(x) = a_0 + a_1(x - c) + a_2(x - c)^2 + ... + a_n(x - c)^n$ is the Taylor expansion centered at c
  - $a_k = \frac{p^{(k)}(c)}{k!}$

10.11 Nested Multiplication for Newton's Form of Polynomial Interpolation

Given Newton's form $p(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + ... + a_n(x - x_0)...(x - x_{n-1})$, we can evaluate it at a point z as

$$p(z) = a_0 + (z - x_0)(a_1 + (z - c_2)(a_2 + ...(a_{n-1} + (z - c_n)a_n)))$$

A recursive algorithm can be written as follows

$$\begin{aligned}
{a_n}' &= a_n \\
{a_{n-1}}' &= a_{n-1} + (z - c_n)a_n \\
{a_{n-2}}' &= a_{n-2} + (z - c_{n-1}){a_{n-1}}' \\
&\vdots \\
p(z) &= {a_0}'
\end{aligned}$$

Suppose we have evaluated $p(x)$ at a point z and we stored the intermediate values ${a_0}', {a_1}', ..., {a_{n-1}}', {a_n}'$ for that point z. Then for any x

$$p(x) = {a_0}' + {a_1}'(x - z) + {a_2}'(x - z)(x - x_1) + ... + a_n(x - z)(x - x_1)...(x - x_{n-1})$$

10.12 Divided Difference

Suppose $x_0, x_1, ..., x_n$ are distinct and $f(x_0), f(x_1), ..., f(x_k)$ are given. The k^{th} order divided difference of f, denoted $f[x_0, x_1, ..., x_k]$, is the coefficient of x^k in the unique polynomial $p_k(x)$ which interpolates f at $x_0, x_1, ..., x_k$

$$p_k(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + ...f[x_0, ..., x_k](x - x_0)...(x - x_{k-1})$$

We can determine the divided differences of f with the recursive algorithm

$$f[x_0] = f(x_0)$$
$$f[x_0, x_1, ..., x_k] = \frac{f[x_1, x_2, ..., x_k] - f[x_0, x_1, ..., x_{k-1}]}{x_k - x_0} \qquad \text{for } k \geq 1$$

- Changing the order of arguments does not change the divided difference
  - i.e. $f[x_0, x_1] = f[x_1, x_0]$
- The 0^{th} order divided difference $f[x_i]$ is always equal to $f(x_i)$

10.13 Divided Difference Table

Given distinct $x_0, x_1, ..., x_n$ and $f(x_0), f(x_1), ..., f(x_n)$, we can compute the following divided difference table using $f[x_i] = f(x_i)$ and the recursive formula above

$$\begin{array}{c||ccccccc}
x_0 & f[x_0] & & & & & & \\
 & & \searrow & f[x_0, x_1] & & & & \\
x_1 & f[x_1] & \nearrow\searrow & & \searrow & f[x_0, x_1, x_2] & & \\
 & & & f[x_1, x_2] & \nearrow & & \searrow & f[x_0, x_1, x_2, x_3] \\
x_2 & f[x_2] & \nearrow\searrow & & \searrow & f[x_1, x_2, x_3] & \nearrow & \\
 & & & f[x_2, x_3] & \nearrow & & & \\
x_3 & f[x_3] & \nearrow & & & & &
\end{array}$$

11 Approximation Theory and Interpolation

11.1 Approximation Theory

Let the true unknown function $f(x)$ be defined on $[a, b]$. Suppose that we know the value of f at a finite number of points/nodes x. We want to find a function $g(x)$ such that

- $g(x)$ is easy to compute for all $x \in [a, b]$
- $g(x)$ closely approximates $f(x)$
  - We can measure this closeness using norms
    - $||f - g||_1 = \displaystyle\int_a^b |f(x) - g(x)| \, dx$
    - $||f - g||_2 = \left(\displaystyle\int_a^b |f(x) - g(x)|^2 \, dx\right)^{\frac{1}{2}}$
    - $||f - g||_\infty = \displaystyle\max_{a \le x \le b} |f(x) - g(x)|$

Approximation theory notation

- $\mathcal{P}_n$ is the set of all real polynomials of degree at most n
- $C[a, b]$ is the set of all continuous real-valued functions on $[a, b]$

11.2 Weierstrass Approximation Theorem

Given any function $f \in C[a, b]$ and any $\varepsilon > 0$, there exists a polynomial p such that $||f - p||_\infty < \varepsilon$

- This means that polynomials can approximate arbitrary continuous functions on a closed, bounded interval to an arbitrarily small tolerance

11.3 Best Approximation Theorem

Given $f \in C[a, b]$ and an integer $n \ge 0$, there exists a unique polynomial $p^* \in \mathcal{P}_n$ for which

$$||f - p^*||_\infty \le ||f - p||_\infty$$

for all $p \in \mathcal{P}_n$

- This means that the best polynomial approximation to an arbitrary continuous functions on a closed, bounded interval is unique

11.4 $p_n(x)$ Approximation With Remainder

Suppose f has k continuous derivatives and $x_0, ..., x_k$ are distinct. Then there exists ξ between $\min(x_i)$ and $\max(x_i)$ such that

$$f[x_0, ..., x_k] = \frac{f^{(k)}(\xi)}{k!}$$

Let $x_0, ..., x_n$ be distinct, and $x \neq x_i$. Then

$$\begin{aligned} f(x) = f[x_0] + f[x_0, x_1](x - x_0) + ...+ \\ f[x_0, ..., x_n](x - x_0)...(x - x_{n-1}) + f[x_0, ..., x_n, x](x - x_0)...(x - x_n) \end{aligned}$$

Suppose $f \in C^{n+1}[a, b]$ and $x_0, ..., x_n$ are $n + 1$ distinct points in $[a, b]$. Then for each $x \in [a, b]$, there exists $\xi \in [a, b]$ such that

$$\begin{aligned} f(x) &= f[x_0] + f[x_0, x_1](x - x_0) + ...+ \\ &\underbrace{\vdots \qquad f[x_0, ..., x_n](x - x_0)...(x - x_{n-1})}_{\text{polynomial approximation}} + \underbrace{\frac{f^{(n+1)}(\xi)}{(n + 1)!}(x - x_0)...(x - x_n)}_{\text{error value}} \\ &= \underbrace{p_n(x)}_{\substack{\text{polynomial}\\ \text{approximation}}} + \underbrace{\frac{f^{(n+1)}(\xi)}{(n + 1)!}(x - x_0)...(x - x_n)}_{\text{error value}} \end{aligned}$$

11.5 $p_n(x)$ Approximation Error

The error equation for the polynomial $p_n(x)$ is given by

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n + 1)!}(x - x_0)...(x - x_n)$$

Let $M_{n+1} = \max\limits_{x\in[a,b]} \left|f^{(n+1)}(x)\right|$ and $W(x) = (x - x_0)...(x - x_n)$. Then the upper bound to this error equation is given by

$$\begin{aligned} &|f(x) - p_n(x)| \leq \frac{M_{n+1}}{(n + 1)!}|W(x)| \\ &||f(x) - p_n(x)||_\infty \leq \frac{M_{n+1}}{(n + 1)!}||W(x)||_\infty \end{aligned}$$

where $M_{n+1} = \left|\left|f^{(n+1)}(x)\right|\right|_\infty = \max\limits_{x\in[a,b]} \left|f^{(n+1)}(x)\right|$ and $||W(x)||_\infty = \max\limits_{x\in[a,b]} |W(x)|$

11.6 Chebyshev Polynomials of the First Kind

Let $T_k(x) = \cos\left(k \cos^{-1} x\right)$ be the Chebyshev polynomial on $[-1, 1]$ for $k = 0, 1, 2, ...$ and $x \in [-1, 1]$. Then

$$T_0(x) = \cos(0 \cdot \cos^{-1} x) = 1$$
$$T_1(x) = \cos(1 \cdot \cos^{-1} x) = x$$

Substituting $x = \cos\theta$, we get the recursive formula

$$T_{k+1}(x) = 2x \cdot T_k(x) - T_{k-1}(x)$$

For $k \geq 1$, the k distinct zeros of $T_k(x)$ in the interval $[-1, 1]$ are described by

$$x_j = \cos\left(\frac{j + \frac{1}{2}}{k}\pi\right) \qquad \text{for } j = 0, ..., k-1$$

Given x_j for $j = 0, ..., k-1$, $T_k(x)$ can be factorized as

$$T_k(x) = 2^{k-1}(x - x_0)(x - x_1)...(x - x_{k-1})$$

For $k \geq 1$, let $y_j = \cos\left(\frac{j}{k}\pi\right)$ for $j = 0, ..., k$. Then

$$T_k(y_j) = \cos\left[k\left(\frac{j}{k}\pi\right)\right] = (-1)^j$$

11.7 Minimizing $||W(x)||_\infty$ With Chebyshev Polynomials on $[-1, 1]$

Given distinct points $x_0, ..., x_n \in [-1, 1]$ and $W(x) = (x - x_0)...(x - x_n)$, let x_j for $j = 0, ..., n$ be the zeros / Chebyshev nodes of $T_{n+1}(x)$. Then

$$x_j = \cos\left(\frac{j + \frac{1}{2}}{n+1}\pi\right) \qquad \text{for } j = 0, ..., n$$

Then $W(x) = \frac{1}{2^n} \cdot T_{n+1}(x)$ and $||W(x)||_\infty = \frac{1}{2^n}$ such that

$$||f(x) - p_n(x)||_\infty \leq \frac{M_{n+1}}{(n+1)!}||W(x)||_\infty \leq \frac{M_{n+1}}{(n+1)!} \cdot \frac{1}{2^n}$$

11.8 Minimizing $||W(x)||_\infty$ With Chebyshev Polynomials on $[a, b]$

Let t be the variable in $[-1, 1]$ and x be the variable in $[a, b]$ where

$$x = \frac{b-a}{2}t + \frac{a+b}{2}$$
$$t = 2 \cdot \frac{x-a}{b-a} - 1$$

Let $\widetilde{T}_k(x)$ for $k = 0, 1, 2, ...$ and $x \in [a, b]$ be the shifted Chebyshev polynomial on $[a, b]$. Then

$$\widetilde{T}_k(x) = T_k(t) = T_k\left(2 \cdot \frac{x-a}{b-a} - 1\right)$$

Given distinct points $x_0, ..., x_n \in [-1, 1]$ and $W(x) = (x - x_0)...(x - x_n)$, let t_j for $j = 0, ..., n$ be the zeros / Chebyshev nodes of $T_{n+1}(x)$. Then

$$t_j = \cos\left(\frac{j + \frac{1}{2}}{n+1}\pi\right) \qquad \text{for } j = 0, ..., n$$

Let x_j for $j = 0, ..., n$ be the zeros / Chebyshev nodes of $\widetilde{T}_{n+1}(x)$. Then

$$x_j = \frac{b-a}{2}\cos\left(\frac{j + \frac{1}{2}}{n+1}\pi\right) + \frac{a+b}{2}$$

Then $W(x) = \frac{1}{2^n}\left(\frac{b-a}{2}\right)^{n+1}\widetilde{T}_{n+1}(x)$ and $||W(x)||_\infty = \frac{1}{2^n}\left(\frac{b-a}{2}\right)^{n+1}$ such that

$$||f(x) - p_n(x)||_\infty \leq \frac{M_{n+1}}{(n+1)!}||W(x)||_\infty \leq \frac{M_{n+1}}{(n+1)!} \cdot \frac{1}{2^n}\left(\frac{b-a}{2}\right)^{n+1}$$

11.9 Interpolation at Equally Spaced Points

Suppose $f(x)$ is defined on $[a, b]$ and n is a positive integer. Let $h = \frac{b-a}{n}$ and $x_i = a + ih$ for $i = 0, ..., n$. Let us define the forward difference of f as

$$\begin{aligned}
\Delta f(x) &= f(x+h) - f(x) \\
\Delta^2 f(x) &= \Delta(\Delta f(x)) \\
&= \Delta f(x+h) - \Delta f(x) \\
&= f(x+2h) - 2f(x+h) + f(x) \\
&\quad \vdots \\
\Delta^k f(x) &= \Delta^{k-1} f(x+h) - \Delta^{k-1} f(x)
\end{aligned}$$

By induction, we can show that

$$\Delta^k f(x) = \sum_{j=0}^{k} \binom{k}{j} (-1)^{k-j} f(x+jh)$$

11.10 Forward Differences and Divided Differences

The forward difference $\Delta^k f(x)$ represents the numerator of the divided difference $f[x_i, ..., x_{i+k}]$

$$f[x_i, ..., x_{i+k}] = \frac{\Delta^k f(x)}{k! \cdot h^k}$$

11.11 Osculatory Interpolation

Let $x_0, ..., x_n$ be not necessarily distinct points in $[a, b]$. Let k_α be the number of times each distinct point α appears in $x_0, ..., x_n$. If a polynomial $p(x)$ interpolates $f(x)$ at $x_0, ..., x_n$, then

$$p^{(j)}(\alpha) = f^{(j)}(\alpha) \qquad \text{for } j = 0, ..., k_\alpha - 1$$

11.12 Osculatory Interpolation Theorem

Let $x_0, ..., x_n$ be not necessarily distinct points in $[a, b]$ and suppose for each distinct point α in $x_0, ..., x_n$, $f^{(j)}(\alpha)$ is defined for $j = 0, ..., k_\alpha - 1$. Then there exists a unique polynomial $p_n(x)$ where $\deg(p_n) \leq n$ which interpolates $f(x)$ at $x_0, ..., x_n$

- $f[x_0, ..., x_n]$ is defined to be the coefficient of x^n in the unique polynomial $p_n(x)$ which interpolates f at $x_0, ..., x_n$
- If $x_0 \neq x_k$, the recursive definition of $f[x_0, ..., x_n]$ still holds
- If $f \in C^k$, then $f[x_0, ..., x_k]$ is a continuous function of the $k+1$ variables $x_0, ..., x_k$
- $f[\underbrace{c, c, ..., c}_{(k+1) \text{ times}}] = \frac{f^{(k)}(c)}{k!}$

11.13 Piecewise Polynomial Functions

A piecewise-polynomial function of order k on $[a, b]$ with interior breakpoints at $x_1, ..., x_{n-1}$ is a function of the form

$$S(x) = \begin{cases} S_0(x) & \text{if } x \in [x_0, x_1) \\ \vdots & \\ S_j(x) & \text{if } x \in [x_j, x_{j+1}) \\ \vdots & \\ S_{n-1}(x) & \text{if } x \in [x_{n-1}, x_n] \end{cases}$$

where $S_j(x)$ is a polynomial of degree at most k

$$S_j(x) = c_{0j} + c_{1j}x + ... + c_{kj}x^k$$

For $0 \leq m \leq k$, define $\mathcal{PP}_k^m$ to be the set of all piecewise-polynomial functions of order k which are in $C^m[a, b]$

11.14 Piecewise Polynomial Function Terminology

- The endpoints and interior breakpoints $x_0, x_1, ..., x_n$ are called knots
- Elements of $\mathcal{PP}_k^{k-1}$ are called splines of order k
  - Splines are the smoothest (have the most continuous derivatives) piecewise polynomials

11.15 Continuity Conditions on Piecewise-Polynomial Functions

Let S be a piecewise-polynomial function of order k. For S to be in C^m, there are $m+1$ conditions which must be satisfied at each of the $n-1$ interior breakpoints x_j for $j = 1, ..., n-1$

$$S_{j-1}{}^{(\nu)}(x_j) = S_j{}^{(\nu)}(x_j) \qquad \text{for } \nu = 0, ..., m$$

- These conditions form a system of $(m+1)(n-1)$ equations which the $(k+1)n$ coefficients must satisfy
- The dimension of the vector space $\mathcal{PP}_k^m$ when $m \leq k$ is

$$\dim\left(\mathcal{PP}_k^m\right) = (k+1)n - (m+1)(n-1) = (k-m)n + m + 1$$

  - The dimension of the vector space $\mathcal{PP}_k^m$ tells us how many free variables are in S

11.16 Piecewise-Linear Interpolation

Let S be a piecewise-linear polynomial function in $\mathcal{PP}_1^0$ that interpolates $f(x)$ at $x_0, ..., x_n$. Then S must satisfy the following $n+1$ conditions

$$S(x_j) = f(x_j) \qquad \text{for } j = 0, ..., n$$

Let S_j be the j^{th} sub-function in S where $j = 0, ..., n-1$. Then S_j is the polynomial of degree at most 1 that interpolates f at x_j and x_{j+1} and is defined as

$$S_j(x) = f(x_j) + f[x_j, x_{j+1}](x - x_j)$$

The error equation for the piecewise-linear polynomial function is given by

$$||f(x) - S(x)||_\infty \leq \frac{M_2}{8} \cdot h^2$$

where $h = \max\limits_{0 \leq j \leq n-1}(x_{j+1} - x_j)$ and $M_2 = ||f''(x)||_\infty$

11.17 Piecewise-Cubic Hermite Interpolation

Let S be a piecewise-cubic Hermite polynomial function in $\mathcal{PP}_3^1$ that interpolates $f(x)$ at $x_0, ..., x_n$. Then S must satisfy the following $2n+2$ conditions

$$\begin{aligned} S(x_j) &= f(x_j) \\ S'(x_j) &= f'(x_j) \end{aligned} \qquad \text{for } j = 0, ..., n$$

Let S_j be the j^{th} sub-function in S where $j = 0, ..., n-1$. Then S_j is the polynomial of degree at most 3 that interpolates f at x_j and x_{j+1} and satisfies

$$\begin{aligned} &S_j(x_j) = f(x_j) \\ &S_j(x_{j+1}) = S_{j+1}(x_{j+1}) = f(x_{j+1}) \\ &{S_j}'(x_j) = f'(x_j) \\ &{S_j}'(x_{j+1}) = {S_{j+1}}'(x_{j+1}) = f'(x_{j+1}) \end{aligned}$$

The error equation for the piecewise-cubic Hermite polynomial function is given by

$$||f(x) - S(x)||_\infty \leq \frac{M_4}{384} \cdot h^4$$

where $h = \max\limits_{0 \leq j \leq n-1}(x_{j+1} - x_j)$ and $M_4 = \left|\left|f^{(4)}(x)\right|\right|_\infty$

11.18 Natural Cubic Spline Interpolation

Let S be a natural piecewise-cubic polynomial function in $\mathcal{PP}_3^2$ that interpolates $f(x)$ at $x_0, ..., x_n$. Then S must satisfy the following $n+3$ conditions

$$\begin{aligned} &S(x_j) = f(x_j) \\ &S''(x_0) = 0 \qquad\qquad \text{for } j = 0, ..., n \\ &S''(x_n) = 0 \end{aligned}$$

Let S_j be the j^{th} sub-function in S where $j = 0, ..., n-1$. Then S_j is the polynomial of degree at most 3 that interpolates f at x_j and x_{j+1} and satisfies

$$\begin{aligned} &S_j(x_j) = f(x_j) \\ &S_j(x_{j+1}) = S_{j+1}(x_{j+1}) = f(x_{j+1}) \\ &S_j''(x_j) = y_j'' \\ &S_j''(x_{j+1}) = S_{j+1}''(x_{j+1}) = y_{j+1}'' \end{aligned}$$

where $y_0'', ..., y_n''$ are solutions to the system of equations

$$\begin{bmatrix} \gamma_1 & h_1 & 0 & \dots & \dots & \dots & 0 \\ h_1 & \gamma_2 & h_2 & \ddots & & & \vdots \\ 0 & h_2 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & h_{n-3} & 0 \\ \vdots & & & \ddots & h_{n-3} & \gamma_{n-2} & h_{n-2} \\ 0 & \dots & \dots & \dots & 0 & h_{n-2} & \gamma_{n-1} \end{bmatrix} \begin{bmatrix} y_1'' \\ y_2'' \\ y_3'' \\ \vdots \\ y_{n-3}'' \\ y_{n-2}'' \\ y_{n-1}'' \end{bmatrix} = \begin{bmatrix} b_1 - h_0 y_0'' \\ b_2 \\ b_3 \\ \vdots \\ b_{n-3} \\ b_{n-2} \\ b_{n-1} - h_{n-1} y_n'' \end{bmatrix}$$

where $h_j = x_{j+1} - x_j$, $\gamma_j = 2(x_j + x_{j-1})$, and $b_j = 6\left(\frac{f(x_{j+1}) - f(x_j)}{h_j} - \frac{f(x_j) - f(x_{j-1})}{h_{j-1}}\right)$

The error equation for the natural piecewise-cubic polynomial function is given by

$$\begin{aligned} &||f - S||_\infty \le C_0 ||f''||_\infty \cdot h^2 \\ &||f' - S'||_\infty \le C_1 ||f''||_\infty \cdot h \end{aligned}$$

where $h = \max\limits_{0 \le j \le n-1} (x_{j+1} - x_j)$ and C_0, C_1 are constants independent of f and h

12 Numerical Integration

12.1 Terminology

- Given a known and fixed interval $[a, b]$, let $I(f) : C[a, b] \to \mathbb{R}$ be defined as $I(f) = \int_a^b f(x)\, dx$
  - The domain of $I(f)$ is a set of functions
  - If we want to make the interval explicit, we can write $I_a^b(f)$
- A numerical integration formula / quadrature formula is any formula which approximates $I(f)$ using values of f

12.2 Interpolatory Quadrature

Let $a \leq x_0 < x_1 < ... < x_n \leq b$ be all fixed, and let $Q_n(f)$ be the interpolatory quadrature given by $Q_n(f) \equiv I(p_n)$ where $p_n(x)$ is the unique polynomial of degree $\leq n$ which interpolates f at $x_0, ..., x_n$

$$Q_n(f) = \sum_{j=0}^{n} A_j f(x_j) \text{ where } A_j = \int_a^b \ell_j(x)\, dx$$

- $\ell_j(x)$ is a Lagrange basis polynomial
- An interpolatory quadrature is a weighted sum of the function values $f(x_0), ..., f(x_n)$
- The weights are the terms $A_0, ..., A_n$
- The nodes are the terms $x_0, ..., x_n$

12.3 Precision of Interpolatory Quadrature

A quadrature formula Q on $[a, b]$ has precision at least k if $Q(p) = I(p)$ for all $p \in \mathcal{P}_k$

- Every $(n + 1)$-point interpolatory quadrature has precision at least n

12.4 Interpolatory Quadrature by Undetermined Coefficients

Let Q_n be the $(n + 1)$-point interpolatory quadrature on $[a, b]$ with nodes $x_0, ..., x_n$. Let $f_k(x) = x^k$ for $k = 0, ..., n$. Since Q_n has precision at least n, $Q_n(f_k) = I(f_k)$ for $k = 0, ..., n$. This gives us an $(n + 1) \times (n + 1)$ linear system for the weights

$$\sum_{j=0}^{n} {x_j}^k \cdot A_j = \int_a^b x^k\, dx$$

for $k = 0, ..., n$

12.5 Newton-Cotes Formulas on $[-1, 1]$

The closed Newton-Cotes formulas are obtained using interpolatory quadrature with equally spaced nodes $x_0, ..., x_n$ with $x_j = a + jh$ where $h = \frac{b-a}{n}$ for $j = 0, ..., n$

- Closed Newton-Cotes formulas use the end-points such that $x_0 = a$ and $x_n = b$

The open Newton-Cotes formulas are obtained using interpolatory quadrature with equally spaced nodes $x_1', ..., x_{n+1}'$ with $x_j' = a + jh$ where $h = \frac{b-a}{n+2}$ for $j = 1, ..., n+1$

- Open Newton-Cotes formulas do not use the end-points such that $a < x_1'$ and $x_{n+1}' < b$

12.6 Newton-Cotes Formulas on $[a, b]$

Let t_j be the variable in the closed Newton-Cotes formula on $[-1, 1]$. Then the variable x_j in $[a, b]$ is given by

$$x_j = \frac{b-a}{2} t_j + \frac{a+b}{2}$$

for $j = 0, ..., n$

$$\int_a^b f(x)\, dx = \int_{-1}^1 f\left(\frac{b-a}{2} t + \frac{a+b}{2}\right) \cdot \frac{b-a}{2}\, dt$$

Let u_j be the variable in the open Newton-Cotes formula on $[-1, 1]$. Then the variable x_j' in $[a, b]$ is given by

$$x_j' = \frac{b-a}{2} u_j + \frac{a+b}{2}$$

for $j = 1, ..., n+1$

$$\int_a^b f(x)\, dx = \int_{-1}^1 f\left(\frac{b-a}{2} u + \frac{a+b}{2}\right) \cdot \frac{b-a}{2}\, du$$

12.7 Interpolatory Quadrature Approximation Error

Let $Q_n \equiv I(p_n)$ be the $(n+1)$-point interpolatory quadrature on $[a, b]$ with nodes $x_0, ..., x_n$. Let $f \in C^{n+1}[a, b]$ and p_n be the unique polynomial of degree $\leq n$ which interpolates f at $x_0, ..., x_n$

The error $e_n(f)$ for the interpolatory quadrature $I(p_n)$ is given by

$$e_n(f) = I(f) - I(p_n) = \int_a^b f(x) - p_n(x)\, dx = \int_a^b \frac{f^{(n+1)}(\xi_x)}{(n+1)!} W(x)\, dx$$

Let $M_{n+1} = \max\limits_{a \leq x \leq b} \left| f^{(n+1)}(x) \right|$ and $W(x) = (x - x_0)...(x - x_n)$. Then the upper bound to this error equation is given by

$$|e_n(f)| \leq \frac{M_{n+1}}{(n+1)!} \int_a^b |W(x)|\, dx$$

12.8 Examples of Newton-Cotes Formulas on $[-1, 1]$

Closed Newton-Cotes formulas

- Trapezoid Rule ($n = 1$)

  $Q_1(f) = f(-1) + f(1)$

- Simpson's Rule ($n = 2$)

  $Q_2(f) = \frac{1}{3}f(-1) + \frac{4}{3}f(0) + \frac{1}{3}f(1)$

- Simpson's Rule ($n = 3$)

  $Q_3(f) = \frac{1}{4}f(-1) + \frac{3}{4}f\left(-\frac{1}{3}\right) + \frac{3}{4}f\left(\frac{1}{3}\right) + \frac{1}{4}f(1)$

Open Newton-Cotes formulas

- Midpoint Rule ($n = 0$)

  $Q_0(f) = 2f(0)$

- Midpoint Rule ($n = 1$)

  $Q_1(f) = f\left(-\frac{1}{3}\right) + f\left(\frac{1}{3}\right)$

- Midpoint Rule ($n = 2$)

  $Q_2(f) = \frac{4}{3}f\left(-\frac{1}{2}\right) - \frac{2}{3}f(0) + \frac{4}{3}f\left(\frac{1}{2}\right)$

12.9 Examples of Newton-Cotes Formulas on $[a, b]$

Closed Newton-Cotes formulas

- Trapezoid Rule ($n = 1$)

  $Q_1(f) = \frac{b-a}{2}(f(a) + f(b))$

- Simpson's Rule ($n = 2$)

  $Q_2(f) = \frac{b-a}{2}\left(\frac{1}{3}f(a) + \frac{4}{3}f\left(\frac{a+b}{2}\right) + \frac{1}{3}f(b)\right)$

Open Newton-Cotes formulas

- Midpoint Rule ($n = 0$)

  $Q_0(f) = \frac{b-a}{2}\left(2f\left(\frac{a+b}{2}\right)\right)$

12.10 Examples of Newton-Cotes Formulas on N Subintervals of $[a, b]$

Closed Newton-Cotes formulas

- Trapezoid Rule ($n = 1$)

  $$T_{x_j}^{x_{j+1}}(f) = \frac{h_j}{2}(f(x_j) + f(x_{j+1}))$$

- Simpson's Rule ($n = 2$)

  $$S_{x_j}^{x_{j+1}}(f) = \frac{h_j}{2}\left(\frac{1}{3}f(x_j) + \frac{4}{3}f\left(\frac{x_j + x_{j+1}}{2}\right) + \frac{1}{3}f(x_{j+1})\right)$$

Open Newton-Cotes formulas

- Midpoint Rule ($n = 0$)

  $$M_{x_j}^{x_{j+1}}(f) = \frac{h_j}{2}\left(2f\left(\frac{x_j + x_{j+1}}{2}\right)\right)$$

where $h_j = x_{j+1} - x_j$ and $j = 0, ..., N - 1$

12.11 Composite Closed Newton-Cotes Formulas

Partition $[a, b]$ into N subintervals by choosing $x_0, ..., x_N$ with $a = x_0 < ... < x_N = b$ and let $h_j = x_{j+1} - x_j$

Composite Trapezoid Rule

$$T_N(f) = \sum_{j=0}^{N-1} \frac{h_j}{2}(f(x_j) + f(x_{j+1}))$$

Composite Simpson's Rule

$$S_N(f) = \sum_{j=0}^{N-1} \frac{h_j}{6}\left(f(x_j) + 4f\left(\frac{x_j + x_{j+1}}{2}\right) + f(x_{j+1})\right)$$

12.12 Composite Closed Newton-Cotes Formulas With Equally Spaced Points

Let $h = \frac{b-a}{N}$ and $x_j = a + jh$ for $j = 0, ..., N$

Composite Trapezoid Rule

$$T_N(f) = \frac{h}{2}(f(x_0) + f(x_N)) + h \cdot \sum_{j=1}^{N-1} f(x_j)$$

Composite Simpson's Rule

$$S_N(f) = \frac{h}{6}(f(x_0) + f(x_N)) + \frac{h}{3}\left(\sum_{j=1}^{N-1} f(x_j) + 2 \cdot \sum_{j=0}^{N-1} f\left(\frac{x_j + x_{j+1}}{2}\right)\right)$$

12.13 Composite Trapezoid Rule Approximation Error

The error ${e_N}^T(f)$ for the composite Trapezoid Rule with equally spaced points is given by

$$ {e_N}^T(f) = -\frac{f''(\eta)}{3}\left(\frac{h}{2}\right)^2(b-a) $$

for some $\eta \in [a, b]$

12.14 Composite Simpson's Rule Approximation Error

The error ${e_N}^S(f)$ for the composite Simpson's Rule with equally spaced points is given by

$$ {e_N}^S(f) = -\frac{f^{(4)}(\eta)}{180}\left(\frac{h}{2}\right)^4(b-a) $$

for some $\eta \in [a, b]$

12.15 Richardson Extrapolation

Suppose an unknown quantity a_0 is given by

$$ a_0 = A(h) + a_1 h^{k_1} + ... + a_m h^{k_m} + C_m(h) h^{k_{m+1}} $$

where $k_1 < ... < k_{m+1}$ are known, $a_1, ..., a_m$ are unknown, and $C_m(h)$ is bounded and unknown

Let $A_0(h) = A(h)$. Then by Richardson extrapolation we get

$$ \begin{aligned} A_1(h) &= \frac{A_0(rh) - r^{k_1} A_0(h)}{1 - r^{k_1}} \\ A_2(h) &= \frac{A_1(rh) - r^{k_2} A_1(h)}{1 - r^{k_2}} \\ &\vdots \\ A_m(h) &= \frac{A_i(rh) - r^{k_m} A_i(h)}{1 - r^{k_m}} \end{aligned} $$

where r is some constant that satisfies $0 < r < 1$ (usually we choose $r = \frac{1}{2}$)

12.16 Romberg Integration

Given $f \in C^{\nu}[a,b]$ and N equally spaced points, then

$$I(f) = T_N(f) + c_2 h^2 + c_4 h^4 + ... + C_{\nu}(h) h^{\nu}$$

where $h = \frac{b-a}{N}$, $c_2, c_4, ...$ are unknown, and $C_{\nu}(h)$ is bounded and unknown

Let us define $T_{0,m} = T_{2^m}(f)$ for $m = 0, 1, 2, ...$ Then

$$T_{1,m} = \frac{T_{0,m+1} - \left(\frac{1}{4}\right)^1 T_{0,m}}{1 - \left(\frac{1}{4}\right)^1}$$

$$\vdots$$

$$T_{i,m} = \frac{T_{i-1,m+1} - \left(\frac{1}{4}\right)^i T_{i-1,m}}{1 - \left(\frac{1}{4}\right)^i}$$

12.17 Romberg Table

We can compute the following Romberg table using $T_{0,m} = T_{2^m}(f)$ and the recursive formula above

$$\begin{array}{cccc}
T_{0,0} & & & \\
 & T_{1,0} & & \\
T_{0,1} & & T_{2,0} & \\
 & T_{1,1} & & T_{3,0} \\
T_{0,2} & & T_{2,1} & \\
 & T_{1,2} & & \\
T_{0,3} & & &
\end{array}$$