Revision notes - MA2213

Ma Hongqiang

November 7, 2017

Contents

1 Computer Arithmetic and Computational Errors **2**

2 Numerical Solution of Linear Systems of Equations **9**

3 Error of Approximation **18**

4 Numerical Integration **27**

5 Solutions of Nonlinear Equations **35**

1 Computer Arithmetic and Computational Errors

1.1 Number Systems

Definition 1.1 (Decimal System)**.**
The number system we are used to is the **decimal system** and the number "10" which plays an important role is called the **base** of the decimal system.

In general, we can take other positive integer $N > 1$ as a base. In particular, $N = 2, 8, 16$ are used in most digital computers and the systems with $N = 2, 8, 16$ are known as **binary**, **octal**, **hexadecimal** number systems respectively.

1.1.1 Decimal to Binary

The conversion of decimal number to binary is performed in two steps.
Take $(53.7)_{10}$ as an example.

1. **Integer Part**. Convert decimal integers to binary by deviding by 2 successively and recording the remainders.
   The remainders, 0 or 1, are recorded by starting at the decimal point and moving away to the left. For$(53)_{10}$, we have

$$\begin{aligned} 53/2 &= 26\,\mathbf{R}\,1 \\ 26/2 &= 13\,\mathbf{R}\,0 \\ 13/2 &= 6\ \ \mathbf{R}\,1 \\ 6/2 &= 3\ \ \mathbf{R}\,0 \\ 3/2 &= 1\ \ \mathbf{R}\,1 \\ 1/2 &= 0\ \ \mathbf{R}\,1 \end{aligned}$$

   Therefore, the base 10 number 53 can be written as binary

$$(110101)_2 = 2^5 + 2^4 + 2^2 + 2^0 = 53$$

2. **Fractional Part**. Convert $(0.7)_{10}$ to binary as follows:
   Multiply by 2 successfully and record the integer parts, moving away from the decimal point to the right:

$$\begin{aligned} .7 \times 2 &= .4\ \ +1 \\ .4 \times 2 &= .8\ \ +0 \\ .8 \times 2 &= .6\ \ +1 \\ .6 \times 2 &= .2\ \ +1 \\ .2 \times 2 &= .4\ \ +0 \\ .4 \times 2 &= .8\ \ +0 \\ &\cdots\cdots\cdots \end{aligned}$$

Notice that the process repeats after four steps and will repeat infinitely exactly the same way. Therefore,

$$(0.7)_{10} = (.\overline{10110})_2$$

3. Hence,

$$(53.7)_{10} = (110101.1\overline{0110})_2$$

1.1.2 Binary to Decimal

The conversion of binary number to decimal needs to tackle nonterminating binary numbers. Take $x = (.10\overline{101})_2$ as an example.
Multiplying by 2^2 shifts x to

$$y := 2^2 x = (10.\overline{101})_2 = (10)_2 + (.\overline{101})_2 = (2)_{10} + (.\overline{101})_2$$

The fractional part of y, i.e.,

$$z := (.\overline{101})_2$$

is calculated as follows:

$$2^3 z = (101.\overline{101})_2$$

So,

$$(2^3 - 1)z = (101)_2 = (5)_{10}$$

i.e.

$$z = \left(\frac{5}{7}\right)_{10}$$

Thus,

$$y = (2)_{10} + \left(\frac{5}{7}\right)_{10} = \left(\frac{19}{7}\right)_{10}$$

and hence,

$$x = \frac{y}{2^2} = \left(\frac{19}{28}\right)_{10}$$

Theorem 1.1 (Representation of Numbers)**.**
In general, any **natural numbers** $N \geq 2$ can be used as base.
Every positive real number a has a unique representation of the form

$$a = a_m N^m + a_{m-1} N^{m-1} + \cdots + a_1 N^1 + a_0 N^0 + a_{-1} N^{-1} + a_{-2} N^{-2} + \cdots$$

where

$$0 \leq a_i \leq N - 1, \quad a_m \neq 0$$

Equivalently,

$$a = \sum_{i=0}^{\infty} a_{m-i} N^{m-i}$$

where

$$0 < a_m < N; 0 \leq a_{m-i} \leq N - 1 \quad \forall i \geq 1$$

The above non-terminating representation of number a is not feasible in practical computation. We onl deal with, in practice, numbers which have *terminating* representation, i.e. of the following form

$$\alpha = \sum_{i=0}^{n-1} \alpha_{m-i} N^{m-i}$$

where $0 < \alpha_m < N$ and $0 \leq \alpha_{m-i} < N$ for $i = 1, 2, \ldots, n-1$.

Definition 1.2 (Significant decimal digits)**.**
If $N = 10$, then the numbers $\alpha_{m-i}, 0 \leq i \leq n-1$ are called **significant decimal digits** and the number α is said to have n significant decimal digits.

1.2 Chopping and Rounding

If the number a has more significant digits than what we wish to have, then we need to replace a by an approximate number a^* which contains a smaller number of digits, say n. There are two ways of terminating the number a to a given significant number of digits, namely **chopping** or **rounding**.

Definition 1.3 (Chopping)**.**
In **chopping**, we retain only the first n significant digits in the number a.

Definition 1.4 (Rounding)**.**
In **rounding**, the following rules are usually practiced:

- Retain the first n significant digits, and
- if the $(n+1)$th significant digit is less than 5, leave the nth significant digit unchanged;
- otherwise, if the $(n+1)$th significant digit is greater or equal to 5, add unity to the nth significant digit.

1.3 Floating Point Representation

Definition 1.5 (Floating Point Numbers)**.**
Number of the form

$$\pm(0.a_1a_2a_3 \ldots a_m) \times N^e, \text{with } a_1 \neq 0$$

are called **floating point numbers**. The factor $0.a_1a_2a_3 \ldots a_n$ is called the **mantissa**, e is called the **exponent** and N is the **base**.

The folating-point mode is for storing real numbers.

1.4 Basic Concepts in Error Estimation

Definition 1.6 (Sources of Error)**.**
Numerical results are affected by many types of error.

- **Error in Given Input Data**
  The input data can be *result of measurements* which are inexact, or produced by some *arithmetic process using round-off process.*

- **Round-off Errors During Computation**
  This is due to working with *finite machine precision.*

- **Truncation Error**
  Consider the Taylor series xpansion for $\sin(x)$ about $x = 0$

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

  which is an infinite series.
  Truncation error occurs when the *infinite series is broken off after a finite number of terms.*
  In general, truncation errors are error committed when *a limiting process is truncated before one has come to the limiting value.*

- Simplifications in the Mathematical Model

- Human Error

1.5 Absolute and Relative Error

Definition 1.7 (Absolute Error)**.**
Let x^* be an approximation to x, then the **absolute error** in approximating x is given by $|x - x^*|$.

From the above definition, the floating point representation of x, $\mathrm{fl}\,(x)$ has absolute error

$$|x - \mathrm{fl}\,(x)\,|$$

Definition 1.8 (Absolute Error Bound)**.**
Any non-negative number $\delta(x^*)$ satisfying the inequality

$$|x - x^*| < \delta(x^*)$$

is called an **absolute error bound**.

Definition 1.9 (Decimal Places)**.**
The two numbers x is said to agree to k **decimal places** if k is the largest non-negative integer such that

$$|x - x^*| \leq 0.5 \times 10^{-k}$$

Definition 1.10 (Relative Error)**.**
If x^* is an approximation to x, then the **relative error** is defined by

$$\frac{|x - x^*|}{|x|} \quad \text{provided that } x \neq 0$$

For a t-digit machine,

$$\frac{|x - \mathrm{fl}\,(x)\,|}{|x|} \leq \frac{1}{2} \times 10^{1-t}$$

in rounding mode and

$$\frac{|x - \mathrm{fl}\,(x)\,|}{|x|} \leq 10^{1-t}$$

in chopping mode.

1.6 Computation in Floating Point Arithmetic

Take the example of addition/subtraction of two floating point number $0.a_1a_2a_3\ldots a_m \times 10^{N_1} \pm 0.b_1b_2b_3\ldots b_m \times 10^{N_2}$.

$$\begin{aligned} & 0.a_1a_2a_3\ldots a_m \times 10^{N_1} \pm 0.b_1b_2b_3\ldots b_m \times 10^{N_2} \\ = & \begin{cases} (0.a_1a_2a_3\ldots a_m \pm 0.b_1b_2b_3\ldots b_m) \times 10^{N_1} & \text{if } N_1 = N_2 \\ (0.a_1a_2a_3\ldots a_m 0\ldots 0 \pm 0.0\ldots 0b_1b_2b_3\ldots b_m) \times 10^{N_1} & \text{if } N_1 > N_2 \end{cases} \end{aligned}$$

1.7 Propagation of Errors in Function Evaluation

1.7.1 Function of one variable

In finding the value of $f(x)$ by approximating $f(x^*)$, where x^* is the known approximating value of x,
let $\delta(x^*)$ be an absolute error bound for $|x - x^*|$, i.e.,

$$|x - x^*| \leq \delta(x^*)$$

If the function $f(x)$ is differentiable, then by the mean-value theorem, we have

$$f(x) - f(x^*) = f'(p)(x - x^*)$$

where p is some number between x and x^*.
Therefore,

$$\delta(f(x^*)) \leq \max_{t \in \mathbf{I}} |f^*(t)|\delta(x^*)$$

where $\mathbf{I}$ is the interval (x, x^*) if $x < x^*$ or (x^*, x).
Sometimes, upper bound of f' is unavailable. Yet, with the following assumption:

- $\delta(x^*)$ is small
- $f'(x^*) \neq 0$, and
- $f'(t)$ is nearly constant near x^* (i.e., $f'(t)$ does not very greatly for t between x and x^*)

We have $D \approx |f'(x^*)|$, and consequently

$$\delta(f(x^*)) \approx |f'(x^*)|\delta(x^*)$$

1.7.2 Function of Several Variables

Theorem 1.2 (Error bounds of function of several variables)**.**
Generalisation suggests, if x_i^* is an estimate of x_i, $1 \le i \le n$, then with $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$, we have

$$\delta(f(x^* = (x_1^*, x_2^*, \ldots, x_n^*)) \approx \left|\frac{\partial f(x^*)}{\partial x_1}\right| \delta(x_1^*) + \left|\frac{\partial f(x^*)}{\partial x_2}\right| \delta(x_2^*) + \cdots + \left|\frac{\partial f(x^*)}{\partial x_n}\right| \delta(x_n^*)$$

1.8 Catastrophic Cancellation

Calculations involving the **subtraction** of two *nearly equal* numbers can result in considerable loss of accuracy due to cancellation.
Let x_1 and x_2 be two nearly equal numbers and denote the error in x_1 and x_2 by $\delta(x_1)$ and $\delta(x_2)$, respectively, we have

$$\begin{aligned} y &= x_1 - x_2 \\ \Rightarrow |\delta(y)| &\le |\delta(x_1)| + |\delta(x_2)| \\ \Rightarrow \left|\frac{\delta(y)}{y}\right| &\le \frac{|\delta(x_1)| + |\delta(x_2)|}{|x_1 - x_2|} \end{aligned}$$

Specifically, suppose two nearly equal numbers x_1 and x_2, having the same exponent n and being in the k digi representations, are represented in floating point form:

$$\begin{aligned} x_1 &= 0.a_1 a_2 \cdots a_p \alpha_{p+1} \cdots \alpha_k \times 10^n \\ x_2 &= 0.a_1 a_2 \cdots a_p \beta_{p+1} \cdots \beta_k \times 10^n \end{aligned}$$

Then their difference $x_1 - x_2$ in floating point representationwill have some digits in the mantissa to be 0, i.e.,

$$x_1 - x_2 = \text{fl}\left(\text{fl}\left(x_1\right) - \text{fl}\left(x_2\right)\right) = 0.\gamma_{p+1} \cdots \gamma_k \underbrace{0 \cdots 0}_{p\ 0\text{'s}} \times 10^{n-p}$$

So, in floating point calculation, $x_1 - x_2$ has at most $k - p$ significant digits; p significant digits have been lost or canceled due to the subtraction.

1.8.1 Ways to Avoid Subtraction

Theorem 1.3 ($\sqrt{x+\varepsilon} - \sqrt{x}$)**.**
If $|\varepsilon| \ll x$, then

$$\sqrt{x+\varepsilon} - \sqrt{x} = \frac{\varepsilon}{\sqrt{x+\varepsilon} + \sqrt{x}}$$

presents a better way of calculation that reduces loss of accuracy.

Theorem 1.4 (Roots of $ax^2 + bx + c = 0$)**.**
Consider the quadratic equation

$$ax^2 + bx + c = 0 \quad a \neq 0$$

The roots of equations are

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{and} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

To reduce loss of accuracy, some alternative pathway include:

- $x_1 = \frac{-2c}{b+\sqrt{b^2-4ac}}$ and $x_2 = \frac{2c}{-b+\sqrt{b^2-4ac}}$
- $x_1 + x_2 = -b$
- $x_1 x_2 = c$

Theorem 1.5 (Evaluation of polynomial $f(x)$)**.**
First note that $ax^n = \text{fl}\left(\text{fl}\left(a\right) \times \text{fl}\left(x^n\right)\right)$ where $\text{fl}\left(x^n\right) = \text{fl}\left(\text{fl}\left(x^{n-1}\right) \times \text{fl}\left(x\right)\right)$ recursively.
Instead of calculating $f(x) = \text{fl}\left(\sum_{i=0}^{n} a_i x^i\right)$, nesting may provide better result:

$$f(x) = \text{fl}\left((\cdots(a_n x + a_{n-1})x + \cdots)x + a_0\right)$$

1.9 Numerical Instability

Definition 1.11 (Numerical Stability)**.**
An algorithm is said to be **stable** if the effect of its errors on the final result is **negligible**.

2 Numerical Solution of Linear Systems of Equations

Definitions and method of computation of matrix can be found in `MA2101.pdf`, and are therefore omitted here.

2.1 System of Linear Equations

In this chapter, only linear systems with *unique* solution is concerned.
Consider the linear system

$$Ax = b$$

where $A \in \mathbb{M}_n(\mathbb{R})$ and $b \in \mathbb{R}^n_c$. If A is invertible, a *unique* solution x exists and given by

$$x = A^{-1}b$$

There are two classes of method of solving $Ax = b$ numerically:

- **Direct** methods
- **Iterative** methods

In this chapter, only direct methods are studied.
Two basic direct method include

- Cramer's rule
- Gaussian Elimination

2.1.1 Direct Method: Cramer's Rule

Definition 2.1 (Cramer's Rule)**.**
By **Cramer's rule**, the $n \times n$ linear system

$$Ax = b$$

has solution

$$x_i = \frac{d_i}{d} \quad i = 1, \ldots, n$$

where $d := \det(A) \neq 0, d_i = \det(A_i)$, A_i the matrix obtained by replacing the i-th column of A by b.
The determinant d is given by **Laplace Theorem**:

$$d = (-1)^{i+1}a_{i1}D_{i1} + (-1)^{i+2}a_{i2}D_{i2} + \cdots + (-1)^{i+n}a_{in}D_{in}$$

where D_{ij} is the determinant of the submatrix obtained from A by deleting its ith row and jth column.

Theorem 2.1 (Computational Complexity of Cramer's Rule)**.**
Suppose it needs m_n operations of multiplication to compute the determinant of a $n \times n$ matrix. Then, we have

$$m_n = n + nm_{n-1} \quad m_1 = 1$$

Here, the first term n is results from multiplication of a_{ij} and D_{ij}.
Thus,

$$\begin{aligned} m_n &= n + nm_{n-1} = n + n[(n-1) + (n-1)m_{n-2}] \\ &= n + n(n-1) + n(n-1)(n-2) + \cdots + n(n-1)\cdots 3 \cdot 2 \\ &> n! \end{aligned}$$

Hence, in order to solve an $n \times n$ linear system by Cramer's rule and Laplace Theorem, we have do at least

$$(n+1)n! = (n+1)!$$

multiplication.
Clearly, Cramer's rule is too *computationally expensive*.

Also, solving nonlinear linear system by **matrix inversion** is *computationally expensive* and often leads to more *inaccuracies*.

2.2 Gaussian elimination

Definition 2.2 (Elementary Row Operations)**.**
Let E_i be the ith equation in a linear system $Ax = b$. The three **elementary row operations** permitted to solve this linear system are:

- $E_i \leftarrow \lambda E_i$, where λ is a nonzero constant.

$$\begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & \lambda & \\ & & & 1 \end{bmatrix} \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{i1} & \cdots & a_{in} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ \lambda a_{i1} & \cdots & \lambda a_{in} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$$

- $E_i \leftarrow E_i - lE_j$, where l is a nonzero constant.

$$\begin{bmatrix} 1 & & & \\ & \ddots & & \\ & -l \quad 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}^{1} \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{i1} & \cdots & a_{in} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{i1} - la_{j1} & \cdots & a_{in} - la_{jn} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$$

[1] $-l$ is at ith row and jth column of the leftmost matrix.

- $E_i \leftrightarrow E_j$

$$\begin{bmatrix} 1 & & & & \\ & \ddots & & 1 & \\ & & 1 & & \\ & 1 & & \ddots & \\ & & & & 1 \end{bmatrix} \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ a_{i1} & \cdots & a_{in} \\ \vdots & \cdots & \vdots \\ a_{j1} & \cdots & a_{jn} \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ a_{j1} & \cdots & a_{jn} \\ \vdots & \cdots & \vdots \\ a_{i1} & \cdots & a_{in} \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$$

By a sequence of the above operations, a linear system can be transformed to a more easily solved linear system with the **same** set of solutions.
The Gaussian elimination for solving $Ax = b$ involves 2 stages/

- A forward course where a sequence of elementary row operations are applied to $\mathbf{A}$ to reduce it to an upper triangular form $\mathbf{U}$.
- Backward substitution process

Theorem 2.2 (General method of Gaussian Elimination on $\mathbf{Ax} = \mathbf{B}$)**.**
Let the original system be denoted by

$$\mathbf{A}^{(0)}\mathbf{x} = \mathbf{b}^{(0)}$$

where $\mathbf{A}^{(0)} = \mathbf{A} = (a_{ij}^{(0)})$, $\mathbf{b}^{(0)} = \mathbf{b} = \begin{bmatrix} b_1^{(0)} \\ \vdots \\ b_i^{(0)} \\ \vdots \\ b_n^{(0)} \end{bmatrix}$

Step 1: If $a_{11}^{(0)} \neq 0$, let

$$\begin{aligned} l_{i,1} &= \frac{a_{i,1}^{(0)}}{a_{1,1}^{(0)}}{}^{2} && i = 2, 3, \ldots, n \\ a_{i,j}^{(1)} &= a_{i,j}^{(0)} - l_{i,1} \times a_{1,j}^{(0)}{}^{3} && i, j = 2, 3, \ldots, n \\ b_i^{(1)} &= b_i^{(0)} - l_{i,1} \times b_1^{(0)} && i = 2, \ldots, n \end{aligned}$$

Then we obtain the equivalent system

$$\mathbf{A}^{(1)}\mathbf{x} = \mathbf{b}^{(1)}$$

where

$$\mathbf{A}^{(1)} = \begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \cdots & a_{1n}^{(0)} \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ 0 & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} \end{bmatrix}, \quad \mathbf{b}^{(1)} = \begin{bmatrix} b_1^{(0)} \\ b_2^{(1)} \\ \vdots \\ b_n^{(1)} \end{bmatrix}$$

[3]In general, $l_{p,q}$ is used in the operations on qth column, and denotes the ratio between the entry on the pth row$(p > q)$,$a_{p,q}^{(q-1)}$, to the qth diagonal entry, $a_{q,q}^{(q-1)}$.

[3]In general, $a_{i,j}^{(k)}$,$(i < j)$ is obtained by operations on the kth column, and is the result of zeroing the $a_{i,k}^{(k)}$ entry$(i > k)$, so it equals the previous entry $a_{i,j}^{(k-1)}$ minus $l_{i,k}$ times the entry on the kth row $a_{k,j}^{(k-1)}$.

Step 2: If $a_{22}^{(1)} \neq 0$, repeat Step 1 to eliminate $\mathbf{x}_2$ from row 3 to n. After k steps, we obtain the equivalent system

$$\mathbf{A}^{(k)}\mathbf{x} = \mathbf{b}^{(k)}$$

where $\mathbf{A}^{(k)}$ takes the form

$$\mathbf{A}^{(k)} = \begin{bmatrix} a_{11}^{(0)} & \cdots & \cdots & \cdots & \cdots & a_{1n}^{(0)} \\ & a_{22}^{(1)} & \cdots & \cdots & \cdots & a_{2n}^{(1)} \\ & & \ddots & \vdots & \cdots & \cdots \\ & & & a_{k+1,k+1}^{(k)} & \cdots & a_{k+1,n}^{(k)} \\ & & & \vdots & \ddots & \vdots \\ & & & a_{n,k+1}^{(k)} & \cdots & a_{n,n}^{(k)} \end{bmatrix}$$

If $a_{k+1,k+1}^{(k)} \neq 0$, then let

$$\begin{aligned} l_{i,k+1} &= \frac{a_{i,k+1}^{(k)}}{a_{k+1,k+1}^{(k)}} \\ a_{i,j}^{(k+1)} &= a_{i,j}^{(k)} - l_{i,k+1} \times a_{k+1,j}^{(k)} \\ b_i^{(k+1)} &= b_i^{(k)} - l_{i,k+1} b_{k+1}^{(k)} \end{aligned}$$

Finally, after at most $n-1$ steps, the system becomes

$$\mathbf{A}^{(n-1)}\mathbf{x} = \mathbf{b}^{(n-1)}$$

which is an upper triangular system, where

$$\mathbf{U} = \mathbf{A}^{(n-1)} = \begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \cdots & a_{1n}^{(0)} \\ & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ & & \ddots & \vdots \\ & & & a_{nn}^{(n-1)} \end{bmatrix}$$

and

$$\mathbf{b}^{(n-1)} = \begin{bmatrix} b_1^{(0)} \\ b_2^{(1)} \\ \vdots \\ b_n^{(n)} \end{bmatrix}$$

Step 3: The solution vector $\mathbf{x}$ is solved by back substitution:

$$x_n = \frac{b_n^{(n-1)}}{a_{nn}^{(n-1)}}$$

and for $k = n-1, n-2, \ldots, 1$,

$$x_k = \frac{1}{a_{kk}^{(k-1)}} \left(b_k^{(k-1)} - \sum_{j=k+1}^{n} a_{kj}^{(k-1)} x_j \right)$$

The above algorithm is known as **Gaussian elimination**.

2.3 Triangular Factorisation: $\mathbf{A} = \mathbf{LU}$

We note, in the $(k+1)$th step of above Gaussian elimination, all rows below the $(k+1)$th row are minused by a multiple of the $(k+1)$th row. This operation can be viewed by a collection of elementary row operation 2 and represented by

$$\mathbf{A}^{(k+1)} = \mathbf{L}^{(k)}\mathbf{A}^{(k)}$$

where

$$\mathbf{L}^{(k)} = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -l_{k+2,k+1} & \ddots & & \\ & & \vdots & & \ddots & \\ & & -l_{n,k+1} & & & 1 \end{bmatrix}$$

This relation gives

$$\mathbf{U} = \mathbf{A}^{(n-1)} = \mathbf{L}^{(n-2)}\mathbf{L}^{(n-3)}\cdots\mathbf{L}^{(0)}\mathbf{A}^{(0)}$$

Also, note the inverse of $\mathbf{L}^{(k)}$ is

$$(\mathbf{L}^{(k)})^{-1} = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & l_{k+2,k+1} & \ddots & & \\ & & \vdots & & \ddots & \\ & & l_{n,k+1} & & & 1 \end{bmatrix}$$

a flip of signs of all $l_{i,k+1}$ from $\mathbf{L}^{(k)}$.
Therefore,

$$(\mathbf{L}^{(0)})^{-1}(\mathbf{L}^{(1)})^{-1}\cdots(\mathbf{L}^{(n-2)})^{-1}\mathbf{U} = \mathbf{A}$$

Note that $(\mathbf{L}^{(0)})^{-1}(\mathbf{L}^{(1)})^{-1}\cdots(\mathbf{L}^{(n-2)})^{-1}$ is upper triangular and is defined as $\mathbf{L}$:

$$(\mathbf{L}^{(0)})^{-1}(\mathbf{L}^{(1)})^{-1}\cdots(\mathbf{L}^{(n-2)})^{-1} = \begin{bmatrix} 1 & & & & & \\ l_{21} & \ddots & & & & \\ \vdots & & 1 & & & \\ l_{k+2,1} & \cdots & l_{k+2,k+1} & \ddots & & \\ \vdots & & \vdots & & \ddots & \\ l_{n,1} & \cdots & l_{n,k+1} & \cdots & l_{n,n-1} & 1 \end{bmatrix} := \mathbf{L}$$

Hence, $\mathbf{A} = \mathbf{LU}$, where $\mathbf{U} = \mathbf{A}^{(n-1)}$ and $\mathbf{L}$ is a lower triangular matrix.

Theorem 2.3. If $\mathbf{U}$ and $\mathbf{L}$ are the upper and lower triangular matrices defined above, then $\mathbf{A} = \mathbf{LU}$.

2.4 Compact Forms of Gaussian Elimination

If $\mathbf{A}$ admits an $\mathbf{LU}$ decomposition, we can solve the system of equations $\mathbf{LUx} = \mathbf{b}$ in two stages.

- Set $\mathbf{z} := \mathbf{Ux}$. Solve $\mathbf{Lz} = \mathbf{b}$ for $\mathbf{z}$.
- Solve $\mathbf{Ux} = \mathbf{z}$ for $\mathbf{x}$

Theorem 2.4 (Existence and Uniqueness of $\mathbf{LU}$ decomposition)**.**
Let $\mathbf{A}$ be an $n \times n$ matrix and $\mathbf{A}^{(k)}$ be the $k \times k$ matrix formed from the first k rows and columns of $\mathbf{A}$. If $\det(\mathbf{A}^{(k)}) \neq 0 \forall k = 1, 2, \ldots, n-1$, then there exists a unique lower triangular matrix $\mathbf{L} = (l_{ij})$ with $l_{ii} = 1 \forall i = 1, 2 \ldots, n$ and a unique upper triangular matrix $\mathbf{U} = (u_{ij})$ such that $\mathbf{A} = \mathbf{LU}$.

2.5 Compact Forms of Gaussian Elimination

The $\mathbf{LU}$ decomposition can be calculated directly by Gaussian Elimination.
Given that $mathbf A$ admits a $\mathbf{LU}$ decomposition $\mathbf{A} = \mathbf{LU}$, we have, entry-wise

$$a_{ij} = \sum_{k=1}^{n} l_{ik} u_{kj} \quad i, j = 1, 2, \ldots, n$$

Note that the above system has n^2 equations and $n^2 + n$ unknowns $l_{ik}, i \geq k$ and $u_{kj}, k \leq j$. Thus n unknowsn may be set arbitrarily.

Theorem 2.5 (Doolittle Method)**.**
For Doolittle Method, set diagonal entries of $\mathbf{L}$, $l_{kk} := 1, k = 1, 2, \ldots n$ and assume that $u_{kk} \neq 0, \forall k \in [1, n] \cap \mathbb{Z}$.
The sequence of calculation is as follows:

1. First row of $\mathbf{U}$: $u_{1k}, 1 \leq k \leq n$
2. First column of $\mathbf{L}$: $l_{k1}, 2 \leq k \leq n$[4]
3. Second row of $\mathbf{U}$: $u_{2k}, 2 \leq k \leq n$.

   $\vdots$

Obviously, when calculating kth row of $\mathbf{U}$, row 1 to $k-1$ of $\mathbf{U}$ and column 1 to $k-1$ of $\mathbf{L}$ will be known.
Also, when calculating kth row of $\mathbf{L}$, row 1 to k of $\mathbf{U}$ and column 1 to $k-1$ of $\mathbf{L}$ will be known.
Suppose $u_{kj}, (j \geq k)$ is of concern, we have

$$a_{kj} = \sum_{r=1}^{k} l_{kr} u_{rj}$$

[4] if $k = 1, l_{k1} = l_{11} = 1$ by definition

Substitute in $l_{kk} = 1$ and rearranging,

$$u_{kj} = a_{kj} - \sum_{r=1}^{k-1} l_{kr} u_{rj}$$

Afterwards, suppose $l_{ik}, (i > k)$ is of concern, we have

$$a_{ik} = \sum_{r=1}^{k} l_{ir} u_{rk}$$

Rearranging,

$$a_{ik} = \sum_{r=1}^{k-1} l_{ir} u_{rk} + l_{ik} u_{kk}$$

Therefore,

$$l_{ik} = \frac{a_{ik} - \sum_{r=1}^{k-1} l_{ir} u_{rk}}{u_{kk}}$$

The above two equation can be interleaved to obtain $\mathbf{L}$ and $\mathbf{U}$.

Theorem 2.6 (Crout Method)**.**
For Crout Method, set diagonal entries of $\mathbf{U}$, $u_{ii} = 1, k = 1, 2, \ldots, n$ and assume that $l_{kk} \neq 0 \forall k \in [1, n] \cap \mathbb{Z}$.
The sequence of calculation is as follows:

1. First column of $\mathbf{L}$: $l_{k1}, 1 \leq k \leq n$
2. First row of $\mathbf{U}$: $u_{1k}, 2 \leq k \leq n$
3. Second column of $\mathbf{L}$: $l_{k2}, 2 \leq k \leq n$

   $\vdots$

Obviously, when calculating kth column of L, column 1 to $k-1$ of $\mathbf{L}$ and row 1 to $k-1$ of $\mathbf{U}$ will be known.
Also, when calculating kth row of $\mathbf{U}$, column 1 to k of $\mathbf{L}$ and row 1 to $k-1$ of $\mathbf{U}$ will be known.
Suppose $l_{ik}, (i \geq k)$ is of concern, we have

$$a_{ik} = \sum_{r=1}^{k} l_{ir} u_{rk}$$

Substitute in $u_{kk} = 1$ and rearranging,

$$l_{ik} = a_{ik} - \sum_{r=1}^{k-1} l_{ir} u_{rk}$$

Afterwards, suppose $u_{kj}, (j \geq k+1)$ is of concern, we have

$$a_{kj} = \sum_{r=1}^{k} l_{kr} u_{rj}$$

Rearranging,

$$u_{kj} = \frac{a_{kj} - \sum_{r=1}^{k-1} l_{kr} u_{rj}}{l_{kk}}$$

The above two equation can be interleaved to obtain **L** and **U**.

2.6 LU Decomposition for Tridiagonal Matrices

Theorem 2.7.
Let **A** be a **tridiagonal matrices**

$$\mathbf{A} = \begin{bmatrix} a_1 & c_1 & & & & \\ b_2 & a_2 & c_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & b_{n-1} & a_{n-1} & c_{n-1} \\ & & & & b_n & a_n \end{bmatrix}$$

If an **LU** decomposition exists for **A**, then

$$\mathbf{A} = \mathbf{LU}$$

where

$$\mathbf{L} = \begin{bmatrix} 1 & & & & & \\ \beta_2 & 1 & & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \beta_{n-1} & 1 & \\ & & & & \beta_n & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{U} = \begin{bmatrix} \alpha_1 & c_1 & & & & \\ & \alpha_2 & c_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & & \alpha_{n-1} & c_{n-1} \\ & & & & & \alpha_n \end{bmatrix}$$

By applying Doolittle's method, it can be shown that

$$\alpha_1 = a_1$$

and for $k = 2, 3, \ldots, n$,

$$\beta_k = \frac{b_k}{\alpha_{k-1}}$$
$$\alpha_k = a_k - \beta_k c_{k-1}$$

Theorem 2.8 (Thomas's Algorithm)**.**
If $\mathbf{Ax} = \mathbf{g}$ and define $\mathbf{Ux} = h$ where $\mathbf{x} := (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$, $\mathbf{g} = (g_1, g_2, \ldots, g_n)^{\mathrm{T}}$ and $\mathbf{h} = (h_1, h_2, \ldots, h_n)^{\mathrm{T}}$.

$$\begin{aligned} h_1 &= g_1 \\ h_i &= g_i - \beta_i h_{i-1}, \quad i = 2, 3, \ldots, n \end{aligned}$$

and

$$\begin{aligned} x_n &= \frac{h_n}{\alpha_n} \\ x_i &= \frac{h_i - c_i x_{x+1}}{\alpha_i} \end{aligned}$$

This method is known as **Thomas algorithm**.

2.7 Pivoting Strategies

2.7.1 Partial Pivoting

Definition 2.3 (Partial Pivoting)**.**
At the $(k+1)$th step of the Gaussian Elimination process, $k = 0, 1, \ldots, n-2$, choose the element having maximum absolute value in the $(k+1)$th column of $\mathbf{A}^{(k)}$ that lies on or below the diagonal so that

$$|a_{s,k+1}^{(k)}| = \max_i |a_{i,k+1}^{(k)}| \quad k+1 \leq i \leq n$$

and interchange row $k+1$ with row s.

2.7.2 Scaled Partial Pivoting

Definition 2.4 (Scaled Partial Pivoting)**.**
In the beginning, calculate s_i for all row i, where

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

At the $(k+1)$th step of the Gaussian Elimination process, $k = 0, 1, \ldots, n-2$, choose rth row of $\mathbf{A}^{(k)}$ that lies on or below the diagonal where r is determined by

$$\frac{|a_{r,k+1}^{(k)}|}{s_r} = \max_i \frac{|a_{i,k+1}^{(k)}|}{s_i} \quad k+1 \leq i \leq n$$

and exchange row $k+1$ with row r and s_{k+1} with s_r.

3 Error of Approximation

Definition 3.1 (Error)**.**
Given $n+1$ data values $(x_0, f_0), (x_1, f_1), \ldots, (x_n, f_n)$, we may define the error of approximation, E, by

$$E = \sum_{i=0}^{n} |p(x_i) - f_i|$$

If we take $p(x)$ to be a polynomial of degree n then by choosing

$$p(x_i) = f_i, i = 0, 1, \ldots, n$$

we can make $E = 0$. The function $p(x)$ is the **Lagrange interpolating polynomial**.
Alternatively, when n is large, we may fit a polynomial of degree k where $k \ll n$. We use

$$E = \sum_{i=0}^{n} [p(x_i) - f_i]^2$$

By minimising E we achieve a **least square fit**.
If the function $f(x)$ is continuous specified on the interval $[a, b]$, we may define the error E in approximating $f(x)$ by $p(x)$ as

$$E = \max_{a \le x \le b} |p(x) - f(x)|$$

It is common in practice to take $p(x)$ as a polynomial and minimise E with respect to variations in the coefficients. This generate the **minimax** polynomial approximation.

The following theorem justifies the choice of polynomial for approximation

Theorem 3.1 (Weierstrass Approximation Theorem)**.**
Suppose f is defined and continuous on $[a, b]$. For each $\varepsilon > 0$ there exists a polynomial $P(x)$, defined on $[a, b]$, with property that

$$|f(x) - P(x)| < \varepsilon \quad \forall x \in [a, b]$$

3.1 Least Square Approximation

3.1.1 Discrete Data

Given a set of $m+1$ discrete data points $(x_0, f(x_0)), (x_1, f(x_0)), \ldots (x_m, f(x_m))$, each carrying weight $w_0, w_1, \ldots, w_m$ respectively.
The task is to find an approximating polynomial function[5]

$$p_n(x; a_0, a_1, \ldots, a_n) := a_0 + a_1 x + \cdots + a_n x^n$$

[5]The subscript n denotes the power of the polynomial; thus the polynomial p_n has $n+1$ unknwons

$(m > n)$ such that

$$E(a_0, a_1, \ldots, a_n) = \sum_{i=0}^{m} w_i [f(x_i) - p_n(x_i; a_0, a_1, \ldots, a_n)]^2$$

is minimised with respect to the parameters $a_0, a_1, \ldots, a_n$.[6]
We require

$$\frac{\partial E}{\partial a_j} = 9 \forall j = 0, 1, \ldots, n$$

Therefore, by differentiating with a_j,

$$\sum_{i=0}^{m} w_i [f(x_i) - p_n(x_i; a_0, a_1, \ldots, a_n)] \frac{\partial}{\partial a_j} \left(-p_n(x_i; a_0, a_1, \ldots, a_n)\right) \quad \forall j = 0, 1, \ldots, n$$

Rearranging, we have

$$\sum_{i=0}^{m} w_i \underbrace{\left(a_0 + a_1 x_i + \cdots + a_n x_i^n\right)}_{p_n(x_i; a_0, a_1, \ldots, a_n)} x_i^j = \sum_{i=0}^{m} w_i f(x_i) x_i^j \quad \forall j = 0, 1, \ldots, n$$

There is $n + 1$ unknowns, and $n + 1$ equations, so the linear system is[7]

$$\begin{bmatrix} \sum_{i=0}^m w_i x_i^0 & \sum_{i=0}^m w_i x_i^1 & \cdots \sum_{i=0}^m w_i x_i^n \\ \sum_{i=0}^m w_i x_i^1 & \sum_{i=0}^m w_i x_i^2 & \cdots \sum_{i=0}^m w_i x_i^{n+1} \\ \vdots & \vdots & \ddots \quad \vdots \\ \sum_{i=0}^m w_i x_i^n & \sum_{i=0}^m w_i x_i^{n+1} & \cdots \sum_{i=0}^m w_i x_i^{2n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^m w_i f(x_i) x_i^0 \\ \sum_{i=0}^m w_i f(x_i) x_i^1 \\ \vdots \\ \sum_{i=0}^m w_i f(x_i) x_i^n \end{bmatrix}$$

3.1.2 Continuous Function

Given a continuous function $f(x)$ defined on an interval $[a, b]$ with each value $x \in [a, b]$ associated with a weight $w(x)$.
The task is to find an approximating polynomial function $p_n(x; a_0, a_1, \ldots, a_n)$ such that

$$E(a_0, a_1, \ldots, a_n) = \int_a^b w(x) [f(x) - p_n(x; a_0, a_1, \ldots, a_n)]^2 \, \mathrm{d}\, x$$

is minimised with respect to the parameters $a_0, a_1, \ldots, a_n$.
The necessary condition sfor $E(a_0, a_1, \ldots, a_n)$ to be minimum are

$$\frac{\partial E}{\partial a_j} = 0 \forall j = 0, 1, \ldots, n$$

Differentiating with respect to a_j,

$$\frac{\mathrm{d}\, E}{\mathrm{d}\, a_j} = -2 \int_a^b w(x) [f(x) - p(x; a_0, a_1, \ldots, a_n)] \frac{\partial}{\partial a_j} (p_n(x; a_0, a_1, \ldots, a_n)) \, \mathrm{d}\, x = 0$$

[6] E can be understood as a weighted sum of squares of residue for each data.
[7] the jth column corresponding to the equation of specific j

Rearranging,[8]

$$\int_a^b w(x)(a_0 + a_1x + \cdots + a_nx^n)x^j \,\mathrm{d}\,x = \int_a^b w(x)f(x)x^j \,\mathrm{d}\,x \quad \forall j = 0, 1, \ldots, n$$

$$\sum_{k=0}^{n} \int_a^b w(x)x^{k+j} \,\mathrm{d}\,x = \int_a^b w(x)f(x)x^j \,\mathrm{d}\,x \quad \forall j = 0, 1, \ldots, n$$

Similarly, a matrix equation can be formed from this linear system of $n+1$ variables and $n+1$ equations. However, this matrix is **ill-conditioned**.

3.2 Lagrange Interpolation

Definition 3.2.
Given a set of $n+1$ data points $(x_0, f_0), (x_1, f_1), \ldots, (x_n, f_n)$, we seek to find a polynomial $p_n(x)$ of degree n which passes through each of the given points, i.e.

$$p_n(x_i) = f_i \quad i = 0, 1, 2, \ldots, n$$

If we write $p_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$, we have

$$\begin{bmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & x_1 & \cdots & x_1^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix}$$

The determinant of the $(n+1) \times (n+1)$ Vandermonde matrix is $\prod_{0 \le i < j \le n} (x_j - x_i)$, so the system has a unique solution as long as $x_i, i = 1, 2, \ldots, n$ are all distinct.
The solved polynomial is the unique Lagrange interpolating polynomial.

Theorem 3.2 (Computation of Lagrange Interpolating Polynomial)**.**
Write

$$\begin{aligned} p_n(x) = {} & \alpha_0(x - x_1)(x - x_2) \cdots (x - x_n) \\ & + \alpha_1(x - x_0)(x - x_2) \cdots (x - x_n) \\ & + \cdots \\ & + \alpha_n(x - x_0)(x - x_1) \cdots (x - x_{n-1}) \end{aligned}$$

Then, we have

$$\alpha_i = \frac{f_i}{\prod_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)}$$

[8] $\frac{\partial}{\partial a_j}(p_n(x; a_0, a_1, \ldots, a_n)) = x^j$ is obvious.

So,

$$p_n(x) = \sum_{i=0}^{n} l_i(x) f_i$$

where

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

Note that $l_i(x)$ admits the property that

$$l_i(x_i) = 1$$
$$l_i(x_j) \neq 1 \text{ for } j \neq i$$

Theorem 3.3 ($\psi(x)$ and alternative expression of Lagrange Interpolating Polynomial)**.** Let $\psi(x)$ denote the polynomial

$$\psi(x) = (x - x_0)(x - x_1) \cdots (x - x_n) = \prod_{i=0}^{n} (x - x_i)$$

then, by differentiating on each product term once at a time, we have

$$\psi'(x) = \sum_{i=0}^{n} \prod_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)$$

and hence

$$\psi'(x_i) = (x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)$$

as all other terms containing $(x - x_i)$ will vanish.
Thus,[9]

$$l_i(x) = \frac{\psi(x)}{(x - x_i)\psi'(x_i)}$$

So $p_n(x)$ may be written as

$$p_n(x) = \sum_{i=0}^{n} \frac{\psi(x)}{(x - x_i)\psi'(x_i)} f_i$$

3.3 Truncation Error of Interpolating Polynomial

Theorem 3.4 (Extended Mean Value Theorem)**.**
Suppose that

$$a \leq x_0 < x_1 < \cdots < x_k \leq b$$
$$f(x_0) = f(x_1) = \cdots = f(x_k) = 0$$

and $f(x), f'(x), \ldots, f^{(k)}(x)$ are all continuous on $[a, b]$. There is a $\xi \in (x_0, x_n) \subset (a, b)$ such that

$$f^k(\xi) = 0$$

[9]Do NOT forget multiply $(x - x_i)$ while using this formula!

Theorem 3.5 (Lagrange Interpolating Polynomial Error Formula)**.**
Let $f(x) \in C^{(n+1)}[a,b]$. Further, let $p_n(x)$ interpolates $f(x)$ at $(n+1)$ distinct points $x + 0, x_1, \ldots, x_n \in [a,b]$. Then, for $x \in [a,b], x \neq x_i, i = 0,1,\ldots,n$,

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

for some $xi \in \text{Spr}\{x, x_0, x_1, \ldots, x_n\}$.
Here, $\text{Spr}\{x, x_0, x_1, \ldots, x_n\}$ denotes the smallest interval containing $x, x_0, x_1, \ldots, x_n$.

It follows that an upper bound for the error is

$$\delta(f(x)) := |f(x) - p_n(x)| \leq \frac{M}{(n+1)!}|\psi(x)|$$

where

$$M = \max_{a \leq \xi \leq b} |f^{(n+1)}(\xi)|$$

3.4 Divided Differences

Definition 3.3 (Divided Differences)**.**
Let $(x_0, f(x_0)), \ldots, (x_n, f(x_n))$ be $n+1$ given data points. The **first divided difference** of $f(x)$ between x_0 and x_1, denoted by $f[x_0, x_1]$ is defined by

$$f[x_0, x_1] := \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

Analogously, the **second divided difference** for the triplet (x_0, x_1, x_2) is defined by

$$f[x_0, x_1, x_2] := \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}$$

And, the nth divided difference of $f(x)$ at $(n+1)$-tuple $(x_0, \ldots, x_n)$ is defined recursively in terms of the $(n-1)$th divided difference as follows:

$$f[x_0, x_1, \ldots, x_n] := \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0}$$

Theorem 3.6 (Divided difference of self)**.**
We define $f[x_i, x_i]$ by

$$f[x_i, x_i] = \lim_{x_{i+1} \to x_i} f[x_i, x_{i+1}] = f'(x_i)$$

provided $f'(x_i)$ exists.
Similarly, we define rth divided difference of self is

$$f[\underbrace{x_i, \ldots, x_i}_{rx_i's}] = \frac{1}{r!}f^{(r)}(x_i)$$

3.5 Newton Interpolation Formula

It is easy to see, by definition of first order divided difference, that

$$f(x) = f(x_0) + (x - x_0)f[x, x_0]$$

Therefore, we can obtain

$$\begin{aligned} f(x) &= f(x_0) + (x - x_0)f[x_0, x_1] + \cdots + (x - x_0)\cdots(x - x_{n-1})f[x_0, \ldots, x_n] + R_n(x) \\ &= f(x_0) + \sum_{i=0}^{n-1} f[x_0, \ldots, x_{i+1}] \prod_{j=0}^{i} (x - x_j) + R_n(x) \end{aligned}$$

where

$$R_n(x) = \psi(x)f[x, x_0, x_1, \ldots, x_n]$$

The Newton interpolating formula of these $n + 1$ data points is

$$p_n(x) = f(x_0) + (x - x_0)f[x_0, x_1] + \cdots + (x - x_0)\cdots(x - x_{n-1})f[x_0, \ldots, x_n]$$

To see this, we note that at $x_0, \ldots, x_n$, $R_n = 0$, so $f(x) = p_n(x)$; also $p_n(x)$ is at most of degree n.
When expanding the formula p_n to p_{n+1}, we note that $f = p_{n+1} = p_n + (x - x_0)\cdots(x - x_n)f[x_0, \cdots, x_{n+1}]$, so the divided difference can be easily acquired.

Remark: $f[x_0, \ldots, x_n] = f[x_{i_0}, \ldots, x_{i_n}]$, where $\{i_k\}_{k=1}^n$ is a permutation of $\{k\}_{k=1}^n$.

3.6 Cubic Spline Interpolation

Cubic Spline interpolation is a piecewise approximation used to replace high order polynomial interpolation, latter of which has the problem of wild oscillation.

Definition 3.4 (Cubic Spline Approximation)**.**
Given a function f defined on $[a, b]$ and a set of data points x_i with $a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$, a **cubic spline interpolant** to f is a *piecewise* function S satisfying the following conditions:

1. S is a cubic polynomial, denoted S_j on the subinterval $[x_j, x_{j+1}]$ for $j = 0, 1, \ldots, n-1$.
2. Interpolation condition: $S(x_j) = f(x_j)$.
3. $S \in C^1(a, b)$:$S'_j(x_{j+1}) = S'_{j+1}(x_{j+1}) \quad j = 0, 1, \ldots, n-2$.
4. $S \in C^2(a, b)$:$S''_j(x_{j+1}) = S''_{j+1}(x_{j+1}) \quad j = 0, 1, \ldots, n-2$.
5. One of the following sets of boundary condition is satisfied
   - $S''_0(x_0) = S''_{n-1}(x_n) = 0$ (Natural boundary condition)
   - Or, $S'_0(x_0) = f'(x_0)$ and $S'_{n-1}(x_n) = f'(x_n)$ (Clamped boundary condition)

Theorem 3.7 (Solving cubic spline interpolation efficiently)**.**
The following method will produce the cubic spline interpolation of $n+1$ data points by solving a linear system of maximum size $(n+1)\times(n+1)$.
Let $s_i(x)$ be a cubic polynomial on $[x_i, x_{i+1}]$. Then $s_i''(x)$ is the linear polynomial on $[x_i, x_{i+1}]$.
Let $M_i = s_i''(x_i)$. M_i will be calculated later using (3) and (5).
Using lagrange interpolation polynomial, we have

$$s_i''(x) = \frac{x_{i+1}-x}{h_i}M_i + \frac{x-x_i}{h_i}M_{i+1}$$

where $h_i = x_{i+1} - x_i$ the step size.
Integrating twice, we have

$$s_i(x) = \frac{(x_{i+1}-x)^3}{6h_i}M_i + \frac{(x-x_i)^3}{6h_i}M_{i+1} + Ax + B$$

for some constant A and B.
Using (2), i.e.,

$$s_i(x_i) = f_i \qquad s_i(x_{i+1}) = f_{i+1}$$

we have, nicely[10]

$$\frac{h_i^2}{6}M_i + Ax_i + B = f_i$$
$$\frac{h_i^2}{6}M_{i+1} + Ax_{i+1} + B = f_{i+1}$$

Solving which we have

$$\begin{cases} A &= \frac{f_{i+1}-f_i}{h_i} + \frac{h_i}{6}(M_i - M_{i+1}) \\ B &= \frac{x_{i+1}f_i - x_i f_{i+1}}{h_i} + \frac{h_i}{6}(x_i M_{i+1} - x_{i+1}M_i) \end{cases}$$

Therefore,

$$\begin{aligned} s_i(x) = &\frac{6}{h_i}[(x_{i+1}-x)^3 M_i + (x-x_i)^3 M_{i+1}] \\ &- \frac{h_i}{6}[(x_{i+1}-x)M + (x-x_i)M_{i+1}] \\ &+ \frac{1}{h_i}[(x_{i+1}-x)f_i + (x-x_i)f_{i+1}] \end{aligned}$$

From here, differentiate $s_i(x)$, we have expression for $s_i'(x)$:

$$s_i'(x) = \frac{1}{2h_i}[-(x_{i+1}-x)^2 M_i + (x-x_i)^2 M_{i+1}] - \frac{h_i}{6}(-M_i + M_{i+1}) + \frac{1}{h_i}(-f_i + f_{i+1})$$

[10] At each substitution, one cubic power vanishes and the other degenerates to h_i, which simplifies the calculation.

Replacing the subscript i by $i-1$, we have

$$s'_{i-1}(x) = \frac{1}{2h_{i-1}}[-(x_i - x)^2 M_{i-1} + (x - x_{i-1})^2 M_i] - \frac{h_{i-1}}{6}(-M_{i-1} + M_i) + \frac{1}{h_{i-1}}(-f_{i-1} + f_i)$$

Imposing (3), i.e. $s'_{i-1}(x_i) = s'_i(x_i)$, we will have

$$h_{i-1}M_{i-1} + 2(h_{i-1} + h_i)M_i + h_i M_{i+1} = 6\frac{f_{i+1} - f_i}{h_i} - 6\frac{f_i - f_{i-1}}{h_{i-1}}$$

for $i = 1, \ldots n-1$. This equation represents a system of $n-1$ linear equations with $n+1$ unkwowns; the other 2 equation is obtained from the boundary condition.

Natural boundary condition:

Under natural boundary condition, $M_0 = M_n = 0$ by definition. Then we have

$$\mathbf{Am} = \mathbf{b}$$

where

$$\mathbf{A} = \begin{pmatrix} 2(h_0 + h_1) & h_1 & & & \\ h_1 & 2(h_1 + h_2) & h_2 & & \\ & \ddots & \ddots & \ddots & \\ & & h_{n-3} & 2(h_{n-3} + h_{n-2}) & h_{n-2} \\ & & & h_{n-2} & 2(h_{n-2} + h_{n-1}) \end{pmatrix}$$

$$\mathbf{m} = \begin{pmatrix} M_1 & M_2 & \cdots & M_{n-1} \end{pmatrix}^t$$

and

$$\mathbf{b} = 6 \times \begin{pmatrix} \frac{f_0}{h_0} - \frac{f_1}{h_0} - \frac{f_1}{h_1} + \frac{f_2}{h_1} \\ \vdots \\ \frac{f_{i-1}}{h_{i-1}} - \frac{f_i}{h_{i-1}} - \frac{f_i}{h_i} + \frac{f_{i+1}}{h_i} \\ \vdots \\ \frac{f_{n-2}}{h_{n-2}} - \frac{f_{n-1}}{h_{n-2}} - \frac{f_{n-1}}{h_{n-1}} + \frac{f_n}{h_{n-1}} \end{pmatrix}$$

Clamped boundary condition:

Under clamped boundary condition, $s'_0(x_0) = f'(x_0)$ and $s'_{n-1}(x_n) = f'(x_n)$, which we will have, from $s'_i(x)$ equation,

$$2h_0 M_0 + h_0 M_1 = 6(-\frac{f_0}{h_0} + \frac{f_1}{h_0} - f'_0)$$

$$h_{n-1}M_{n-1} + 2h_{n-1}M_n = 6(\frac{f_{n-1}}{h_{n-1}} - \frac{f_n}{h_{n-1}} + f'_n)$$

Therefore, we can form $\mathbf{Am} = \mathbf{b}$ accordingly, where

$$\mathbf{A} = \begin{pmatrix} 2h_0 & h_0 & & & \\ h_0 & 2(h_0 + h_1) & h_1 & & \\ & \ddots & \ddots & \ddots & \\ & & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \\ & & & h_{n-1} & 2h_{n-1} \end{pmatrix}$$

$$\mathbf{m} = \begin{pmatrix} M_0 & M_1 & \cdots & M_n \end{pmatrix}^t$$

and

$$\mathbf{b} = 6 \times \begin{pmatrix} -\frac{f_0}{h_0} + \frac{f_1}{h_0} - f'_0 \\ \frac{f_0}{h_0} - \frac{f_1}{h_0} - \frac{f_1}{h_1} + \frac{f_2}{h_1} \\ \vdots \\ \frac{f_{i-1}}{h_{i-1}} - \frac{f_i}{h_{i-1}} - \frac{f_i}{h_i} + \frac{f_{i+1}}{h_i} \\ \vdots \\ \frac{f_{n-2}}{h_{n-2}} - \frac{f_{n-1}}{h_{n-2}} - \frac{f_{n-1}}{h_{n-1}} + \frac{f_n}{h_{n-1}} \\ \frac{f_{n-1}}{h_{n-1}} - \frac{f_n}{h_{n-1}} + f'_n \end{pmatrix}$$

4 Numerical Integration

4.0.1 Prerequisite – Useful Theorems

Theorem 4.1 (Intermediate Value Theorem)**.**
If $f(x)$ is continuous on $[a, b]$, let $m = \min_{a \le x \le b} f(x)$ and $M = \max_{a \le x \le b} f(x)$, then for any K satisfying $m \le K \le M$, there exists $xi \in (a, b)$ such that $f(\xi) = K$.

From this theorem we have the following theorem.

Theorem 4.2 (Mean Value Theorem for Integrals)**.**
Let $g(x)$ be a **non-negative** or **non-positive** integrable function on $[a, b]$. If $f(x)$ is continuous on $[a, b]$, then there is at least one point ξ in $[a, b]$ such that

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

4.1 Quadrature based on Interpolation Polynomials

We first select a set of $(n + 1)$ distinct nodes $\{x_0, \ldots, x_n\}$ from the interval $[a, b]$. The Lagrange interpolating polynomial is

$$p_n(x) = \sum_{i=0}^{n} l_i(x)f(x_i)$$

with truncation error term

$$E_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\psi(x)$$

which admits $f(x) = p_n(x) + E_n(x)$.
We approximate $\int_a^b f(x)$ by considering the integral of $p_n(x)$, denoted by $I_n(f)$

$$I_n(f) := \int_a^b p_n(x)\,\mathrm{d}\,x = \int_a^b \sum_{i=0}^{n} l_i(x)f(x_i)\,\mathrm{d}\,x = \sum_{i=0}^{n} a_i f(x_i)$$

where $a_i = \int_a^b l_i(x)$. Therefore, the truncation error will generate the error term of $I_n(f)$, denoted by $E_n(f)$

$$E_n(f) := \int_a^b \frac{f^{(n+1)}(\xi(x))\psi(x)}{(n+1)!}\,\mathrm{d}\,x = \frac{1}{(n+1)!}\int_a^b f^{(n+1)}(\xi(x))\psi(x)\,\mathrm{d}\,x$$

where $\xi(x)$ is in $[a, b]$ for each x.
Therefore we can write the integral of $f(x)$ as

$$I(f) := \int_a^b f(x)\,\mathrm{d}\,x = I_n(f) + E_n(f)$$

where $I(f) \approx I_n(f)$.

4.2 Trapezoidal Rule

Trapezoidal rule is produced by let $n = 1$, $x_0 = a$ and $x_1 = b$.
Under these condition, interpolating polynomial is

$$p_1(x) = \frac{x-b}{a-b}f(a) + \frac{x-a}{b-a}f(b)$$

a line passes through $(a, f(a))$ and $(b, f(b))$.
Then,

$$\begin{aligned} I(f) = \int_a^b f(x)\,\mathrm{d}\,x &= \int_a^b p_1(x)\,\mathrm{d}\,x + E_1(f) \\ &\approx \frac{b-a}{2}[f(a)+f(b)] \quad \text{(Trapezoidal Rule)} \end{aligned}$$

by ignoring error term $E_1(f)$.
The error term is

$$\begin{aligned} E_1(f) &= \int_a^b \frac{1}{2}f''(\xi(x))(x-a)(x-b)\,\mathrm{d}\,x \\ &= \frac{1}{2}f''(\xi)\int_a^b (x-a)(x-b)\,\mathrm{d}\,x \quad \text{(by MVT for integrals)} \\ &= \frac{1}{2}f''(\xi)\left(-\frac{(b-a)^3}{6}\right) \\ &= -\frac{1}{12}f''(\xi) \end{aligned}$$

Definition 4.1 (Degree of Precision)**.**
A quadrature formula $I_n(f)$ that approximates $I(f)$ has degree of precision m if it is exact for all polynomial f of degree $\leq m$, and $I_n(f) \neq I(f)$ for some polynomial f of degree $m+1$.

Theorem 4.3 (Equivalent Definition of degree of precision)**.**
The following are equivalent.

1. $I_n(f)$ has degree of precision m.
2. $\int_a^b f(x)\,\mathrm{d}\,x - I_n(f) = 0 \ \ \forall f(x) = p_m(x)$ and $\int_a^b f(x)\,\mathrm{d}\,x - I_n(f) \neq 0$ for some $f(x) = p_{m+1}(x)$
3. $\int_a^b x^i\,\mathrm{d}\,x - I_n(x^i) = 0 \ \ \forall i = 0, 1, \ldots, m$ and $\int_a^b x^{m+1}\,\mathrm{d}\,x - I_n(x^{m+1}) \neq 0$

4.3 Simpson's Rule

Using the similar technique, by setting $n = 2$, a three equidistant point $x_0 = a, x_1 = \frac{1}{2}(a+b)$ and $x_2 = b$, we have

$$\begin{aligned} p_2(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}f(x_1) + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}f(x_2) \\ &= \frac{1}{2h^2}[(x-x_1)(x-x_2)f(x_0) - 2(x-x_0)(x-x_2)f(x_1) + (x-x_0)(x-x_1)f(x_2)] \end{aligned}$$

where $h = \frac{1}{2}(b-a) = x_{i+1} - x_i, i = 0, 1$ is the step size, and

$$E_2(f) = \frac{f'''(\xi)}{3!}\psi(x) = (x-x_0)(x-x_1)(x-x_2)f[x, x_0, x_1, x_2]$$

It can be shown that $I_2(f)$ is

$$I_2(f) = \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)]$$

and

$$E_2(f) = -\frac{h^5}{90}f^{(4)}(\xi)$$

for some $\xi \in [x_0, x_2]$.
It is then clear the degree of precision of Simpson's rule is $m = 3$.

4.3.1 Proof of Simpson's error formula

let F be an anti-derivative of f, i.e. $F' = f$. WLOG, $a + b = 0$, set $x = \frac{b-a}{2}$ then we are interested in the error expression

$$E_4(f) = F(x) - F(-x) - \frac{x}{3}(f(x) + 4f(0) + f(-x))$$

This has derivatives

$$g'(x) = \frac{2}{3}(f(x) - 2f(0) + f(-x)) - \frac{x}{3}(f'(x) - f'(-x)) \tag{1}$$

$$g''(x) = \frac{1}{3}(f'(x) - f'(-x)) \qquad - \frac{x}{3}(f''(x) + f''(-x)) \tag{2}$$

$$g'''(x) = \qquad - \frac{x}{3}(f'''(x) - f'''(-x)) \quad (\#). \tag{3}$$

Consequently, as $g(0) = g'(0) = g''(0) = g'''(0) = 0$, we have, by extended mean value theorem[11]

$$\frac{g(x)}{x^5} = \frac{g'(x_1)}{5x_1^4} = \cdots = \frac{g'''(x_3)}{(5 \cdot 4 \cdot 3x_3^2)}$$

By substituting (#), we have

$$\begin{aligned}\frac{g(x)}{x^5} = \frac{-\frac{x_3}{3}(f'''(x_3) - f'''(-x_3))}{60x_3^2} &= -\frac{1}{90} \cdot \frac{f'''(x_3) - f'''(-x_3)}{2x_3} \\ &= \frac{1}{90} \cdot f^{(4)}(x_4) \quad \text{By MVT}\end{aligned}$$

Other error bounds are

- $|E_2(f)| \leq \frac{h^5}{12}|f^{(4)}(\xi)|$. This is derived by $E_2(f) = \int_{x_0}^{x_1} E_2(x)\,\mathrm{d}\,x + \int_{x_1}^{x_2} E_2(x)\,\mathrm{d}\,x$ and apply integral MVT on each integral.[12]

[11] Extended mean value theorem:

[12] Integral of $\int_{x_0}^{x_n}(x-x_0)\cdot(x-x_n)\,\mathrm{d}\,x$ can be more easily evaluated by substitution: $x = x_0 + th$, where $h = x_{i+1} - x_i$ is the step size. Another useful substitution is to make the above integrand symmetric.

- $|E_2(f)| = -\frac{h^5}{12}[\frac{1}{3}f^{(4)}(\eta_1) - \frac{1}{5}f^{(4)}(\eta_2)]$. This is proved by (1) Taylor expansion of $f(x)$ around x_1 with remainder containing 4th power derivative. Odd power derivative are eliminated due to symmetry. (2) Next, consider Taylor expansion with remainder containing 4th power derivative around x_0 and x_2, both evaluated at x_1, to obtain an expression for f'' , as odd power derivatives are again eliminated. The fourth power derivative is bounded by IVT. Together the result is proved.

4.4 General Newton Cotes formula

Theorem 4.4 (Newton Cotes formula)**.**
Suppose that

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

denotes the $(n+1)$-point Newton-Cotes formula with

$$x_0 = a, \;\; x_n = b, \;\; h = \frac{b-a}{n}$$

There exists $\xi \in (a, b)$ such that

$$\int_a^b f(x)\,\mathrm{d}\,x = \sum_{i=0}^{n} a_i f(x_i) + \frac{h^{n+3} f^{(n+2)}(\xi)}{(n+2)!} \int_0^n t^2(t-1)\cdot(t-n)\,\mathrm{d}\,t$$

if n is **even** and $f \in C^{n+2}[a, b]$; and

$$\int_a^b f(x)\,\mathrm{d}\,x = \sum_{i=0}^{n} a_i f(x_i) + \frac{h^{n+2} f^{(n+1)}(\xi)}{(n+1)!} \int_0^n t(t-1)\cdot(t-n)\,\mathrm{d}\,t$$

if n is **odd** and $f \in C^{n+1}[a, b]$.
Therefore, when n is an **even** integer, the degree of precision is $n+1$; when n is odd, the degree of precision is only n.

4.5 Composite Numerical Integration

Composite numerical integration is performed in a piecewise approach.

4.5.1 Composite Trapezoidal Rule

Let

$$h = \frac{b-a}{n}, \;\; n \geq 1$$

and

$$x_i = a + ih, \;\; i = 0, 1, \ldots, n$$

Then composite trapezoidal rule is stated as following

$$\begin{aligned} I(f) = \int_a^b f(x)\,\mathrm{d}\,x &= \sum_{i=1}^{n} \int_{x_i-1}^{x_i} f(x)\,\mathrm{d}\,x \\ &= \sum_{i=1}^{n} \left\{ \frac{h}{2}[f(x_{i-1} + f(x_i)] - \frac{h^3}{12} f''(\xi_i) \right\} \\ &:= T_n(f) + E_{T_n}(f) \end{aligned}$$

for some $\xi_i \in [a, b]$. Denoting $f_i := f(x_i)$, we have

$$T_n(f = \frac{h}{2}[(f_0 + f_1) + (f_1 + f_2) + \cdots + (f_{n-1} + f_n)] = \frac{h}{2}[f_0 + 2\sum_{i=1}^{n-1} f_i + f_n]$$

and

$$\begin{aligned} E_{t_n}(f) = I(f) - T_n(f) &= \sum_{i=1}^{n} -\frac{h^3}{12} f''(\xi_i) \\ &= -\frac{nh^3}{12} \left[\frac{1}{n} \sum_{i=1}^{n} f''(\xi_i) \right] \\ &= -\frac{b-a}{12} h^2 \left[\frac{1}{n} \sum_{i=1}^{n} f''(\xi_i) \right] \\ &= -\frac{b-a}{12} h^2 f''(\eta) \quad \text{by Intermediate Value Theorem} \end{aligned}$$

From the error formula E_{T_n}, the composite trapezoidal rule has degree of precision 1.

4.5.2 Composite Simpson's Rule

For $n \geq 2$ and n even, define $h = \frac{b-a}{n}$ and $x_i = a + ih, f_i = f(x_i)$ where $i = 0, 1, \ldots, n$.
The composite Simpson's rule is stated as following

$$\begin{aligned} I(f) = \int_a^b f(x)\,\mathrm{d}\,x &= \sum_{i=1}^{\frac{n}{2}} \int_{x_{2i-2}}^{x_{2i}} f(x)\,\mathrm{d}\,x \\ &= \sum_{i=1}^{\frac{n}{2}} \left[\frac{h}{3}(f_{2i-2} + 4f_{2i-1} + f_{2i}) - \frac{h^5}{90} f^{(4)}(\xi_i) \right] \end{aligned}$$

where $\xi_i \in [x_{2i-2}, x_{2i}]$.
It can be shown that the expression for composite Simpson's rule, $I_{S_n}(f)$, is

$$I_{S_n}(f) = \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + \cdots + 2f_{n-2} + 4f_{n-1} + f_n)$$

and the error $E_{S_n}(f)$ is

$$\begin{aligned} E_{s_n}(f) &= I(f) - I_{S_n}(f) \\ &= \sum_{i=1}^{\frac{n}{2}} \left[-\frac{h^5}{90} f^{(4)}(\xi_i) \right] \\ &= -\frac{h^5}{90} \sum_{i=1}^{\frac{n}{2}} f^{(4)}(\xi_i) \\ &= -\frac{h^5}{90} \frac{n}{2} f^{(4)}(\eta) \quad \text{by Intermediate Value Theorem} \\ &= -\frac{h^4(b-a)}{180} f^{(4)}(\eta) \end{aligned}$$

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

4.6 Romberg Integration

The Romberg Integration is carried out in a recursive manner. The base case $R_{k,1}$ is calculated first.
Let

$$h_k = \frac{b-a}{2^{k-1}} \quad k = 1, 2, \ldots$$

which corresponds to $n_k = 2^{k-1}$ subintervals.
We denote the Composite Trapezoida; rule with h_k above by $R_{k,1}$, i.e.,

$$R_{k,1} = \frac{h_k}{2} \sum_{i=1}^{2^{k-1}} [f(a + (i-1)h_k) + f(a + ih_k)]$$

It turns out that we can reduce calculation by using $R_{k-1,1}$ in R_k:

$$R_{k,1} = \frac{1}{2}[R_{k-1,1} + h_{k-1} \sum_{i=1}^{2^{k-2}} f(a + (2i-1)h_k)]$$

It can be shown that

$$\int_a^b f(x)\,\mathrm{d}\,x - R_{k,1} = \sum_{i=1}^{\infty} K_i h_k^{2i} = K_1 h_k^2 + \sum_{i=2}^{\infty} K_i h_k^{2i}$$

where $K_i (i = 1, 2, \ldots)$ are **constants** *independent* of h_k and they depend only on $f^{(2i-1)}(a)$ and $f^{(2i-1)}(b)$.
With this result, we can accelerate the covergence of the Composite Trapezoidal rule $R_{k,1}$, by eliminating the terms involving h_k^2, using expression $R_{k,1}$ and $R_{k+1,1}$. It can be shown that

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

and

$$\int_a^b f(x)\,\mathrm{d}\,x = R_{k+1,2} + O(h_k^4)$$

whose degree of precision improves.
In general, we have

$$R_{k,j} = \frac{4^{j-1}R_{k,j-1} - R_{k-1,j-1}}{4^{j-1}-1}$$

and

$$\int_a^b f(x)\,\mathrm{d}\,x = R_{k,j} + O(h_k^{2j})$$

In general, if we want to calculate $R_{n,n}$, we follow the following table

$$\begin{bmatrix} R_{1,1} & & & \\ R_{2,1} & R_{2,2} & & \\ \vdots & \vdots & \ddots & \\ R_{n,1} & R_{n,2} & \cdots & R_{n,n} \end{bmatrix}$$

and compute column by column.

4.7 Adaptive Quadrature Methods

Adaptive Quadrature is based on Simpson's rule. The objective is to bound the error of approximation to $\int_a^b f(x)\,\mathrm{d}\,x$ by a specified error $\varepsilon > 0$.
The first step is to apply Simpson's rule with step size $h = \frac{b-a}{2}$, producing

$$\int_a^b f(x)\,\mathrm{d}\,x = S(a,b) - \frac{h^5}{90}f^{(4)}(\mu) \quad \text{for some } \mu \in (a,b)$$

where $S(a,b) = \frac{h}{3}[f(a) + 4f(a+h) + f(b)]$. The next step is to apply composite Simpson's rule with $n = 4$, and step size $\frac{h}{2} = \frac{b-a}{4}$,giving

$$\int_a^b f(x)\,\mathrm{d}\,x = S(a, \frac{a+b}{2}) + S(\frac{a+b}{2}, b) - \frac{1}{16}\frac{h^5}{90}f^{(4)}(\mu')$$

By assumption $\mu = \mu'$, we have

$$\frac{h^5}{90}f^{(4)}(\mu) = \frac{16}{15}[S(a,b) - S(a, \frac{a+b}{2}) - S(\frac{a+b}{2}, b)]$$

and this estimation of error gives

$$\left|\int_a^b f(x)\,\mathrm{d}\,x - S(a, \frac{a+b}{2}) - S(\frac{a+b}{2}, b)\right| \approx \frac{1}{15}|S(a,b) - S(a, \frac{a+b}{2}) - S(\frac{a+b}{2}, b)|$$

Therefore, we check whether RHS $< \varepsilon$. If yes, stop.
If not, distribute the error into two subintervals, and try bound the error, in this case $\frac{1}{2}\varepsilon$ using the same technique. Repeat this step until the error is contained.

4.8 Gaussian Quadrature

Gaussian quadrature is of the form of $\sum_{i=1}^{n} w_i f(x_i)$, where $x_i, i = 1, 2, \ldots, n$ are chosen from interval $[a, b]$ and $w_i, i = 1, \ldots, n$ are chosen to minimise the expected error by maximising the degree of precision.
To obtain Gaussian quadrature of degree n on interval $[-1, 1]$, we solve the equation

$$E_n(x^i) = I(x^i) - I_n(x^i), \quad i = 0, 1, \ldots, 2n-1$$

Case 1: $n = 1$.
Then $I_1(f) = 2f(0)$.
Case 2: $n = 2$.
Then $I_2(f) = f(\frac{-1}{\sqrt{3}}) + f(\frac{1}{\sqrt{3}})$
Case 3: $n = 3$.
Then $I_3(f) = \frac{5}{9}f(-\sqrt{\frac{3}{5}}) + \frac{8}{9}f(0) + \frac{5}{9}f(\sqrt{\frac{3}{5}})$.
Case n:
Solve

$$\int_{-1}^{1} x^j \,\mathrm{d}\, x = \sum_{i=1}^{n} w_i x_i^j \quad j = 0, 1, \ldots, 2n-1$$

for $w_i, x_i, i = 1, 2, \ldots, n$.
For general interval $[a, b]$, introduce linear change of variable

$$x = \frac{(b+a) + t(b-a)}{2} \quad -1 \le t \le 1$$

Thus,

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

where $g(t) = f\left(\frac{(b+a)+t(b-a)}{2}\right)$.

5 Solutions of Nonlinear Equations

There are three numerical methods for finding the root of $f(x) = 0$, namely

- Bisection Method
- Fixed Point Method
- Newton-Rhapson Method

5.1 Bisection Method

Intermediate value theorem is an important theorem used in bisection method, to demonstrate the existence of root.

Theorem 5.1 (Intermediate value theorem)**.**
If the function $f(x)$ is defined and continuous on the real interval $[a, b]$ and if

$$f(a)f(b) < 0$$

then there exists at least one real number x^* such that

$$f(x^*) = 0 \text{ and } a < x^* < b$$

The uniqueness of the root can be guaranteed if we have

$$f'(x) > 0 \text{ or } f'(x) < 0 \text{ on } [a, b]$$

provided that $f'(x)$ exists on $[a, b]$.

Theorem 5.2 (Algorithm: Bisection Method)**.**
Let $f(x)$ be continuous on the interval $[a, b]$.
Suppose $f(a_0)f(b_0) < 0$, then for $k = 1, 2, \ldots$, compute

$$x_k = \frac{1}{2}(a_{k-1} + b_{k-1})$$

and the new interval I_k is

$$I_k = (a_k, b_k) = \begin{cases} (x_k, b_{k-1}) & \text{if } f(x_k)f(b_{k-1}) < 0 \\ (a_{k-1}, x_k) & \text{if } f(x_k)f(a_{k-1}) < 0 \end{cases}$$

Note that the solution lies in $I_k = (a_k, b_k)$.
If $f(x_k) = 0$, then we have computed a solution. If $f(x_k) \neq 0$ we can continue till $b_k - a_k \leq$ a specified tolerance and x_k will be an approximate solution.

Theorem 5.3 (Error of Bisection Method)**.**
From the estimate of solution after the kth bisection,

$$x_k = \frac{1}{2}(a_{k-1} + b_{k-1})$$

Since the exact solution is bounded in the interval $[a_{k-1}, b_{k-1}]$, the error estimates for x_k the approximated solution to $f(x) = 0$, is

$$|x^* - x_k| \leq \frac{b_{k-1} - a_{k-1}}{2} = \frac{b_0 - a_0}{2^k}$$

where x^* is the exact solution to $f(x) = 0$.

Remark:

- Error of bisection method will always converges to 0.
- The rate of convergence is independent of the function $f(x)$.
- Error of estimate x_k is said to be within t correct places, if $|x^* - x_k| < \frac{1}{2} \times 10^{-t}$ is satisfied.

5.2 Fixed Point Method

Definition 5.1 (Fixed Point)**.**
Consider the equation $g(x) = x$ where g is the given function and x is in some finite interval. A solution η satisfies

$$g(\eta) = \eta$$

is called a **fixed point** of g.

Theorem 5.4 (Existence and Uniqueness of Fixed Point)**.**
If g is continuous on $[a, b]$ and $a \leq g(x) \leq b$ for all $x \in [a, b]$, then g has a fixed point $\eta \in [a, b]$.
If, in addition, $g'(x)$ exists on (a, b) and

$$|g'(x)| \leq \alpha < 1, \quad \forall x \in (a, b) \quad (\#)$$

then g has a **unique** fixed point $\eta \in [a, b]$.

Theorem 5.5 (Error Bound)**.**
Suppose (#) holds, an error bound is

$$|x_n - \eta| \leq \alpha^n |x_0 - \eta|$$

A computable error bound is

$$|x_0 - \eta| \leq \frac{\alpha^n}{1 - \alpha} |x_0 - x_1|$$

Remark: The important step is to find a qualifying g before applying fixed point method.

5.3 Order of Convergence

Definition 5.2 (Order of Convergence)**.**
In general, a sequence $\{x_n\}_{n=1}^{\infty}$ is said to **converge with order** k to the limit η if there is a constant $\lambda \neq 0$ such that

$$\lim_{n\to\infty} \frac{|x_{n+1} - \eta|}{|x_n - \eta|^k} = \lambda$$

An iterative method

$$x_{n+1} = g(x_n) \quad n = 0, 1, \ldots$$

is said to be of **order** k if $\{x_n\}$ converges with order k to the fixed point of g.

Theorem 5.6 (Degree of Convergence)**.**
Suppose that the iterative method

$$x_{n+1} = g(x_n) \quad n = 0, 1, \ldots$$

converges to a fixed point η of g.
If $g'(\eta) = g''(\eta) = \cdots = g^{(k-1)}(\eta) = 0$ and $g^{(k)}(\eta) \neq 0$, and $g^{(k)}$ is continuous on an interval containing η, then

$$\lim_{n\to\infty} \frac{|x_{n+1} - \eta|}{|x_n - \eta|^k} \neq 0$$

that is, the convergence $x_n \to \eta$ is of order k.

5.4 Newton-Rhapson's Method

Definition 5.3 (Newton-Rhapson's Method)**.**
Newton-Rhapson's algorithm computes the solution of $f(x) = 0$ with tolerance ε. It follows the steps below:

1. Choose an initial approximation x_0.
2. For $k = 0, 1, 2, \ldots$

   $$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

   Stop if

   $$\frac{|x_{k+1} - x_k|}{|x_k|} < \varepsilon$$

   Otherwise, continue with next k.

Remark: For this algorithm, it is required that $f'(\eta) \neq 0$ to achieve quadratic convergence.

Definition 5.4 (Mutiplicity of Solution)**.**
In general, if $f(\eta) = f^{(1)}(\eta) = \cdots = f^{(m-1)}(\eta) = 0$ and $f^{(m)}(\eta) \neq 0$, we say that η is a solution of **multiplicity** m of $f(x) = 0$.

Specifically, if $f(\eta) = 0$ and $f'(\eta) \neq 0$, η is a **simple solution**. Otherwise, η is a multiple solution.

Definition 5.5 (Modified Newton Rhapson's Method)**.**
If η is a solution of $f(x) = 0$ of multiplicity m, then the iterative scheme is

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