Numerical Analysis

Sai Saandeep.S, Dr. Sivaram

2023-08-03

Contents

Chapter 1

Introduction

1.1 Why Numerical Analysis?

1.2 Representing Numbers on a Machine

1.3 Condition Number of a Problem

Consider a function in one variable $f : \mathbb{R} \to \mathbb{R}$. Condition number for a function $f(x)$ tells about the error amplification of a function $f(x)$ i.e., for a given error in input x, how much is the error in the output $f(x)$.

Absolute Condition Number κ_{abs} of the function $f(x)$ is defined as:

$$\kappa_{\text{abs}} = \frac{\text{Absolute Change in Output}}{\text{Absolute Change in Input}} = \lim_{\delta x \to 0} \left| \frac{f(x+\delta x) - f(x)}{x + \delta x - x} \right| = |f'(x)| \quad (1.1)$$

Relative Condition Number κ_r of the function $f(x)$ is defined as:

$$\kappa_r = \frac{\text{Relative Change in Output}}{\text{Relative Change in Input}} = \lim_{\delta x \to 0} \frac{\left| \frac{f(x+\delta x) - f(x)}{f(x)} \right|}{\left| \frac{x+\delta x - x}{x} \right|} = \left| \frac{x}{f(x)} f'(x) \right| \quad (1.2)$$

Now what if the function has multiple inputs? Or What if the function has multiple outputs?

Examples:-

1. Input 2 numbers $a, b \in \mathbb{R}$ and then find $f(a, b) = a + b$?. This problem takes 2 inputs- a, b and one output $f(a, b)$.

2. Find the roots of a polynomial $a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$. We are inputting the vector $\begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_n \end{bmatrix}^T$ and the output is x in this case.
3. Given a matrix $A \in \mathbb{R}^{m\times n}$. Input a vector $\mathbf{x} \in \mathbb{R}^{n\times 1}$ and then find $f(\mathbf{x}) = A\mathbf{x} \in \mathbb{R}^{m\times 1}$?
4. Solve the linear system $A\mathbf{x} = \mathbf{b}$ where $A \in \mathbb{R}^{m\times n}$, $\mathbf{x} \in \mathbb{R}^{n\times 1}$ and $\mathbf{b} \in \mathbb{R}^{m\times 1}$. Inputs are $A, \mathbf{b}$ and output is a vector $\mathbf{x}$.

To accommodate these cases, a generalized definition of a (relative) condition number κ_r for a function $f : X \to Y$ where $X \subset \mathbb{R}^{m\times 1}$ and $Y \subset \mathbb{R}^{n\times 1}$ is shown below:

$$\kappa_r = \lim_{r\to 0} \sup_{\|x\|_q \le r} \frac{\frac{\|f(x+\delta x)-f(x)\|_p}{\|f(x)\|_p}}{\frac{\|\delta x\|_q}{\|x\|_q}} \tag{1.3}$$

where $p, q \in \mathbb{N}$ and $\|.\|_p$ denotes the vector $p-$ norm.

1.3.1 Vector Norms(Recap)

For a vector x in the vextor space X over a field F, $\|.\| : F \to R$ is defined such that:

1. $\|x\| \ge 0 \quad \forall x \in X.$
2. $\|\alpha x\| = |\alpha|\,, \quad \forall x \in X, \quad \alpha \in F$
3. $\|x + y\| \le \|x\| + \|y\|, \quad \forall x, y \in X.$
4. $\|x\| = 0 \iff x = 0$

Let $x = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T$. Different possible vector norms which satisfy the above conditions are:

1. Euclidean norm (2-norm)
$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} \tag{1.4}$$
2. Supremum norm(max. norm)
$$\|x\|_{\max} = \|x\|_\infty = \max_{1\le i\le n} |x_i| \tag{1.5}$$
3. 1-norm
$$\|x\|_1 = \sum_{i=1}^{n} |x_i| \tag{1.6}$$

4. p-norm

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}} \tag{1.7}$$

NOTE:- Supremum norm of x is p-norm of x as $p \to \infty$

Proof:- From the definition,

$$\lim_{p\to\infty} \|x\|_p = \lim_{p\to\infty} \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$$

$$\left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}} \leq \left(n \max_{1\leq i\leq n} |x_i|^p \right)^{\frac{1}{p}} = n^{\frac{1}{p}} \max_{1\leq i\leq n} |x_i|$$

$$\left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}} \geq \left(\max_{1\leq i\leq n} |x_i|^p \right)^{\frac{1}{p}} = \max_{1\leq i\leq n} |x_i|$$

From the above 2 inequalities, we can say that:

$$\max_{1\leq i\leq n} |x_i| \leq \|x\|_p \leq n^{\frac{1}{p}} \max_{1\leq i\leq n} |x_i|$$

As $p \to \infty$, $\quad n^{\frac{1}{p}} \max_{1\leq i\leq n} |x_i| \to \max_{1\leq i\leq n} |x_i|$. Therefore, by using sandwich theorem, we can say that

$$\|x\|_p = \max_{1\leq i\leq n} |x_i|$$

1.3.2 Examples on finding Condition number

1. Let $f(a, b) = a + b$. Find the condition number of this problem?

The inputs are a, b. Let the inputs have an error δa, δb respectively.

$$\text{Relative error in input} = \frac{\left\| \begin{bmatrix} a+\delta a \\ b+\delta b \end{bmatrix} - \begin{bmatrix} a \\ b \end{bmatrix} \right\|_p}{\left\| \begin{bmatrix} a \\ b \end{bmatrix} \right\|_p}$$

For simplicity, let us consider 2-norm. Any norm can be used in fact. Therefore,

$$\text{Relative error in input} = \frac{\sqrt{\delta a^2 + \delta b^2}}{\sqrt{a^2 + b^2}}$$

The output $f(a+\delta a, b+\delta b) = a+b+\delta a+\delta b$. Therefore,

$$\text{Relative Error in output} = \frac{|(a+b+\delta a+\delta b)-(a+b)|}{|a+b|} = \frac{|\delta a+\delta b|}{|a+b|}$$

The relative condition number is:

$$\kappa_r = \lim_{r\to 0} \sup_{\left\| \begin{bmatrix} \delta a \\ \delta b \end{bmatrix} \right\|_2 \leq r} \frac{\frac{|\delta a+\delta b|}{|a+b|}}{\frac{\sqrt{\delta a^2+\delta b^2}}{\sqrt{a^2+b^2}}}$$

$$\implies \kappa_r = \lim_{r\to 0} \sup_{\left\| \begin{bmatrix} \delta a \\ \delta b \end{bmatrix} \right\|_2 \leq r} \frac{|\delta a+\delta b|}{\sqrt{\delta a^2+\delta b^2}} \cdot \frac{\sqrt{a^2+b^2}}{|a+b|}$$

To calculate

$$\lim_{r\to 0} \sup_{\left\| \begin{bmatrix} \delta a \\ \delta b \end{bmatrix} \right\|_2 \leq r} \frac{|\delta a+\delta b|}{\sqrt{\delta a^2+\delta b^2}}$$

we assume that $\delta a = \alpha\cos\theta$ and $\delta b = \alpha\sin\theta$ where $\alpha > 0$ and $0 \leq \theta < 2\pi$.

Therefore, we have:

$$\lim_{r\to 0} \sup_{\left\| \begin{bmatrix} \delta a \\ \delta b \end{bmatrix} \right\|_2 \leq r} \frac{|\delta a+\delta b|}{\sqrt{\delta a^2+\delta b^2}} = \lim_{r\to 0} \sup_{\alpha<r} \frac{|\alpha\cos\theta+\alpha\sin\theta|}{\alpha} = \lim_{r\to 0} \sup_{\alpha<r} |\cos\theta+\sin\theta| = \sqrt{2}$$

Thus, the condition number for adding 2 numbers is:

$$\kappa_r = \frac{\sqrt{2(a^2+b^2)}}{|a+b|} \leq \sqrt{2} \text{ (if } a,\, b > 0)$$

(as $|a+b| \geq \sqrt{a^2+b^2}$ for $a, b \in \mathbb{R}^+$)

For $a, b > 0$, we can clearly see that the condition number is bounded above by $\sqrt{2}$. In other words, **addition is well-conditioned.**

By performing a similar exercise, we can show that the **subtraction is ill-conditioned** as for $\frac{a}{b} \to 1$, $\kappa_r \to \infty$.

Multiplication and division operations are also ill-conditioned.

2. Condition number on finding roots of the polynomial $x^2 - 2x + 1$.

Chapter 2

Numerical Linear Algebra

2.1 Columnspace, Nullspace and all

Consider a matrix $A \in \mathbb{R}^{m \times n}$ defined as:

$$A = \begin{bmatrix} - & r_1^T & - \\ - & r_2^T & - \\ & \vdots & \\ - & r_m^T & - \end{bmatrix} = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & & | \end{bmatrix}$$

where $r_i \in \mathbb{R}^{n \times 1}$ for $1 \leq i \leq m$ are the rows and $a_i \in \mathbb{R}^{m \times 1}$ for $1 \leq i \leq n$ are the columns of A.

Columnspace of a matrix A is the span(linear combination) of columns of A. Also called as Range of A.

$$\text{Range}(A) = \text{Columnspace}(A) = \{Ax : x \in \mathbb{R}^{n \times 1}\} \tag{2.1}$$

$$Ax = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \sum_{i=1}^{n} a_i x_i$$

Rowspace of a matrix A is the span(linear combination) of rows of A.

$$\text{Rowspace}(A) = \{A^T y : y \in \mathbb{R}^{m \times 1}\} \tag{2.2}$$

$$A^T y = \begin{bmatrix} | & | & & | \\ r_1 & r_2 & \cdots & r_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^{n} r_i y_i$$

Nullspace of a matrix A is defined as follows:

$$\text{Nullspace}(A) = \{z \in \mathbb{R}^{n\times 1} : Az = 0\} \tag{2.3}$$

NOTE:-

1. A linear system $Ax = b$ has a solution ONLY IF $b \in \text{Range}(A)$.
2. Dimension of Range(A) is the number of linearly independent columns of A or the column rank of A. Similarly, the dimension of Rowspace(A) is the row rank or the number of independent rows of A.
3. For a matrix A, row rank = column rank = rank$\leq \min(m, n)$.
4. Nullspace of a matrix is orthogonal to row space of a matrix.i.e, Given any vector $z \in$ Nullspace(A) and $w \in$ Rowspace(A) , z is orthogonal to w.

Proof:- Let $w \in$ Rowspace(A), then $\exists y \in \mathbb{R}^{n\times 1}$ such that $w = A^T y$.

Also as $z \in$ Nullspace(A), we have $Az = 0$.

Therefore,

$$\langle w, z\rangle = w^T z = y^T Az = 0$$

Thus, nullspace of a matrix is orthogonal to row space of a matrix.

5. **Rank-Nullity Theorem:** Dimension of Nullspace(A)+Rank(A) = n = No. of columns of A

2.2 Matrix Norms

Consider a matrix $A \in \mathbb{R}^{m\times n}$. Just like how we have defined a vector norm, we could have defined an "element wise matrix norm" as follows:

$$\|A\|_p^* = \left(\sum_{i=1}^{n} |A_{ij}|^p\right)^{\frac{1}{p}} \tag{2.4}$$

But this definition of norm does not satisfy the **submultiplicative property**. We are interested in this property as this dictates the convergence of iterative schemes.

A matrix norm is said to be submultiplicative if for any matrices $A \in \mathbb{R}^{m\times k}$ and $B \in \mathbb{R}^{k\times n}$, we have

$$|AB\| \leq \|A\|\|B\| \tag{2.5}$$

Consider the case

$$A = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$$

and $p \to \infty$, Therefore we have

$$\|A\|_{\infty}^{*} = \max_{1 \le i \le m, 1 \le j \le n} |A_{ij}| = 2$$

.

$$A^2 = \begin{bmatrix} 8 & 8 \\ 8 & 8 \end{bmatrix}$$

Therefore,

$$\|A^2\|_{\infty}^{*} = 8$$

We can clearly see that:

$$\|A^2\|_{\infty}^{*} = 8 \geq \|A\|_{\infty}^{*} \cdot \|A\|_{\infty}^{*} = 2 \times 2 = 4$$

which violates the submultiplicative property.

Hence, we define a p-norm of matrix which satisfies submultiplicative property as follows.

$$\|A\|_p = \sup_{x \in \mathbb{R}^n \{\mathbf{0}\}} \frac{\|Ax\|_p}{\|x\|_p} = \sup_{\|y\|_p = 1} \|Ay\|_p \tag{2.6}$$

p-norms are submultiplicative.

PROOF:- From the definition of p-norm,

$$\begin{aligned} \|AB\|_p &= \sup_{x \in \mathbb{R}^n \{\mathbf{0}\}} \frac{\|ABx\|_p}{\|x\|_p} \\ &= \sup_{x \in \mathbb{R}^n \{\mathbf{0}\}} \frac{\|ABx\|_p}{\|Bx\|_p} \frac{\|Bx\|_p}{\|x\|_p} \\ &\leq \left[\sup_{x \in \mathbb{R}^n \{\mathbf{0}\}} \frac{\|ABx\|_p}{\|Bx\|_p} \right] \cdot \left[\sup_{x \in \mathbb{R}^n \{\mathbf{0}\}} \frac{\|Bx\|_p}{\|x\|_p} \right] \\ &= \|A\|_p \cdot \|B\|_p \end{aligned}$$

$$\therefore \|AB\|_p \leq \|A\|_p \cdot \|B\|_p$$

Using this property, we can say that

$$\|A^n\|_p \leq \|A\|_p^n \tag{2.7}$$

$\|A\|_1$ = Maximum of column sum of absolute values.

$\|A\|_\infty$ = Maximum of row sum of absolute values.

2.3 Condition number of Matrix vector products

Consider a matrix $A \in \mathbb{R}^{m\times n}$ and a vector $x \in \mathbb{R}^{n\times 1}$. Assume that there is no error in representing A. We are interested in finding the condition number of the Matrix-Vector product $f(x; A) = Ax$.

From the definition of condition number, we can write the condition number κ_r of the matrix vector product as:

$$\kappa_r = \lim_{r\to 0} \sup_{\|\delta x\|_q \le r} \frac{\frac{\|A(x+\delta x)-Ax\|_p}{\|Ax\|_p}}{\frac{\|x+\delta x-x\|_q}{\|x\|_q}}$$

For simplicity let us choose $p = q$. Therefore,

$$\kappa_r = \lim_{r\to 0} \sup_{\|\delta x\|_p \le r} \frac{\|A\delta x\|_p}{\|\delta x\|_p} \frac{\|x\|_p}{\|Ax\|_p}$$

From the definition of matrix p-norm, we can say that:

$$\lim_{r\to 0} \sup_{\|\delta x\|_p \le r} \frac{\|A\delta x\|_p}{\|\delta x\|_p} = \|A\|_p$$

Therefore the condition number of the matrix vector product is:

$$\kappa_r = \frac{\|A\|_p \|x\|_p}{\|Ax\|_p} \tag{2.8}$$

From Sub-multiplicative property, as $\|Ax\|_p \ge \|A\|_p \|x\|_p$, we can show that $\kappa_r \ge 1$.

Case-1:- $A \in \mathbb{R}^{m\times n}$ is a fat matrix$(m < n)$

From Rank-Nullity Theorem, we know that

$$\text{Dimension of Nullspace}(A)+\text{Rank}(A) = n$$

We know that $\text{Rank}(A) \le \min(m, n) \implies$ Rank $(A) \le m$ as A is a fat matrix.

$$\implies \text{Dimension of Nullspace}(A) \ge n - m$$

$\implies \exists$ a non-zero vector $z \in$ Nullspace(A) i.e., $\exists z \in \mathbb{R}^{n\times 1}$ such that $Az = 0$.

From the definition of condition number as in equation(), we have: $\kappa_r(A, x) = \frac{\|A\|_p \|x\|_p}{\|Ax\|_p}$

If $x = z$ then as $Az = 0$, we have $\|Az\|_p = 0$. Therefore $\kappa_r \to \infty$.

Thus, multiplication by a fat matrix is **highly ill-conditioned.**

Case-2:- A is an invertible square matrix

From the definition of condition number as in equation(), we have:

$$\kappa_r(A, x) = \frac{\|A\|_p \|x\|_p}{\|Ax\|_p} \tag{2.9}$$

$$= \frac{\|A\|_p \|A^{-1}(Ax)\|_p}{\|Ax\|_p} \tag{2.10}$$

$$\tag{2.11}$$

From submultiplicative property of matrix norms, we can write $\|A^{-1}(Ax)\|_p \leq \|A^{-1}\|_p \|Ax\|_p$. Therefore,

$$\kappa_r(A, x) = \frac{\|A\|_p \|A^{-1}(Ax)\|_p}{\|Ax\|_p} \tag{2.12}$$

$$\implies \kappa_r(A, x) \leq \frac{\|A\|_p \|A^{-1}\|_p \|Ax\|_p}{\|Ax\|_p} \tag{2.13}$$

$$\tag{2.14}$$

$$\kappa_r(A, x) \leq \|A\|_p \|A^{-1}\|_p \tag{2.15}$$

Therefore condition number is bounded above by $\|A\|_p \|A^{-1}\|_p$ which is independent of vector x.

Define Condition number of matrix A as

$$\kappa_p(A) = \|A\|_p \|A^{-1}\|_p$$

.

From submultiplicative property we can show that $\kappa_p(A) \geq 1$.

Let us find condition number for some special matrices. Let us consider $A = Q$ to be an orthogonal/ unitary matrix.

As Q is an orthogonal matrix $Q^T Q = I \implies Q^T = Q^{-1}$.

Therefore

$$\|Qx\|_2^2 = (Qx)^T Qx = x^T Q^T Qx = x^T x = \|x\|_2^2$$

Thus,

$$\|Qx\|_2 = \|x\|_2 = \|Q^T x\|_2$$

From definition of 2-norm of a matrix, we have,

$$\|Q\|_p = \sup_{x \in \mathbb{R}^n \{\mathbf{0}\}} \frac{\|Qx\|_p}{\|x\|_p} \tag{2.16}$$

$$\implies \|Q\|_2 = \sup_{x \in \mathbb{R}^n \{\mathbf{0}\}} \frac{\|x\|_2}{\|x\|_2} \tag{2.17}$$

$$\implies \|Q\|_2 = 1 \tag{2.18}$$

Therefore,

$$\|Q^T\|_2 = 1 \tag{2.19}$$

The condition number of Q is therefore,

$$\kappa_2(Q) = \|Q\|_p \|Q^{-1}\|_p = \|Q\|_2 \|Q^T\|_2 = 1 \tag{2.20}$$

2.4 Solving Linear Systems

Consider the linear system $Ax = b$ where $A \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^{n \times 1}$ and $\mathbf{b} \in \mathbb{R}^{m \times 1}$. Inputs are $A, \mathbf{b}$ and output is a vector $\mathbf{x}$.

In this course, we assume that A is a square(i.e., $m = n$) and invertible matrix (so that we have a unique x). Our goal is to find that unique x which satisfies the system of equations.

Let's go step by step. Let us consider special matrices first and then discuss about a general matrix A.

2.4.1 Special Matrices

1. If A is a unitary matrix, then $AA^T = I = A^T A$.

$$Ax = b \implies A^T Ax = A^T b \implies x = A^T b$$

   To get x, we need to perform n^2 multiplications and $n(n-1)$ additions.

2. If $A = U$ is an upper triangular matrix.

Let $Ux = b$ in matrix form be:

$$\begin{bmatrix} U_{11} & U_{12} & U_{13} & \cdots & U_{1n} \\ 0 & U_{22} & U_{23} & \cdots & U_{2n} \\ 0 & 0 & U_{33} & \cdots & U_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & U_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix} \tag{2.21}$$

The last row of U corresponds to the equation

$$U_{nn}x_n = b_n \tag{2.22}$$

Now as U is invertible, all the diagonal elements are non-zero i.e., $U_{ii} \neq 0 \ \forall i \in \{1, 2, \cdots, n\}$.

Therefore

$$U_{nn}x_n = b_n \implies x_n = \frac{b_n}{U_{nn}} \tag{2.23}$$

Similarly, we have:

$$U_{(n-1),(n-1)}x_{n-1} + U_{(n-1),n}x_n = b_{n-1} \implies x_{n-1} = \frac{b_{n-1} - U_{(n-1),n}x_n}{U_{(n-1),(n-1)}} \tag{2.24}$$

Using induction, we can prove that for $i \in \{1, 2, \cdots, n-1\}$, we have:

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} U_{i,j}x_j}{U_{ii}} \tag{2.25}$$

To calculate x_i(for $i \in \{1, 2, \cdots, n\}$), we need to perform 1 division, $(n-i)$ multiplications and $(n-i)$ additions/subtractions.

Therefore to calculate the output vector x, we need to perform

- $\sum_{i=1}^{n} 1 = n$ divisions
- $\sum_{i=1}^{n}(n-i) = \frac{n^2-n}{2}$ multiplications
- $\sum_{i=1}^{n}(n-i) = \frac{n^2-n}{2}$ additions/subtractions
- In total $n + \frac{n^2-n}{2} + \frac{n^2-n}{2} = n^2$ operations are required to get x. In other words "Computational Complexity is n^2"

NOTE:-

1. Loosely speaking computational complexity of an algorithm means the number of operations performed in the algorithm.
2. We say that $f(n) \in \mathcal{O}(g(n))$

2.4.2 General Case

We know that to solve $Ax = b$, we convert the Augmented matrix $[A|b]$ to its Row Reduced Echelon Foem(RREF) by doing elementary row operations on A to get an Identity matrix.

Instead of fully converting to an Identity matrix, let us convert to an upper triangular matrix, say U.i.e., we do row operations to get

$$Ux = c$$

from $Ax = b$ where U is an upper triangular matrix and c is the vector obtained by performing corresponding operations on b.

Let $Ax = b$ in matrix form be:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & \cdots & A_{1n} \\ A_{21} & A_{22} & A_{23} & \cdots & A_{2n} \\ A_{31} & A_{32} & A_{33} & \cdots & A_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & A_{n3} & \cdots & A_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix} \tag{2.26}$$

To convert A to an upper triangular matrix U (i.e., to have $U_{ij} = 0$ for $i > j$) using row operations, we need to make all elements below the diagonal to be zero. To achieve this, we firstly make all elements below A_{11} in the 1st column to be zero using the below operations on j^{th} row ($j > 1$) :(We assume that $A_{11} \neq 0$)

$$R_{j,1} = 0 \tag{2.27}$$

$$R_{j,2:n} \leftarrow R_{j,2:n} - \frac{A_{j1}}{A_{11}} R_{1,2:n} \tag{2.28}$$

$$b_j \leftarrow b_j - \frac{A_{j1}}{A_{11}} b_1 \tag{2.29}$$

We proceed like this to make all elements below the diagonal to be zero. Consider k^{th} step in this process. Let the matrix in this step be:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & \cdots & A_{1,k} & \cdots & A_{1n} \\ 0 & A_{22} & A_{23} & \cdots & A_{2,k} & \cdots & A_{2n} \\ 0 & 0 & A_{33} & \cdots & A_{3,k} & \cdots & A_{3n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_{k,k} & \cdots & A_{k,n} \\ 0 & 0 & 0 & \cdots & A_{k+1,k} & \cdots & A_{k+1,n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_{n,k} & \cdots & A_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix} \tag{2.30}$$

The row operations, to be done on j^{th} row, so that all elements below A_{kk} become zero (in k^{th} column) is given as follows: For $j > k$ and $A_{k,k} \neq 0$

$$R_{j,k} = 0 \tag{2.31}$$

$$R_{j,k+1:n} \leftarrow R_{j,k+1:n} - \frac{A_{j,k}}{A_{k,k}} R_{k,k+1:n} \tag{2.32}$$

$$b_j \leftarrow b_j - \frac{A_{j,k}}{A_{k,k}} b_k \tag{2.33}$$

Let us now calculate the number of operations required to be done for converting A to RREF. In 1st step, for the row operation

$$R_{j,2:n} \leftarrow R_{j,2:n} - \frac{A_{j1}}{A_{11}} R_{1,2:n}$$

for a particular row-j, we need to do 1 division, $n-1$ multiplications and $n-1$ additions. There are $n-1$ such rows. Therefore, for the row operation on all these $n-1$ rows, $n-1$ divisions, $(n-1)^2$ multiplications and $(n-1)^2$ additions have to be done.

For k^{th} step($1 \leq k \leq n-1$), row operation

$$R_{j,k+1:n} \leftarrow R_{j,k+1:n} - \frac{A_{j,k}}{A_{k,k}} R_{k,k+1:n}$$

on row-j for $k+1 \leq j \leq n$, requires 1 division, $(n-k)$ multiplications and $(n-k)$ additions. Therefore, we need to perform $n-k$ divisions, $(n-k)^2$ multiplications and $(n-k)^2$ additions.

Therefore, the number of operations required to convert A to U are:

1. $\sum_{k=0}^{n-1}(n-k) = \sum_{k=1}^{n} k = \frac{n(n-1)}{2}$ Divisions
2. $\sum_{k=0}^{n-1}(n-k)^2 = \sum_{k=1}^{n} k^2 = \frac{n(n-1)(2n-1)}{6}$ Multiplications
3. $\sum_{k=0}^{n-1}(n-k)^2 = \sum_{k=1}^{n} k^2 = \frac{n(n-1)(2n-1)}{6}$ Additions

Total number of operations to convert from A to $U = \frac{n(n-1)}{2} + 2\frac{n(n-1)(2n-1)}{6}$

Now, let us calculate the number of operations required to get c from b. Consider the operation on the k^{th} step($1 \leq k \leq n-1$):

$$b_j \leftarrow b_j - \frac{A_{j,k}}{A_{k,k}} b_k$$

. This operation requires $(n-k)$ multiplications and $(n-k)$ additions. Note that we have already calculated $\frac{A_{j,k}}{A_{k,k}}$ while converting A to U. Hence, no separate division operation is involved.

Therefore, number of operations on RHS are:

1. $\sum_{k=0}^{n-1}(n-k) = \sum_{k=1}^{n} k = \frac{n(n-1)}{2}$ Multiplications.
2. $\sum_{k=0}^{n-1}(n-k) = \sum_{k=1}^{n} k = \frac{n(n-1)}{2}$ Additions

After converting $Ax = b$ to $Ux = c$, we have seen that solving $Ux = c$ requires n divisions, $\frac{n(n-1)}{2}$ additions and $\frac{n(n-1)}{2}$ multiplications.

Therefore, for solving $Ax = b$:

1. Number of divisions required $= \frac{n(n+1)}{2} + n$
2. Number of multiplications required $= \frac{n(n-1)(2n-1)}{6} + \frac{n(n-1)}{2} + \frac{n(n-1)}{2}$
3. Number of additions required $= \frac{n(n-1)(2n-1)}{6} + \frac{n(n-1)}{2} + \frac{n(n-1)}{2}$

We can see that multiplications & addition operations are the most expensive(time consuming)

Therefore, the total number of operations $= 2\left[\frac{n(n-1)(2n-1)}{6} + n(n-1)\right] + \frac{n(n+1)}{2} + n$

In other words Total number of operations $\in \mathcal{O}(n^3)$

Now the question arises. Where will this fail?

1. We have assumed that $A_{11} \neq 0$ in the first step. This may not always hold true. Also, there is a possibility that $A_{kk} = 0$ in the intermediate steps also. If this occurs then we can't proceed further.
2. Precision issues might occur. Consider solving the system of equations.

$$2^{-60}x_1 + x_2 = 1 \tag{2.34}$$
$$x_1 + x_2 = 2 \tag{2.35}$$

The equations can be written in matrix form as:

$$\begin{bmatrix} 2^{-60} & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{2.36}$$

The solution is $x_1 \approx 1$ and $x_2 \approx 1$.

Let us try to solve this with the help of method described in this section. First, we do the row operations $R_1 \leftarrow R_2 - \frac{1}{2^{-60}} R_1 \implies R_1 \leftarrow R_2 - 2^{60} R_2$ to get:

$$\begin{bmatrix} 2^{-60} & 1 \\ 0 & 1 - 2^{60} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 - 2^{60} \end{bmatrix} \tag{2.37}$$

$1 - 2^{60}$ and $2 - 2^{60}$ is represented on the 64 bit machine as 2^{-60}. Therefore, the machine represents the equations as:

$$\begin{bmatrix} 2^{-60} & 1 \\ 0 & -2^{60} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -2^{60} \end{bmatrix} \tag{2.38}$$

This implies that $x_2 = 1$ and $2^{-60} x_1 + x_2 = 1 \implies x_1 = 0$. But this is **significantly** different from the exact solution.

If we have had an infinite precision machine, then we would have got accurate results. But since, we have only finite precision machines, we have this issue.

Let us calculate condition number of this matrix. The condition number as calculated by MATLAB using `cond()` function gives condition number as 2.6180 which is a relatively small condition number. This means that small changes in input don't give very large changes in output. Hence it is a well conditioned problem. Therefore, condition number is not an issue.

Therefore, Algorithm has to be blamed here

We say that an algorithm is stable if it gives almost correct answer to a well conditioned problem. As this gives a completely different answer, we say that this algorithm is unstable.(even if it doesn't encounter zero in the diagonal)

How to overcome these issues? Let us think this way. For the first issue, if a zero diagonal entry is encountered, let us try swapping row with a row which has a non-zero element in that column. Now the question arises which row? We tend to swap with the row which has largest absolute value. Instead of swapping only when zero is encountered, we do it in every step.

Let us consider the example problem, we were discussing earlier.

$$\begin{bmatrix} 2^{-60} & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{2.39}$$

In first column, the maximum value is 1. Thus, we swap 1st and 2nd rows.

$$\begin{bmatrix} 1 & 1 \\ 2^{-60} & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \tag{2.40}$$

We do a row operation $R_2 \leftarrow R_2 - 2^{-60} R_1$ to get:

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 - 2^{-60} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 - 2^{-59} \end{bmatrix} \tag{2.41}$$

$1-2^{-60}$ and $1-2^{-59}$ is represented on the 64 bit machine as 1. Therefore, the machine represents the equations as:

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \tag{2.42}$$

The solution to these system of equations give $x_2 = 1$ and $x_1 = 1$.

This method in which we swap rows is called **partial pivoting**. Partial pivoting is seen to be stable in practice.

Eg: Solve the system using partial pivoting

$$x_1 + 2x_2 + 3x_3 = 6 \tag{2.43}$$
$$x_1 + 2x_2 + 4x_3 = 7 \tag{2.44}$$
$$x_1 - 2x_2 + x_3 = 0 \tag{2.45}$$

A much more better way is including a column swap as well. The process is decribed in the below example. This process is called **Complete Pivoting** which is stable but has more computation cost(due to multiple swaps/ comparisions)

Eg: Solve the system using complete pivoting.

$$x_1 + 2x_2 + 3x_3 = 6 \tag{2.46}$$
$$x_1 + 2x_2 + 4x_3 = 7 \tag{2.47}$$
$$x_1 - 2x_2 + 10x_3 = -11 \tag{2.48}$$

ADD SOLUTION

Order of computational cost: Complete Pivoting > Partial Pivoting > No Pivoting

Order of Stability: Complete Pivoting > Partial Pivoting > No Pivoting

2.5 Solving Overdetermined Systems

2.6 Solving Underdetermined Systems

2.7 Iterative Methods for solving Linear Systems

Previously, we have seen "direct methods" to solve linear systems like Gauss-Jordan elimination, Partial and complete pivoting. They can be solved exactly with ∞ precision machines. But the computational cost is high.

For lesser computational cost, we use iterative methods. But the issue is that we can not solve exactly using these methods always.

Chapter 3

Interpolation

3.1 Motivation - Interpolation vs. Approximation

If the exact form of $f(x)$ is known, then we have full information about $f(x)$ i.e., Derivatives etc., But what if the exact form is not known?

Given points $\{x_i\}_{i=1}^n$ and functional values at those points $f(x_1), f(x_2), \dots, f(x_n)$, we wish to find an Approximation to $f(x)$. One way to approximate a function is by *interpolating* it.

> We say that $p(x)$ is an interpolant to $f(x)$ at $\{x_i\}_{i=1}^n$ if $p(x_i) = f(x_i)$ for $1 \leq i \leq n$.

Eg: A step function $p(x) = f(x_i)$ for $x \in \left[\frac{x_i+x_{i-1}}{2}, \frac{x_i+x_{i+1}}{2}\right]$.

The step function, though it is an interpolant, we don't prefer it. The main issue is that it is not continuous. We prefer in some practical applications for the interpolant to be differentiable.

Note that not all approximations are interpolants.

Eg: A polynomial approximation to $\sin x$ is a truncated Taylor series after a few terms(say till degree $2n+1$). This approximation is not an interpolant as it won't intersect $\sin x$ at $2n+2$ points.

Assume $f(x)$ to be continuous. Consider the sequence of interpolants $\{P_n(x)\}_{n=1}^{\infty}$ converging to $f(x)$ on $[a, b]$ such that $P_n(x) = f(x) \quad \forall x \in \{x_1, x_2, \dots, x_n\}$

$$\int_a^b f(x)\,dx \approx \int_a^b P_n(x)\,dx \tag{3.1}$$

$$\frac{df}{dx}(x) \approx \frac{dP_n}{dx}(x) \tag{3.2}$$

Equation(3.1) holds true if $\{P_n(x)\}_{n=1}^{\infty}$ converges to $f(x)$ *uniformly.*

Equation(3.2) holds true if $P_n'(x)$ exists and $\{P_n'(x)\}_{n=1}^{\infty}$ converges to $f'(x)$ *uniformly.*

In this course, we consider the interpolants $p(x) \in C^{\infty}([a,b])$. A simplest such interpolant would be a polynomial.

3.2 Lagrange Interpolation

3.2.1 Motivation

Consider $p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_mx^m$ to be a polynomial interpolant for $f(x)$ with **node points** as $\{x_i\}_{i=1}^{n}$. Therefore,

$$p(x_i) = f(x_i) \implies a_0 + a_1x_i + a_2x_i^2 + \cdots + a_mx_i^m = f(x_i) \quad \forall i \in \{1, 2, \cdots, n\}$$

.

$$\implies \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{bmatrix}_{n\times(m+1)} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix}_{(m+1)\times 1} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix}_{n\times 1} \tag{3.3}$$

Let

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{bmatrix}, \quad \bar{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} \text{ and } \quad \bar{f} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix}$$

$$\implies X\bar{a} = \bar{f}$$

Now the $\bar{a}$ has the coefficients of the interpolant. To find the coefficients, we need to solve the linear system.

If $m + 1 < n$ then X is a thin matrix. i.e., we have lesser number of variables than equations. Therefore, solution may not exist. i.e., we may not be able to find $p(x)$.

If $m + 1 > n$ then X is a fat matrix. Infinitely many polynomial interpolants exist.

If $m+1=n$ then X is a square matrix. X is Vandermonde matrix. It can be shown that:

$$\det(X) = \prod_{1 \le i < j \le n} (x_i - x_j)$$

If $x_i's$ are distinct then $\det(X) \neq 0 \implies X$ is invertible.

$\implies X\bar{a} = \bar{f}$ has a unique solution for $m+1=n$.

$\implies p(x)$ interpolates $f(x)$ uniquely if $\deg(p(x)) = n-1$.

Thus, the minimum degree of the interpolant polynomial is $n-1$.

$p(x)$ interpolates $f(x)$ uniquely if $\deg(p(x)) = n-1$. Thus, solving the equation $X\bar{a} = \bar{f}$. But there are issues in solving the linear system like this.

- Computational complexity in solving the linear system $\mathcal{O}(n^3)$.
- Condition number of X grows exponentially in n. This is not preferred as this might cause large errors with small error in input(say due to roundoff errors etc.,)

To overcome these problems, we use Lagrange interpolation.

3.2.2 Lagrange Interpolant

Consider

$$g(x_i) = \begin{cases} 1 & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \tag{3.4}$$

for $i, j \in \{1, 2, \dots, n\}$. We are interested to find a polynomial interpolant for this. Let us consider the case of $n=3$ points and $j=2$. i.e., $g(x_1) = g(x_3) = 0$ and $g(x_2) = 1$ for simplicity. The least degree of the polynomial interpolant $p(x)$ is $n-1=2$. By intuition, we can say that $(x-x_1)(x-x_3)$ is a factor of interpolant $p(x)$. As $p(x_2) = g(x_2) = 1$, we can say that the interpolant $p(x)$ is:

$$p(x) = \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} \tag{3.5}$$

For n points, we have the interpolant polynomial to $g(x)$ as:

$$p(x) = l_j(x) = \prod_{\substack{i=0 \\ i \neq j}} \frac{x - x_i}{x_j - x_i} \tag{3.6}$$

$l_j(x)$ is a polynomial of degree $n-1$ such that:

$$l_j(x_i) = \delta_{ij}$$

Now consider n node points $\{f(x_i)\}_{i=1}^n$ and consider the polynomial $p(x)$ defined as:

$$p(x) = \sum_{j=1}^{n} f(x_j) l_j(x) \tag{3.7}$$

$p(x)$ is a polynomial of degree atmost $n-1$. Now,

$$p(x_i) = \sum_{j=1}^{n} f(x_j) l_j(x_i) = \sum_{j=1}^{n} f(x_j)\delta_{ij} = f(x_i)$$

This implies that $p(x)$ is an interpolant. This is known as *Lagrange Interpolant*.

3.3 Choice of Nodes

3.3.1 Motivation

Now after finding the interpolant, the next question to be asked is how accurate is the interpolant? Immediate obvious answer would be the numer of points chosen. But are there any other factors which determine the accuracy of the interpolant?

Example: Consider the function $f(x) = \frac{1}{1+25x^2}$ and the interpolation nodes as uniform nodes from [-1,1].

INSERT CODE HERE

We can see that the interpolant is not converging to $f(x)$ uniformly. There are some ***boundary effects***

Thus, selection of node points is also important. The question to ask now is what set of nodes guarantees uniform convergence?

3.3.2 Fundamental Theorem of Polynomial Interpolation

Let $f(x)$ be a smooth function on [-1,1]. Let $P_n(x)$ be a polynomial interpolant to $f(x)$ at $\{x_k\}_{k=0}^n$ with atmost degree n. Then $\exists\, \zeta \in [-1,1]$ such that:

$$e(x) = f(x) - P_n(x) = \frac{f^{(n+1)}(\zeta)}{(n+1)!} \prod_{k=0}^{n} (x - x_k) \tag{3.8}$$

Proof:- Define

$$w(x) := \prod_{k=0}^{n} (x - x_k)$$

and

$$g_x(t) := f(t) - P_n(t) - \left(\frac{f(x) - P_n(x)}{w(x)}\right) w(t) \tag{3.9}$$

where the subscript x denotes that x is fixed.

$$g_x(x) = f(x) - P_n(x) - \left(\frac{f(x) - P_n(x)}{w(x)}\right) w(x) = 0$$

$$g_x(x_j) = f(x_j) - P_n(x_j) - \left(\frac{f(x) - P_n(x)}{w(x)}\right) w(x_j)$$

As $P_n(x)$ is an interpolant to $f(x)$ at nodes $\{x_i\}_{i=0}^n$, $P_n(x_j) = f(x_j)$ and by definition $w(x_j) = 0$. Therefore, for $j \in \{0, 1, 2, ..., n\}$, we have $g_x(x_j) = 0$.

$g_x(t)$ is smooth in the interval (-1,1).

Define intervals

$$I_1 = [x_0, x_1] \tag{3.10}$$
$$I_2 = [x_1, x_2] \tag{3.11}$$
$$\vdots \tag{3.12}$$
$$I_k = [x_{k-1}, x_k] \tag{3.13}$$
$$I_{k+1} = [x_k, x] \tag{3.14}$$
$$I_{k+2} = [x, x_{k+1}] \tag{3.15}$$
$$I_{k+3} = [x_{k+1}, x_{k+2}] \tag{3.16}$$
$$\vdots \tag{3.17}$$
$$I_{n+1} = [x_{n-1}, x_n] \tag{3.18}$$

According to Rolle's theorem, $g_x'(t)$ has atleast $n+1$ zeros on $[-1, 1]$. Again according to Rolle's theorem, we can say that $g_x''(t)$ has atleast n zeros in $[-1, 1]$. If we keep on using Rolle's theorem for further $n-1$ times, we can show that $g_x^{(n+1)}(t)$ has atleast 1 zero on $[-1, 1]$ i.e., $\exists$ a $\zeta \in [-1, 1]$ such that $g_x^{(n+1)}(\zeta) = 0$

$$g_x^{(n+1)}(\zeta) = f^{(n+1)}(\zeta) - P_n^{(n+1)}(\zeta) - \left(\frac{f(x) - P_n(x)}{w(x)}\right) w^{(n+1)}(\zeta)$$

As $P_n(t)$ is an $n^{th} degree polynomial, $P_n^{(n+1)}(\zeta) = 0$.

$$w(t) = \prod_{k=0}^{n} (t - x_k) \implies w^{(n+1)}(t) = (n+1)!$$

Therefore

$$0 = f^{(n+1)}(\zeta) - 0 - \left(\frac{f(x) - P_n(x)}{w(x)}\right) (n+1)!$$

$$\implies e(x) = f(x) - P_n(x) = \frac{f^{(n+1)}(\zeta)}{(n+1)!}w(x) = \frac{f^{(n+1)}(\zeta)}{(n+1)!}\prod_{k=0}^{n}(x - x_k)$$

What if $x \in [a, b]$ instead of $x \in [-1, 1]$?

We can use a linear mapping from $[a, b]$ to $[-1, 1]$.

3.3.3 Different Possible types of nodes

Now the goal is to find the nodes which minimise the maximum absolute interpolation error i.e.,

$$\min \max_{x\in[-1,1]} |e(x)|$$

. In general, we can also try finding nodes which minimise p-norm of the interpolation error i.e., $\min\|e(x)\|_p$ where:

$$\|e(x)\|_p = \left(\int_{-1}^{1} |e(x)|^p \, dx\right)^{\frac{1}{p}}$$

and

$$\lim_{p\to\infty} \|e(x)\|_p = \max_{x\in[-1,1]} |e(x)|$$

Now the issue is it is difficult to know about $f^{(n+1)}(\zeta)$ as $f(x)$ is not known always. Now the best thing we can do is to find nodes which minimise

$$\min \max_{x\in[-1,1]} \left|\prod_{k=0}^{n}(x - x_k)\right| \text{ or } \min\|\prod_{k=0}^{n}(x - x_k)\|_p$$

.

It turns out that Legendre nodes minimise $||w(x)||_2$, Chebyshev nodes of first kind minimise $||w(x)||_\infty$ and Chebyshev nodes of second kind minimises $||w(x)||_1$.

3.3.3.1 Legendre Nodes

Monic Legendre polynomials are defined as:

$$q_0(x) = 1, \; q_1(x) = x$$

$q_n(x)$ is a monic polynomial of degree n such that:

$$\int_{-1}^{1} q_n(x)q_m(x) \, dx = 0 \quad \forall\, m \neq n \tag{3.19}$$

First few monic Legendre polynomials are:

$$\begin{aligned} q_0(x) &= 1 \\ q_1(x) &= x \\ q_2(x) &= x^2 - \frac{1}{3} \\ q_3(x) &= x^3 - \frac{3}{5}x \\ q_4(x) &= x^4 - \frac{6}{7}x^2 + \frac{3}{35} \end{aligned}$$

The zeros of these Legendre polynomials are called Legendre nodes. Legendre nodes minimise $||w(x)||_2$.

3.3.3.2 Chebyshev nodes of first kind

Chebyshev polynomials of first kind are given by $T_n(x) = \cos(n \cos^{-1}(x))$.

First few Chebyshev polynomials are:

$$T_0(x) = 1 \tag{3.20}$$
$$T_1(x) = x \tag{3.21}$$
$$T_2(x) = 2x^2 - 1 \tag{3.22}$$
$$T_3(x) = 4x^3 - 3x \tag{3.23}$$
$$T_4(x) = 8x^4 - 8x^2 + 1 \tag{3.24}$$

An interesting property of these polynomials are $T_m(x)$ and $T_n(x)$ are orthogonal weighted by $\frac{1}{\sqrt{1-x^2}}$.

$$\int_{-1}^{1} \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}} dx = 0, m \neq n \tag{3.25}$$
$$= \pi, m = n = 0 \tag{3.26}$$
$$= \frac{\pi}{2}, m = n \neq 0 \tag{3.27}$$

Proof:-

Let $x = \cos\theta$ where $\theta \in [0, \pi]$. This implies that $dx = -\sin\theta\, d\theta = -\sqrt{1-x^2}\, d\theta$.

And $x = -1 \implies \theta = \pi$ and $x = 1 \implies \theta = 0$. Therefore,

$$\begin{aligned}\int_{-1}^{1} T_m(x)T_n(x)\frac{1}{\sqrt{1-x^2}}dx &= \int_0^{\pi} \cos(m\theta)\cos(n\theta)d\theta \\ &= \frac{1}{2}\int_0^{\pi} \cos(m+n)\theta + \cos(m-n)\theta d\theta \\ &= \frac{1}{2}\left[\sin\frac{(m+n)\theta}{m+n}\right]_0^{\pi} + \left[\sin\frac{(m-n)\theta}{m-n}\right]_0^{\pi} \\ &= 0, \text{ if } m \neq n\end{aligned}$$

Similarly with appropriate substitution, conditions for $m = n=0 $ and $m = n 0 $ can be proved.

The zeros of Chebyshev polynomial $T_{n+1}(x)$ are given by $x_k = \cos\left(\frac{2k+1}{2n+2}\pi\right)$ where $k \in \{0, 1, 2, \cdots, n\}$.

Proof:-

Proof:-

We have

$$T_{n+1}(x) = \cos((n+1)\arccos(x))$$

$$T_{n+1}(x) = 0 \implies \cos((n+1)\arccos(x)) = 0$$

$$\implies (n+1)\arccos(x) = (2k+1)\frac{\pi}{2} \quad k \in \mathbb{Z}$$

$$\implies \arccos(x) = (2k+1)\frac{\pi}{2(n+1)} \quad k \in \mathbb{Z}$$

But as the principle range of $\arccos(x)$ is defined as $[0, \pi]$, we must restrict the values k can take.

$$0 \leq (2k+1)\frac{\pi}{2(n+1)} \leq \pi \implies 0 \leq k \leq n + \frac{1}{2}$$

But as $k \in \mathbb{Z}$, we can say that the possible values k can take are $\{0, 1, 2, \cdots, n\}$.

Therefore $\arccos(x_k) = (2k+1)\frac{\pi}{2(n+1)} \quad k \in \{0, 1, 2, \cdots, n\}$

$$\implies x_k = \cos\left((2k+1)\frac{\pi}{2(n+1)}\right) \quad k \in \{0, 1, 2, \cdots, n\}$$

Therefore the zeros of $T_{n+1}(x)$ are in the interval $[-1, 1]$ are given by $x_k = \cos\left(\frac{2k+1}{2n+2}\pi\right)$ where $k \in \{0, 1, 2, \cdots, n\}$. The number of zeros is also consistent

with the fact that as $T_{n+1}(x)$ is an $(n+1)$ th-degree polynomial, it has $(n+1)$ roots. Also, these zeros are distinct.

These zeros are called the Chebyshev nodes of first kind. They minimise $||w(x)||_\infty$

Theorem

If $f(x)$ is smooth on $[-1, 1]$, then Lagrange interpolation using the roots of Chebyshev nodes of first kind converges uniformly.

INSERT RUNGE FUNCTION-CHEBYSHEV NODES CONVERGENCE CODE

3.4 Wierstrass Approximation theorem

Suppose f is a continuous real-valued function defined on the real interval $[a, b]$. For every $\epsilon > 0$, there exists a polynomial p such that for all x in $[a, b]$, we have $|f(x) - p(x)| < \epsilon$.

Bernstein polynomial:

The Bernstein basis polynomials of degree n are defined as

$$b_{k,n}(x) = {}^nC_k \, x^k (1-x)^{n-k} \tag{3.28}$$

A linear combination of these basis polynomials can be used to obtain other polynomials. One of the properties of these polynomials is that

$$\begin{aligned} \sum_{k=0}^{n} b_{k,n}(x) &= \sum_{k=0}^{n} {}^nC_k \, x^k (1-x)^{n-k} \\ &= (x + (1-x))^n = 1 \end{aligned} \tag{3.29}$$

So these polynomials can also be considered to act as some kind of weights. Now, for approximating functions, the Bernstein Polynomial is defined as

$$B_n(f; x) = \sum_{k=0}^{n} f\left(\frac{k}{n}\right) {}^nC_k \, x^k (1-x)^{n-k} \tag{3.30}$$

Here, f is the function being approximated, n is the order of approximation & x is the point at which the approximation is made.

Proof:

To prove the theorem on closed intervals $[a, b]$, without loss of generality, we can take the closed interval as $[0, 1]$. Thus, f can be considered as a continuous real-valued function on $[0, 1]$. Since f is a continuous function, we can say that for a given $\epsilon > 0$, there exists a $\delta > 0$ such that:

$$|x - y| < \delta \implies |f(x) - f(y)| < \frac{\epsilon}{2} \quad \forall\, x, y \in [0, 1] \tag{3.31}$$

To prove that $B_n(f,x)$ converges to $f(x)$ uniformly, we show that $|B_n(f,x) - f(x)|$ has to be made small. By the definition of Bernstein's polynomial, as shown in Eqn. 3.30 and by using the property given in Eqn. 3.29, we can write $|B_n(f;x) - f(x)|$ as:

$$\begin{aligned} |B_n(f;x) - f(x)| &= \left| \sum_{k=0}^{n} \left(f\left(\frac{k}{n}\right) - f(x) \right) {}^nC_k \, x^k (1-x)^{n-k} \right| \\ &\leq \sum_{k=0}^{n} \left| f\left(\frac{k}{n}\right) - f(x) \right| {}^nC_k \, x^k (1-x)^{n-k} \end{aligned}$$

Now, if we consider $y = \dfrac{k}{n}$ in Eqn. 3.31 and if $\left|\dfrac{k}{n} - x\right| < \delta$, we can say that $\left| f\left(\dfrac{k}{n}\right) - f(x) \right| < \dfrac{\epsilon}{2}$. But this is not true in the entire domain. So, we partition the domain in to two sets: A and B, where A and B have the following properties:

$$\begin{aligned} A \cup B &= [0,1] \\ A \cap B &= \phi \\ A &= \{x : |k/n - x| \leq \delta, x \in [0,1]\} \\ B &= \{x : |k/n - x| > \delta, x \in [0,1]\} \end{aligned} \tag{3.32}$$

By dividing the domain into sets A and B as defined in 3.32, we can write:

$$\begin{aligned} |B_n(f;x) - f(x)| = \sum_{x \in A} \left| f\left(\frac{k}{n}\right) - f(x) \right| {}^nC_k \, x^k (1-x)^{n-k} \\ + \sum_{x \in B} \left| f\left(\frac{k}{n}\right) - f(x) \right| {}^nC_k \, x^k (1-x)^{n-k} \end{aligned}$$

From Eqns. 3.31 and 3.32, we can say that $\left| f\left(\dfrac{k}{n}\right) - f(x) \right|$ has an upper bound of $\dfrac{\epsilon}{2}$ on the set A. Therefore, we can write:

$$\begin{aligned} |B_n(f;x) - f(x)| \leq \sum_{x \in A} \left(\frac{\epsilon}{2} \right) {}^nC_k \, x^k (1-x)^{n-k} \\ + \sum_{x \in B} \left| f\left(\frac{k}{n}\right) - f(x) \right| {}^nC_k \, x^k (1-x)^{n-k} \end{aligned}$$

By using the property described in Eqn. 3.29, we can simplify the above inequality as:

$$|B_n(f,;x) - f(x)| \leq \frac{\epsilon}{2} + \sum_{x \in B} \left| f\left(\frac{k}{n}\right) - f(x) \right| {}^nC_k \, x^k (1-x)^{n-k}$$

Since f is uniformly converging in $[0,1]$, f is bounded from above. So, let the maximum value of f in the domain be M. Thus:

$$\max\{|f(x)|\} = M \quad \forall \, x \in [0,1]$$

Thus, the maximum value $\left|f\left(\frac{k}{n}\right) - f(x)\right|$ can achieve in the domain is $2M$ (considering the case where, one of them is M and the other is $-M$). Thus, we have:

$$|B_n(f;x) - f(x)| \leq \frac{\epsilon}{2} + (2M)\sum_{x\in B} {}^nC_k\, x^k(1-x)^{n-k}$$

Now, in set B, $\left|\frac{k}{n} - x\right| > \delta$. Thus, we get:

$$\frac{(k/n - x)^2}{\delta^2} > 1$$

Multiplying this to the second term of RHS, we get:

$$|B_n(f;x) - f(x)| \leq \frac{\epsilon}{2} + (2M)\sum_{x\in B} \frac{(k/n - x)^2}{\delta^2} {}^nC_k\, x^k(1-x)^{n-k}$$

The second term can now be written as:

$$\frac{2M}{\delta^2 n^2}\sum_{x\in B} (k - nx)^2\, {}^nC_k\, x^k(1-x)^{n-k}$$

The summation term is equivalent to computing the variance of a binomial distribution with parameters n & x. The variance is given by $nx(1-x)$. Thus, we get:

$$|B_n(f;x) - f(x)| \leq \frac{\epsilon}{2} + \frac{2M}{\delta^2 n^2} nx(1-x)$$

We know that using AM $\geq$ GM:

$$x(1-x) \leq \frac{1}{4}$$

Thus, we have:

$$|B_n(f;x) - f(x)| \leq \frac{\epsilon}{2} + \frac{M}{2\delta^2 n}$$

Now, let us define a quantity N such that the above condition holds true for all $n > N$, which gives us $\frac{1}{n} < \frac{1}{N}$. Thus:

$$|B_n(f;x) - f(x)| \leq \frac{\epsilon}{2} + \frac{M}{2\delta^2 N}$$

If we choose the N such that:

$$\frac{M}{2\delta^2 N} = \frac{\epsilon}{2}$$

giving us:

$$|B_n(f;x) - f(x)| \leq \frac{\epsilon}{2} + \frac{\epsilon}{2}$$
$$|B_n(f;x) - f(x)| \leq \epsilon$$

Thus, we have for all $n > N$, where $N = \frac{M}{\delta^2 \epsilon}$, we have

$$|B_n(f;x) - f(x)| \leq \epsilon \tag{3.33}$$

i.e., the Bernstein Polynomial $B_n(f;x)$ uniformly converges to $f(x)$ for all x in the domain $[0,1]$.

NOTE:- 1. A sequence $\{a_n\}_{n\geq 0}$ converges to a **algebraically** if $\exists N > 0$ such that $\forall\, n > N$

$$|a_n - a| < \frac{k}{n^\alpha} \text{ for some } k,\ \alpha > 0$$

Examples:- a. $a_n = 1 + \frac{1}{n^2}$ converges to 1 algebaically as

$$|a_n - 1| = \frac{1}{n^2} < \frac{10}{n^2}$$

b. $a_n = e^{-n}$ converges to 0 algebraically as

$$|a_n| = e^{-n} < \frac{1}{n}\ \forall n \geq 1$$

2. A sequence $\{a_n\}_{n\geq 0}$ converges to a **geometrically** if $\exists N > 0$ such that $\forall\, n > N$

$$|a_n - a| < kc^n \text{ for some } k > 0, |c| < 1$$

3. Given any c and α (such that $|c| < 1$ and $\alpha > 0$), there exists $N = N(c,\alpha) > 0$ such that $\forall n > N$

$$c^n < \frac{1}{n^\alpha}$$

This implies that all geometrically convergent series are algebraically convergent.

4. Error in Bernstein polynomial $B_n(x)$ approximation goes down as $\frac{1}{n}$. This means that $B_n(x)$ uniformly converges to $f(x)$ algebraically at the rate $\frac{1}{n}$.

5. If $f(x)$ is smooth then the Chebyshev interpolant is Geometrically convergent to f. This convergence is also known as Spectral Convergence.

Consider the function $f(x) = \frac{1}{1+25x^2}$. Let us consider the analytic continuation of $f(x)$ on a compact set $\Omega \in \mathbb{C}^2$ such that $[-1,1] \subset \Omega$ which is $\frac{1}{1+25z^2}$. This is analytic on any compact set Ω which doesn't contain $\pm i/5$.

Chapter 4

Quadratures

In this chapter, we see how to find a definite integral numerically. ## Motivation Consider a continuous function $f(x)$ in an interval $[a, b]$. We are interested in finding the integral $\int_a^b f(x)\,dx$.

If we don't know $f(x)$ at all points, then how to find integral? Based on what we have studied till now, one way we can think of to approximate the integral is to first find a polynomial interpolant $p(x)$ using the known points and then find the integral $\int_a^b p(x)\,dx$. We can say that

$$\int_a^b f(x)\,dx \approx \int_a^b p(x)\,dx \tag{4.1}$$

But we have seen that there are issues like non-uniform convergence based in node distribution in the previous chapter. Therefore, we avoid this method. We resort to other methods to find the definite integral numerically.

Now, even if we know the exact form of the function at all points in the interval (a, b), we can not always find a closed form for the anti-derivative of the function.

Example:- Consider $f(x) = \exp(-x^2)$. We can't find the exact form of $\int f(x)\,dx = \int e^{-x^2}\,dx$. We can find $\int_{-\infty}^{\infty} e^{-x^2}\,dx$ using multivariable calculus methods say exactly. But for some random finite interval (a, b), it is impossible to find the exact value. To get the approximate value, we use numerical methods.

In some cases, though finding exact value of the integral is possible, it is extremely difficult to find it.

4.1 Different Schemes

4.1.1 Basic Schemes

ATTACH FIGURES FOR EACH SCHEME

1. Lower/Left Riemann Sum:

$$\int_a^b f(x)\,dx \approx \sum_{i=0}^{n-1} f(x_i)\left(\frac{b-a}{n}\right) \tag{4.2}$$

2. Upper/Right Riemann Sum:

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} f(x_i)\left(\frac{b-a}{n}\right) \tag{4.3}$$

3. Trapezoidal Rule: Instead of rectangles, let us approximate the areas by first dividing into trapeziums and summing up areas.

$$\int_a^b f(x)\,dx \approx \frac{1}{2}h\left[\sum_{i=1}^{n}(f(x_{i-1})+f(x_i))\right] \tag{4.4}$$

We can see that the trapezoidal rule is an average of Left and Right Riemann Sums.

4. Midpoint/Rectangular rule

$$\int_a^b f(x)\,dx \approx h\sum_{i=0}^{n-1} f\left(\frac{x_i+x_{i+1}}{2}\right) \tag{4.5}$$

4.1.2 Accuracy of these schemes

Now the question arises - How accurate are these approximations?

4.1.3 Simpson's Rule

4.1.4 Trapezoidal Rule with End point Corrections

4.2 Euler-Maclaurian Formula

4.2.1 Introduction

4.2.2 Formula and Derivation

4.2.3 Higher Order Quadratures

4.3 Gaussian Quadratures

Chapter 5

Root Finding Algorithms

5.1 Motivation

Let $f(x)$ be a continuous function. We are interested in solving $f(x) = 0$ for x,i.e., we are interested in finding x^* such that $f(x^*) = 0$. But finding x^* analytically is not always easy.

Examples:-

1. Solving for x for $3x = 9$, $x^2 + 2x - 1 = 0$ are easy. We have closed form zeros for x till 4th order polynomial. Beyond 4th order closed form solution is NOT possible.
2. Solving transcendental equations like $\tan x = x$, $x = e^x$ exactly.

Considering the issues present, we need to think of solving the equations numerically.

Consider $f(x) = xe^x - 5$. We are now interested to find x^* such that $f(x^*) = 0$. Let us guess say $x^* = 1.2$. $f(1.2) = -1.015859692716143$. We need to refine this further! For this we need to find a sequence of x_k's iteratively, such that the sequence $\{x_k\}$ converges to x^*.

5.2 Bisection Method

We know from Intermediate Value theorem that if $f(x)$ is continuous on $[a, b]$ and if $f(a)f(b) < 0$, $\exists x^* \in (a, b)$ such that $f(x^*) = 0$.

Given a continuous function $f(x)$ and if we could find $a, b \in \mathbb{R}$ such that $a < b$ and $f(a)f(b) < 0$, then we can find x^* as follows:

1. Choose $x_0 = \frac{a+b}{2}$ and if
   a. $f(x_0)f(a) < 0 \implies \exists$ a $x^* \in (a, x_0)$.
   b. $f(x_0)f(a) > 0 \implies \exists$ a $x^* \in (x_0, b)$
   c. $f(x_0) = 0 \implies x^* = x_0$
2. Refine the guess by taking mean of new interval and check for convergence.
3. The stopping criteria are $|f(x)| < \epsilon$ and $|I_k| < \delta$ where
   - $|I_k|$ is the length of the interval I_k at the k^{th} step. This is found as: Interval at 0^{th} step is $I_0 = [a, b]$. Hence the length of interval I_0 is $|I_0| = b - a$.
     Let the interval at k^{th} step is I_k. Then from step 1,2 we can say that $|I_k| = \frac{|I_{k-1}|}{2}$. Therefore:
     $$|I_k| = \frac{b-a}{2^k}$$
   - ϵ and δ are user specified small tolerences.(eg 1e-10).

INSERT PICTURE HERE

INSERT CODE/PSEUDO-CODE HERE

This method described above is called as Bisection method.

When does Bisection Method fail?

In case of Bisection Method, if 2 initial points $a, b \in \mathbb{R}$ are chosen such that $f(a)f(b) < 0$, the we can guarantee that Root can be found from the Intermediate value theorem. Thus, the sequence of iterates always converge to the root.

5.3 Newton Method

In Bisection method, we need to find functional value at 2 points initially where we expect the root to lie between them. Is it possible to find the root by taking just 1 initial guess? Newton's method allows us to do this! But we need to know the value of derivative at that point.

INSERT PICTURE HERE

Pick guess value x_0. We improve this guess by finding the unique root of the linear approximation at this point. The linear approximation is the tangent at that point. The equation of tangent to $f(x)$ at $x = x_0$ is:

$$y - f(x_0) = f'(x_0)(x - x_0)$$

This intersects x axis at say $(x_1, 0)$. Therefore,

$$0 - f(x_0) = f'(x_0)(x_1 - x_0)$$

If $f'(x_0) \neq 0$, then x_1 exists. Therefore, the improved guess is

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{5.1}$$

On repeating the process, we get the improved guess iterates as:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{5.2}$$

INSERT CODE/PSEUDOCODE HERE

5.3.1 Rate of Convergence

Assume that the iterate sequence $\{x_k\} \to x^*$ where $f(x^*) = 0$. Define $e_k = x_k - x^* \implies x^* = x_k - e_k$.

We assume that $f(x)$ is a continuous function which is twice differentiable with $f''(x)$ being continuous. Writing Taylor series of $f(x^*) = 0$ about x_k gives:

$$f(x^*) = 0 = f(x_k - e_k) = f(x_k) - e_k f'(x_k) + \frac{e_k^2}{2!} f''(\zeta_k)$$

where $\zeta_k \in (\min(x_k, x^*), \max(x_k, x^*))$

$$\implies 0 = f(x_k) - (x_k - x^*) f'(x_k) + \frac{e_k^2}{2!} f''(\zeta_k)$$

$$\implies 0 = f(x_k) - (x_k - x_{k+1} + x_{k+1} - x^*) f'(x_k) + \frac{e_k^2}{2!} f''(\zeta_k)$$

$$\implies 0 = f(x_k) - (x_k - x_{k+1}) f'(x_k) - e_{k+1} f'(x_k) + \frac{e_k^2}{2!} f''(\zeta_k)$$

From equation()

$$\implies 0 = f(x_k) - \frac{f(x_k)}{f'(x_k)} f'(x_k) - e_{k+1} f'(x_k) + \frac{e_k^2}{2!} f''(\zeta_k)$$

$$e_{k+1} = \frac{e_k^2}{2!} \frac{f''(\zeta_k)}{f'(x_k)}$$

As $\zeta_k \in (\min(x_k, x^*), \max(x_k, x^*))$ and as $f''(x)$ and $f'(x)$ are continuous, $\exists\, M, m \in \mathbb{R}$ such that $f''(\zeta_k) \leq M$ and $f'(x_k) \geq m$. Therefore,

$$\frac{f''(\zeta_k)}{2f'(x_k)} \leq \frac{M}{2m} = C \text{ (say)}$$

Therefore,

$$e_{k+1} \leq Ce_k^2 \tag{5.3}$$

This implies that the rate of converge is quadratic for Newton's method. For Bisection, the convergence rate is linear.

RATE OF CONVERGENCE plot/code

5.3.2 Possibility of Non-Convergence?

Now the main question comes! When does Newton Method fail?

1. When $f'(x_k) = 0$ at any step in iteration. This means that the tangent at $x = x_k$ is parallel to x axis and hence, no root can be found from there!

Eg: $f(x) = x^2 - 1$ and $x_0 = 0$ This implies that $f'(x_0) = 2x_0 = 0$. Thus we have to change initial guess.

2. $f(x) = x^{\frac{1}{3}}$ and $x_0 = a > 0$. Root is $x^* = 0$

$f'(x) = \frac{1}{3x^{\frac{2}{3}}}$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^{\frac{1}{3}}}{\frac{1}{3x_k^{\frac{2}{3}}}} = -2x_k$$

Therefore, $x_n = (-2)^n x_0 = (-2)^n a$.

$$\lim_{n\to\infty} x_n \text{ doesn't exist.}$$

Thus, convergence is not always guaranteed in Newton's Method.

Chapter 6

Numerical Differentiation

6.1 Motivation

Given a continuous function $f(x)$. We have seen that finding anti-derivative for all functions is not possible and hence, we resort to finding approximate values of definite integrals using Numerical methods(Quadratures). But derivatives can be found for any differentiable function $f(x)$ using Chain rule etc.,

Eg:- Consider $f(x) = \cos(x^2)$ which is continuous and differentiable.

$\int_0^1 f(x)\,dx = \int_0^1 \cos(x^2)\,dx$ has no closed form. But $f'(x) = -2x\sin(x^2)$ which can be found using chain rule.

Now the question which naturally arises is: Why do we need to think of finding derivatives numerically then?

1. Consider the case where $f(x)$ is known at discrete node points $\{x_i\}_{i=0}^n$. To find derivatives at these nodes, one way is to interpolate using an interpolant polynomial $p(x)$ and then find $p'(x)$. We can find using a less computationally expensive way as will be discussed in this chapter.

2. Solving Differential Equations

   a. **Ordinary Differential Equations(ODEs):**

   Simple Pendulum:-

   ATTACH SIMPLE PENDULUM PIC WITH ITS FBD

   For simple pendulum with no damping and using small angle approximation about equilibrium position, we have the equation of motion as:

$$\frac{d^2\theta}{dt^2} + \frac{g}{l}\theta = 0 \tag{6.1}$$

with initial conditions $\theta(t=0) = \theta_0$ and $\frac{d\theta}{dt}(t=0) = 0$.

The exact solution for this ODE is known which is $\theta = \theta_0 \cos(\sqrt{g}lt)$.

But if the assumptions of small angle oscillations and no damping are relaxed then the equation of motion becomes:

$$\frac{d^2\theta}{dt^2} + C\frac{d\theta}{dt} + \frac{g}{l}\sin\theta = 0 \tag{6.2}$$

with initial conditions remaining the same. But solving this ODE analytically is impossible. Therefore, we need access to numerical methods to solve ODEs.

b. **Partial Differential Equations(PDEs):-**

Some examples are Wave Equation(Hyperbolic PDE)

$$\partial\partial^2 u \partial t^2 - c^2\nabla^2 u = 0$$

Heat/Diffusion Equation(Parabolic PDE)

$$\partial\partial u\partial t - \alpha\nabla^2 u = 0$$

Laplace Equation(Elliptic PDE)

$$\nabla^2 u = 0 \text{ on } \Omega$$

with boundary condition $u = f(x, y, z)$ on $\partial\Omega$

For specific boundaries like a rectangle, the solution can be found analytically. But for a general boundaries, it is extremely difficult to find analytical solutions. Therefore, numerical methods have to be used for solving these PDEs.

In this course, we consider solving only ODEs i.e, 1 independent variable per dependent variable.

6.2 Finite Differences

To construct approximations for derivatives, we use Taylor series. Let $f(t)$ be a real differentiable function whose values are known at $\{t_i\}_{i=0}^n = t_0 + ih$ where $h = \Delta t$ which is a constant

Define $f_i = f(t_i)$ and $f_i' = \frac{df}{dt}(t_i)$

We can write f_{j+1} in terms of f_j and derivatives of f at t_j by writing Taylor series of $f(t_{j+1})$ about $t = t_j$.

$$f_{j+1} = f(t_j + h) = f(t_j) + hf'(t_j) + \frac{h^2}{2!}f''(t_j) + ...$$
$$\frac{f_{j+1} - f_j}{h} - \frac{h}{2!}f''(t_j) - \frac{h^2}{3!}f'''(t_j) - \cdots = f'(t_j) = f'_j$$

$$f'_j = \frac{f_{j+1} - f_j}{h} + \mathcal{O}(h) \tag{6.3}$$

The above approximation is called **Forward Finite Difference Approximation**

Now consider the Taylor series of $f(t_{j-1})$ about $t = t_j$.

$$f_{j-1} = f(t_j - h) = f(t_j) - hf'(t_j) + \frac{h^2}{2!}f''(t_j) + ...$$
$$\frac{f_j - f_{j-1}}{h} + \frac{h}{2!}f''(t_j) - \frac{h^2}{3!}f'''(t_j) + \cdots = f'(t_j) = f'_j$$

$$f'_j = \frac{f_j - f_{j-1}}{h} + \mathcal{O}(h) \tag{6.4}$$

The above approximation is called **Backward Finite Difference Approximation**

We can clearly see that both Forward and Backward Finite differences are **1st order accurate.** Now the question to ask is "Can we improve the order of accuracy?"

Consider the Taylor Series Expansions of f_{j+1} and f_{j-1} about t_j.

$$f_{j+1} = f(t_j) + hf'(t_j) + \frac{h^2}{2!}f''(t_j) + \frac{h^3}{3!}f'''(t_j) + \frac{h^4}{4!}f^{(4)}(t_j) + ...$$
$$f_{j-1} = f(t_j) - hf'(t_j) + \frac{h^2}{2!}f''(t_j) - \frac{h^3}{3!}f'''(t_j) + \frac{h^4}{4!}f^{(4)}(t_j) - ...$$

Subtracting () from (), we get

$$\implies f_{j+1} - f_{j-1} = 2hf'_j + \frac{2h^3}{3!}f'''(t_j) + \frac{2h^5}{5!}f^{(5)}(t_j) + ...$$

$$\implies f_{j+1} - f_{j-1} - \frac{2h^3}{3!}f'''(t_j) - \frac{2h^5}{5!}f^{(5)}(t_j) - \cdots = 2hf'_j$$

$$f'_j = \frac{f_{j+1} - f_{j-1}}{2h} + \mathcal{O}(h^2) \tag{6.5}$$

We can see that the order of accuracy has been improved to 2. This approximation is called **Central Finite Difference Approximation.**

ATTACH CODE/ACCURACY PLOT

To improve the order of accuracy, we need to consider more "grid points". To understand this clearly, let us consider the example below.

Example: Construct Finite difference scheme to find f'_j from f_j, $f_{j\pm1}$ and $f_{j\pm2}$.

Solution:

Let

$$f'_j = af_{j+2} + bf_{j+1} + cf_j + df_{j-1} + ef_{j-2} \tag{6.6}$$

where a, b, c, d, e are to be found.

Write Taylor series of $f_{j\pm1}$ and $f_{j\pm1}$ about t_j on RHS of the above equation(), we get:

$$\begin{aligned}
f'_j = {} & a\left(f(t_j) + 2hf'(t_j) + \frac{(2h)^2}{2!}f''(t_j) + \frac{(2h)^3}{3!}f'''(t_j) + \frac{(2h)^4}{4!}f^{(4)}(t_j) + \frac{(2h)^5}{5!}f^{(5)}(t_j) + ...\right) \\
& + b\left(f(t_j) + hf'(t_j) + \frac{h^2}{2!}f''(t_j) + \frac{h^3}{3!}f'''(t_j) + \frac{h^4}{4!}f^{(4)}(t_j) + \frac{h^5}{5!}f^{(5)}(t_j) + ...\right) \\
& + cf_j \\
& + d\left(f(t_j) - hf'(t_j) + \frac{h^2}{2!}f''(t_j) - \frac{h^3}{3!}f'''(t_j) + \frac{h^4}{4!}f^{(4)}(t_j) - \frac{h^5}{5!}f^{(5)}(t_j) + ...\right) \\
& + e\left(f(t_j) - 2hf'(t_j) + \frac{(2h)^2}{2!}f''(t_j) - \frac{(2h)^3}{3!}f'''(t_j) + \frac{(2h)^4}{4!}f^{(4)}(t_j) - \frac{(2h)^5}{5!}f^{(5)}(t_j) + ...\right)
\end{aligned}$$

$$\begin{aligned}
\implies f'_j = {} & (a+b+c+d+e)f_j + h(2a+b-d-2e)f'_j + \frac{h^2}{2!}(4a+b+d+4e)f''_j \\
& + \frac{h^3}{3!}(8a+b-d-8e)f'''_j + \frac{h^4}{4!}(16a+b+d+16e)f^{(4)}_j \\
& + \frac{h^5}{5!}(32a+b-d-32e)f^{(5)}_j + \mathcal{O}(h^6)
\end{aligned}$$

Now there are 5 variables but infinitely many equations(obtained after comparing coefficients of f and its derivatives at t_j on both sides of the above equation). On comparing coefficients of $f_j, f'_j, ..., f^{(4)}_j$, we get 5 equations in 5 variables and order of accuracy in Taylor series is 5. If we take any other set of 5 equations, we get order of accuracy in Taylor series lower than 5. Thus, to get the best order of accuracy, the following equations must be true.

$$a + b + c + d + e = 0 \tag{6.7}$$

$$h(2a + b - d - 2e) = 1 \tag{6.8}$$

$$4a + b + d + 4e = 0 \tag{6.9}$$

$$8a + b - d - 8e = 0 \tag{6.10}$$

$$16a + b + d + 16e = 0 \tag{6.11}$$

On solving these set of equations, we get the values of (a, b, c, d, e) as:

$$(a, b, c, d, e) = \left(-\frac{1}{12h}, \frac{2}{3h}, 0, -\frac{2}{3h}, \frac{1}{12h}\right) \tag{6.12}$$

Therefore the finite difference approximation with 4th order accuracy is:

$$f'_j = \frac{1}{12h}(-f_{j+2} + 8f_{j+1} - 8f_{j-1} + f_{j-2}) + \mathcal{O}(h^4) \tag{6.13}$$

Chapter 7

Solving Ordinary Differential Equations Numerically

7.1 Introduction

As seen in the previous chapter, our main focus in this course is to solve ODEs i.e., 1 independent variable per dependent variable numerically.

Initial Value Problem(IVP): ODE along with boundary conditions at a single point. Typically, independent variable is time.

Example:-

$$\frac{d^2\theta}{dt^2} + C\frac{d\theta}{dt} + \frac{g}{l}\sin\theta = 0 \tag{7.1}$$

with "initial conditions" given by $\theta(t=0) = \theta_0$ and $\frac{d\theta}{dt}(t=0) = 0$.

Boundary Value Problem(BVP): Conditions known at multiple points Example:-

$$\frac{d^2\theta}{dt^2} + C\frac{d\theta}{dt} + \frac{g}{l}\sin\theta = 0 \tag{7.2}$$

with "initial conditions" given by $\theta(t=0) = \theta_0$ and $\theta(t=1) = 0$.

Note that the above ODE is 2nd order, non-linear and homogeneous ODE. If an external forcing is given, then the equation becomes non-homogeneous.

We can convert the above second order ODE to a set of 2 first order ODEs.

Define

$$\omega = \frac{d\theta}{dt}$$

Therefore equation() can be written as:

$$\frac{d\omega}{dt} + C\omega + \frac{g}{l}\sin\theta = 0$$

Therefore, equation() can be written as a system of 1st order ODEs as:

$$\begin{bmatrix}\dot{\theta}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}\omega\\ -C\omega - \frac{g}{l}\sin\theta\end{bmatrix} \tag{7.3}$$

with initial conditions as

$$\begin{bmatrix}\theta(0)\\ \omega(0)\end{bmatrix} = \begin{bmatrix}\theta_0\\ 0\end{bmatrix} \tag{7.4}$$

Similarly, we can express any nth order ODE as system of n 1st order ODEs.

Hence, we'll first look into solving a first order ODE numerically and then solve Simple Pendulum equation numerically.

7.2 Different Basic Methods

Consider solving the IVP

$$\frac{dy}{dt} = \sin(\exp(y^2)t) \tag{7.5}$$

with the initial condition $y(0) = 1$. We can see that the analytical solution is not possible. To solve the IVP numerically, let us first discretise time as $\{t_k\}_{k=0}^{n}$ where $t_0 = 0$ and $t_n = T$ time at which we are interested in finding y value and $y_{i+1} - y_i = \Delta t$ for $i \in \{0, 1, ..., n-1\}$.

ATTACH GRID DISCRETISATION PIC

We can find derivative $\frac{dy}{dt}$ at $t = t_i$(for all i), using a finite difference formula and then convert the differential equation to a set of algebraic equations. Solving these would give $y_i = y(t_i)$.

Forward difference gives:

$$\frac{dy}{dt}|_{t_i} \approx \frac{y_{i+1} - y_i}{\Delta t} = \sin(\exp(y_i^2)t_i) \tag{7.6}$$

for $i \in \{0, 1, \cdots, n-1\}$

For $i = 0$, we have:

$$y_1 = y_0 + \Delta t \sin(\exp(y_i^2)t_i) \tag{7.7}$$

y_1 is easier to compute as $y_0 = y(t = 0)$ is known. And hence y_2 can be computed and so on.

Backward difference gives:

$$\frac{dy}{dt}|_{t_i} \approx \frac{y_i - y_{i-1}}{\Delta t} = \sin(\exp(y_i^2)t_i) \tag{7.8}$$

for $i \in \{1, 2, \cdots, n\}$ For $i = 1$, we have:

$$y_1 = y_0 + \Delta t \sin(\exp(y_1^2)t_1) \tag{7.9}$$

Given y_0, y_1 can be found by using the above equation. Note that it is not as trivial as that of Forward difference case here. To get y_1, we need to use some root finding algorithm. Similarly to get y_{i+1}, we need to invoke root finding Algorithm. But in the case of Forward difference(also known as **Forward Euler Scheme**), we need not invoke any Root Finding Algorithm to find y_{i+1} and hence it is an **Explicit Scheme**

Backward Difference scheme is also referred to as **Backward Euler Scheme**. This is an **Implicit Scheme**

7.3 Linear Stability Analysis

Forward Euler seems to be more attractive because of the ease to compute the solution. But can we use this to solve all ODEs accurately?

7.4 Accuracy

7.5 Simple Pendulum - Solving System of ODEs

7.6 Simple Pendulum - Finite difference for 2nd order derivative