Lecture Notes in Scientific Computing
MA311-M

Animesh Renanse

Department of Electrical & Electronics Engineering, Indian Institute of Technology, Guwahati

January 2, 2021

Contents

[1] This section contains some details which are rigorously explained in the following chapter.

Chapter 1

Introduction

1.1 Lecture 0 : Course Introduction

Date : September 4, 2020

We'll be learning about various numerical techniques to solve some difficult problems arising in Engineering. We need a measure to analyze performance of such methods, thus performance analysis. Then we also need to design these computational procedures formally and then implement them on a computing machine.

1.1.1 Course Overview and References

- Iterative methods for non-linear equations, interpolation, numerical differentiation and integration, one step and multi-step methods for ODEs and numerical solutions of PDEs by Finite Difference Methods (FDMs).
- Kincaid and Cheney; Numerical Analysis: Mathematics of Scientific Computing
- G.D. Smith, Numerical Solutions of Partial Differential Equations
- **Software Requirements :** MATLAB

1.1.2 Evaluation Policy

- Two assignments : each 10%
- Two quizzes : each 10%

- Four assignments : each 5%
- Viva Voce : each 10%
- End Semester exam : 30%

Weekly Meeting for questions: Every Friday 10 AM - 10:55 AM

Chapter 2

Errors in Numerical Computation

Lecture 2, ***September 9, 2020***

2.1 Real Numbers, Machine Numbers, Rounding

True values are not stored exactly in a computer, it always undergo a rounding or chopping, and additional operators introduce more errors.

Real Numbers: We adopt **binary number system**.

$$x \in \mathbb{R}$$
$$x = \pm \left(b_n 2^n + b_{n-1} 2^{n-1} + \cdots \right)$$

where n is some integer and the **binary digits** b_i takes the value either 0 or 1. Hence we can write in general that,

$$x = \pm \left(b_n b_{n-1} \cdots b_0 b_{-1} b_{-2} \cdots \right)_2$$

to remind us that we are dealing with binary numbers.

Remark: It's not true that a finite decimal number will have a finite representation in a binary computer. However, to a finite binary number there always corresponds a finite decimal representation.

Machine Numbers: Two kinds,

- Floating-point number
- Fixed-point number

Floating Point number

Let $\mathbb{R}(t,s)$ be the set of **real** floating point numbers on a computer, it implies that if,

$$x \in \mathbb{R}(t,s) \quad \text{implies and is implied by} \quad \boxed{x = f.2^e}$$

where,

$$f = \pm (b_{-1}b_{-2}\cdots b_{-t})_2 \ , \ \ e = \pm (c_{s-1}c_{s-2}\cdots c_0)_2$$

Here, t is the number of binary digits allowed by the computer in the fractional part and s is the number of binary digits in the exponent. f **is referred as the mantissa and** e **is referred to as the exponent of** x.

Formula : The **largest** and **smallest** magnitude of a floating point number is given by,

$$\begin{aligned} fl_{max}(x) &= \max_{x\in\mathbb{R}(t,s)} |x| = \left(1 - 2^{-t}\right) 2^{2^s - 1} \\ fl_{min}(x) &= \min_{x\in\mathbb{R}(t,s)} |x| = 2^{-2^s} \end{aligned} \tag{2.1}$$

The trivially used terms ***overflow*** and ***underflow*** refers to the respective cases when during the course of the computation the number produced have a modulus greater than fL_{max} and less than fl_{min}.
However, we can still increase the precsion of the computation by introducing twice the number of allowed digits in the mantissa, i.e. $\mathbb{R}(2t,s)$, this is called **double-precision number**.

Fixed Point number

This is just the special case that $\boxed{x = f \implies e = 0}$.

2.1.1 Rounding

Let,

$$x \in \mathbb{R}, \ \ x = \pm \left(\sum_{k=1}^{\infty} b_{-k} 2^{-k} \right) 2^e$$

be the exact real number in floating point form, thus the **rounded number** is,

$$\boxed{x \in \mathbb{R}, \quad \pm \left(\sum_{k=1}^{t} b^*_{-k} 2^{-k} \right) 2^{e^*}} \tag{2.2}$$

Now, two types of rounding are possible,

- **Chopping**: $x = \text{chop}\,(x)$, it implies that

  $$\begin{aligned} e^* &= e \\ b^*_{-k} &= b_{-k} \end{aligned}$$

  for all $k = 1, 2, \cdots, t$.

- **Symmetric rounding**: The rounding up or rounding down in decimal arithmetic based on the first discarded digit. Example, If the discarded digit ≥ 5, then we round up and otherwise.

  Note: In binary arithmetic, **if the first discarded binary digit is 1 then we round up and if it is 0, then we round down**.

More concretely, we can write the **chop** and **rd** operation as the following,

$$\begin{aligned} x &= \text{rd}\,(x) \\ \text{rd}\,(x) &= \text{chop}\left(x + \underbrace{\frac{1}{2} \cdot 2^{-t} \cdot 2^{e}}_{\text{adds } \frac{1}{2} \text{ to make it compatible}} \right) \end{aligned} \tag{2.3}$$

In the case of **chopping**, the **absolute error** can be calculated as,

$$\begin{aligned} |x - \text{chop}\,(x)| &= \left| \pm \sum_{k=t+1}^{\infty} b_{-k} 2^{-k} \right| 2^{e} \\ &\boxed{\leq \sum_{k=t+1}^{\infty} 2^{-k} \cdot 2^{e} = 2^{-t} \cdot 2^{e}} \end{aligned} \tag{2.4}$$

Similarly, we can calculate the **relative error** in the case of chopping as,

$$\begin{aligned}\left|\frac{x-\text{chop}\,(x)}{x}\right| &\leq \frac{2^{-t}2^{e}}{\left|\pm\sum_{k=1}^{\infty} b_{-k}2^{-k}\right|2^{e}} \\ &\leq \frac{2^{-t}2^{e}}{\underbrace{\frac{1}{2}\ .\ 2^{e}}_{\text{this is minimized}}} \\ &\boxed{\leq 2\ .\ 2^{-t}}\end{aligned} \tag{2.5}$$

$\star$ Note that the **upper bound on the relative error is not dependent on** e**.**

$\star\star$**Remark:** The machine dependent term $\boxed{\text{eps} = 2^{-t}}$ is also called the **machine precision** .

Similarly, for the **rounding**, we can find the relative error to be upper bounded by the following inequality,

$$\boxed{\left|\frac{x-\text{rd}\,(x)}{x}\right| \leq 2^{-t}} \tag{2.6}$$

for convenience, we write this in the following equivalent form,

$$\boxed{\text{rd}\,(x) = x\,(1+\varepsilon)\ ,\quad |\varepsilon| \leq \text{eps}} \tag{2.7}$$

2.1.2 Machine Arithmetic

Definition: *Let* $\circ$ *denote arithmetic operation* $(=,+,-,\times,/)$*. Let* $x, y \in \mathbb{R}\,(t,x)$ *and let* $fl\,(x \circ y)$ *denote the machine produced result of the arithmetic operation, then,*

$$fl\,(x \circ y) = x \circ y\,(1+\varepsilon)\ ,\quad |\varepsilon| \leq eps \tag{2.8}$$

Multiplication

Consider values $x\,(1+\varepsilon_x)$ and $y\left(1+\varepsilon_y\right)$ of x and y *contaminated* by relative errors ε_x and ε_y respectively, in this case, the **relative error in the**

product is,

$$\begin{aligned} &\text{rd}\,(x.y) \approx xy\,(1+\varepsilon_x+\varepsilon_y) \\ \implies &\left|\frac{xy-\text{rd}\,(xy)}{xy}\right| \approx \varepsilon_x+\varepsilon_y \\ &\boxed{\varepsilon_{x.y}=\varepsilon_x+\varepsilon_y} \end{aligned}$$

where we neglect higher order interaction terms of error in product.

Division

$$\begin{aligned} \frac{x\,(1+\varepsilon_x)}{y\,(1+\varepsilon_y)} &= \frac{x}{y}\,(1+\varepsilon_x)\left(1-\varepsilon_y+\varepsilon_y^2-\ldots\right) \\ &\approx \frac{x}{y}\,(1+\varepsilon_x-\varepsilon_y) \end{aligned}$$

therefore,

$$\boxed{\varepsilon_{\frac{x}{y}}=\varepsilon_x-\varepsilon_y}$$

⋆Addition and Subtraction

$$x\,(1+\varepsilon_x)+y\,(1+\varepsilon_y)=(x+y)\left(1+\frac{x\varepsilon_x+y\varepsilon_y}{x+y}\right)\ ,\ \ x+y\neq 0$$

Therefore, we have the interesting case that,

$$\boxed{\varepsilon_{x+y}=\frac{x}{x+y}\varepsilon_x+\frac{y}{x+y}\varepsilon_y} \tag{2.9}$$

Therefore, now the relative error also depends on the inputs! Thus the addition is a **benign** operation. The error in Eq. 2.9 is called **cancellation error**. This type of **cancellation error should be avoided in the calculations as much as possible, thus try to avoid addition and subtraction directly as much as possible.**

Example 2: *If $y=\sqrt{x+\delta}-\sqrt{x}$, $x>0$, $|\delta|$ is very small. To avoid cancellation error, one should write*

$$y=\frac{\delta}{\sqrt{x+\delta}+\sqrt{x}}$$

Thus finding equivalent representations is an important concept for direct Addition and Subtraction as concretely shown in above example (and in slides).

Chapter 3

The Solution of Non-Linear Equations by Bisection Method

We want to find the real root of the following type of equations,

$$f(x) = 0$$

where $f : [a, b] \to \mathbb{R}$. Here the function $f(x)$ might be any of the following,

- a polynomial in x
- a transcendental function
- any combination of the above two.

There is no closed form method to find solutions to such equations, like the following

$$e^{-x} - \sin x = 0$$

Thus to solving these, one **has to look for approximate methods**.

Note that before asking the question that *what are the solutions of* $f(x)$, we should first answer the question that *whether there exists solutions of* $f(x) = 0$ *or not.* In that extent, we have the following theorem,

Theorem 3.0 : *(Bolzano's Theorem) Assume that* $f \in C[a, b]$ *and* $f(a)f(b) < 0$. *Then there exists atleast one number* $r \in (a, b)$ *such that* $f(r) = 0$.

3.1 The Bisection Method

Given $f \in C[a, b]$ be such that $f(a_0)f(b_0) < 0$, then the bisection method can be explained by the following algorithm,

Algorithm

For $n = 0, 1, 2, ...$ do:

1. Set $c_n = \frac{a_n + b_n}{2}$.
2. If $|f(c_n)| < \epsilon$ (*prescribed tolerance*), then **accept** c_n, STOP.
3. If $f(a_n)f(c_n) < 0$, then **set** $a_{n+1} = a_n$ and $b_{n+1} = c_n$, **goto** Step 1.
4. else **set** $a_{n+1} = c_n$ and $b_{n+1} = b_n$, **goto** Step 1.

Remark : To avoid cancellation error, compute the mid-point c_n instead as the following,

$$c_n = a_n + \frac{b_n - a_n}{2}$$

3.2 Error Analysis

Let $[a_0, b_0], [a_1, b_1], \cdots,$ be successive intervals arising in the algorithm above with the following holding true,

$$a_0 \leq a_1 \leq \cdots \leq b_0 \text{ and } b_0 \geq b_1 \geq \cdots \geq a_0$$

We thus have,

$$b_{n+1} - a_{n+1} = \frac{1}{2}(b_n - a_n)$$

Note that the sequence $\{a_n\}$ is bounded above and hence **converges** (*increasing or decreasing sequence which is bounded*) . Similarly, $\{b_n\}$ also converges.

Since,

$$\boxed{b_n - a_n = \frac{1}{2}(b_{n-1} - a_{n-1}) = \cdots = \frac{1}{2^n}(b_0 - a_0)}$$

it thus follows that,

$$\lim_{n\to\infty} b_n - \lim_{n\to\infty} a_n = \lim_{n\to\infty} \frac{1}{2^n}(b_0 - a_0)$$

$$\implies \lim_{n\to\infty} a_n = \lim_{n\to\infty} b_n = r \ \ (say)$$

Hence, we now see that

$$\boxed{\lim_{n\to\infty} f(a_n)f(b_n) \leq 0 \implies f^2(r) \leq 0 \implies f(r) = 0}$$

But suppose we stop the process at the timestep n, thus we'll have the interval $[a_n, b_n]$. *Then the best estimate of the root will clearly be* $c_n = \frac{a_n+b_n}{2}$. The error at this timestep e_n will then be,

$$\boxed{e_n = |r - c_n| \leq \underbrace{\frac{1}{2}(b_n - a_n)}_{\text{len of half inter.}} = \frac{1}{2^{n+1}}(b_0 - a_0)}$$

Theorem 3.1 : *Assume that* $f \in C[a, b]$. *Let* $[a_0, b_0], [a_1, b_1], \cdots, [a_n, b_n], \cdots$ *denotes the intervals in the bisection method. Then* $\lim_{n\to\infty} a_n$ *and* $\lim_{n\to\infty} b_n$ ***exists, are equal and represent a zero of*** f. *If* $r = \lim_{n\to\infty} c_n$, *where* $c_n = \frac{a_n+b_n}{2}$, *then,*

$$|r - c_n| \leq 2^{-(n+1)}(b_0 - a_0)$$

CHAPTER 3. THE SOLUTION OF NON-LINEAR EQUATIONS BY BISECTION METHOD

Chapter 4

Newton-Raphson Method

We will first introduce the method for single variable function and then we will extend it to non-linear functions.

Let x_0 be an initial approximation to r and let h be the **correction** given to x_0 such that

$$f(x_0 + h) = 0$$

Assumption: We assume that $f'(x)$ exist and is continuous.

Now, by Taylor's theorem,

$$0 = f(x_0 + h) = f(x_0) + hf'(x_0) + \text{higher order terms ...}$$

Neglecting the higher order terms for sufficiently small h (*note that we are hence linearizing the original equation*), we simply get

$$h = -\frac{f(x_0)}{f'(x_0)}$$

Hence the *first approximation to the exact root* is given by

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

Now we repeat the above procedure by substituting our initial approximation to be x_1. Hence setting $x_0 \leftarrow x_1$ and repeat the above procedure to have,

$$h = -\frac{f(x_1)}{f'(x_1)}$$

and obtain the second approximation as

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

In general,after n^{th} step an approximaion to the exact root becomes,

$$\boxed{x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}\,, \quad n \geq 0}$$

which is the **Newton-Raphson Method**.

4.1 Geometric Interpretation

The n^{th} approximation of the root by the Newton-Raphson's method is given geometrically by drawing the tangent to $f(x)$ at the point x_n and determining the point of intersection of input axes (X-axis in the case of single variable functions) with this tangent.

More concretely, the equation of the tangent to f at (x_0, y_0) is given by,

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

Now the above tangent line meets the x-axis at $y = 0$, which gives us,

$$x_1 \approx x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

which is just the first approximation.

4.2 Error Analysis

We need some definitions before diving into analysis of Newton's method.

Definition: *(Order of Convergence) Let $\{x_n\}$ be a sequence of real numbers tending to a limit r and set $e_n = x_n - r$. **If** there exists a number p and a constant $C > 0$ such that*

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

*then p is called the **order of convergence of the sequence** $\{x_n\}$ and C is called the **asymptotic error constant**.*

Now, let r be a root of $f(x) = 0$. Let x_n be the approximation to r obtained by the Newton Raphson's method after the n^{th} iteration. Defining the error $e_n = r - x_n$, we thus get,

$$0 = f(r) = f(x_n + (r - x_n)) = f(x_n) + (r - x_n)f'(x_n) + \frac{(r - x_n)^2}{2}f''(\eta_n)$$

where η_n lies between r and x_n. Now solving for r from the above equations gives us

$$\begin{aligned} r &= x_n - \frac{f(x_n)}{f'(x_n)} - \frac{(r - x_n)^2}{2}.\frac{f''(x_n)}{f'(x_n)} \\ &= x_{n+1} - \frac{(r - x_n)^2}{2}.\frac{f''(x_n)}{f'(x_n)} \end{aligned}$$

hence the $r - x_{n+1} = e_{n+1}$ is given by,

$$e_{n+1} = -\frac{e_n^2}{2}.\frac{f''(x_n)}{f'(x_n)} \approx -\frac{1}{2}\frac{f''(r)}{f'(r)}e_n^2 = Ce_n^2$$

Hence, the **Newtons's method converges quadratically.**

NOTE: Newton's method yields quadratic convergence only under the following assumptions

1. r is a simple root of $(x) = 0$, i.e. r should not be a repeated root of f.
2. $f'(x)$ should not be too small in the neighborhood of he initial choice x_0, otherwise the steps taken would be too high for convergence.
3. The initial approximation x_0 should be chosen very close to the root, as there are examples, like piecewise defined functions in which $f'(x_0)$ could be zero, and hence again, the steps would be infinitely large.

But under certain conditions, we can concretely say that the Newton's method converges to a root, as stated in the following theorem.

Theorem 4.2 : *Let $f \in C^2([a, b])$ and the following conditions be satisfied:*

1. $f(a)f(b) < 0$.
2. $f'(x) \neq 0 \ \forall \ x \in [a, b]$.
3. $f''(x)$ *is either* ≥ 0 *or* ≤ 0 *for all* $x \in [a, b]$.

4. *At the endpoints a and b,*

$$\frac{|f(a)|}{|f'(a)|} < b - a \;\; and \;\; \frac{|f(b)|}{|f'(b)|} < b - a$$

Then the Newton's method converges to the unique root r of $f(x) = 0$ *in* $[a, b]$ *for* ***any*** *starting value* $x_0 \in [a, b]$.

REMARK : The regularity conditions implies the following,

- Bolzano's Theorem, there exists atleast one root of $f(x) = 0$ in $[a, b]$.
- Since the function f is now strictly increasing or decreasing, then there exists only one solution in the interval $[a, b]$.
- Since the function is now either concave from upward or downward, thus the derivative f' is either increasing or decreasing in $[a, b]$.
- To make sure the step size isn't more than the size of the interval itself.

Chapter 5

System of Non-Linear Equations

We extend the Newton's linearization technique now to system of non-linear equations. For that aim, we first consider system of two variables and then extend the same idea to any system of n variables.

Consider the system of two non-linear equations in two independent variables x_1 and x_2 as,

$$f_1(x_1, x_2) = f_2(x_1, x_2) = 0$$

Let (x_1^0, x_2^0) be the **initial approximation** to the exact solution and let the h_1^0 and h_2^0 be the **corrections** given to x_1^0 and x_2^0 respectively so that,

$$f_1(x_1^0 + h_1^0, x_2^0 + h_2^0) = f_2(x_1^0 + h_1^0, x_2^0 + h_2^0) = 0$$

Then **using Taylor's expansion upto linear terms** would yield

$$f_1(x_1^0 + h_1^0, x_2^0 + h_2^0) \approx f_1(x_1^0, x_2^0) + h_1^0 \frac{\partial f_1}{\partial x_1^0} + h_2^0 \frac{\partial f_1}{\partial x_2^0}$$
$$f_2(x_1^0 + h_1^0, x_2^0 + h_2^0) \approx f_2(x_1^0, x_2^0) + h_1^0 \frac{\partial f_1}{\partial x_1^0} + h_2^0 \frac{\partial f_1}{\partial x_2^0}$$

where the partial derivatives are evaluated at (x_1^0, x_2^0).

Rewriting the above equation in the matrix form,

$$\mathbf{Jh} = -\mathbf{f}$$

where **J** is the **Jacobian matrix J**(f_1, f_2), the **correction vector h** and the vector **f** are given by,

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1^0} & \frac{\partial f_1}{\partial x_2^0} \\ \frac{\partial f_2}{\partial x_1^0} & \frac{\partial f_2}{\partial x_2^0} \end{bmatrix}$$

$$\mathbf{h} = \begin{bmatrix} h_1^0 \\ h_2^0 \end{bmatrix}$$

$$\mathbf{f} = \begin{bmatrix} f_1(x_1^0, x_2^0) \\ f_2(x_1^0, x_2^0) \end{bmatrix}$$

If J is invertible, then the correction vector **h** is determined simply by,

$$\boxed{\mathbf{h} = -\mathbf{J}^{-1}\mathbf{f}}$$

and then the **the first approximation** would simply become,

$$\boxed{x_1^1 = x_1^0 + h_1^0 \quad , \quad x_2^1 = x_2^0 + h_2^0}$$

and then we can repeat the whole method to get even finer approximations. Hence, more generally, we get,

$$\begin{bmatrix} x_1^{n+1} \\ x_2^{n+1} \end{bmatrix} = \begin{bmatrix} x_1^n \\ x_2^n \end{bmatrix} + \begin{bmatrix} h_1^n \\ h_2^n \end{bmatrix}$$

Remark : Few things to keep in mind are as follows,

- At **each** stage or timestep, we need to solve the following linear system,
$$\mathbf{J} \begin{bmatrix} h_1^n \\ h_2^n \end{bmatrix} = - \begin{bmatrix} f_1(x_1^n, x_2^n) \\ f_2(x_1^n, x_2^n) \end{bmatrix}$$
- The **Stopping Criterion** is, given the tolerance tol,
$$|h_1^n| + |h_2^n| < tol$$

5.1 Newton's Iteration on Non-Linear system of n variables

Now, we extend the idea introduced earlier for system of 2 variables to a more general n variables.

Consider the system,

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}$$

where $\mathbf{f} = \big(f_1, f_2, \ldots, f_n\big)^T$ and $\mathbf{x} = \big(x_1, x_2, \ldots, x_n\big)^T$.

The **linearization of above equation** thus yields the following,

$$\mathbf{f}(\mathbf{x} + \mathbf{h}) \approx \mathbf{f}(\mathbf{x}) + \mathbf{f}'(\mathbf{x})\mathbf{h} \tag{5.1}$$

where the n long **correction vector** $\mathbf{h} = \big(h_1, h_2, \ldots, h_n\big)^T$ and the **Jacobian matrix of f** is,

$$\mathbf{f}'(\mathbf{x}) = \mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

Then, the correction vector **h** is simply obtained by,

$$\boxed{\mathbf{h} = -\mathbf{J}^{-1}\mathbf{f}(\mathbf{x})} \tag{5.2}$$

after this, we simply get the iteration formula as,

$$\begin{aligned} \mathbf{x}^{n+1} &= \mathbf{x}^n + \mathbf{h}^n \\ \mathbf{h}^n &= -\mathbf{J}^{-1}\mathbf{f}(\mathbf{x}^n) \end{aligned} \tag{5.3}$$

Hence, the whole process can be simply stated as the following,

Retrieve the initial point for the current timestep, then calculate the correction vector for this timestep by solving the 5.2, add this correction to the initial point to get the new approximation, substitute the new approximation as the initial approximation for the next timestep. Stop when the n^{th} power of l_n norm of correction vector is less than the tolerance tol hyperparameter.

Chapter 6

The Fixed-Point Method

In this chapter, we look at a new method for computing the roots of a non-linear function using the properties of it's fixed point.

6.1 Fixed Point of a function

Definition: *(Fixed Point) Let $f : \mathbb{R} \to \mathbb{R}$ be a real valued function. A point x_0 is called a fixed point of f if $f(x_0) = x_0$.*

Note that the equation $f(x) = 0$ can be put in the following form,

$$x = g(x)$$

with which, we can thus conclude that **any fixed-point of g above is a solution of $f(x) = 0$.**

For example, consider the function $f(x) = x^2 - x - 2$. Then, the possible choices for $g(x)$ are as follows,

- $g(x) = x^2 - 2$
- $g(x) = \sqrt{x+2}$
- $g(x) = 1 + \frac{2}{x}$
- $g(x) = x - \frac{x^2-x-2}{m}$, $m \neq 0$

We hence get the following definition,

Definition: *(Iteration Function) A function $g(x)$ is called an iteration function for another function $f(x)$, if*

$$g(x) = x$$

implies $f(x) = 0$.

Note that we have effectively reduced our problem of determining the solution of $f(x) = 0$ to the problem of determining a fixed point of the function $g(x)$. Hence we just need to find the fixed point of the particular iteration function to yield us the solution of $f(x) = 0$. For this particular task, we thus have the following algorithm,

Algorithm:-

1. Given an iteration function $g(x)$ and a starting point x_0, compute

$$x_{n+1} = g(x_n)\ , \quad n = 0, 1, 2, \ldots$$

   until $|x_n - r| < \epsilon$ (*eps* is the hyperparameter `tolerance`.).

Note that to implement the above algorithm, one needs to ensure the following,

- For given x_0, we **should** be able to generate $x_1, x_2, \ldots$.
- Sequence $\{x_n\} \to r$.
- r is a fixed point of $g(x)$, thus, $r = g(r)$.

A natural question which thus arises is that whether the above algorithm will guarantee to find the fixed point of g (thus, finding solution of $f(x) = 0$ in the process), for that purpose, we have the following theorem.

Theorem 6.3 : *(Existence of Fixed Point) Set $I = [a, b]$. Let $g(x)$ be an iteration function satisfying the following conditions,*

1. *$g : I \to I$.*
2. *$g : I \to I$ is continuous.*
3. *$\exists$ a constant $0 < K < 1$ such that $|g'(x)| \leq K$ for all $x \in I$*

Then, $g(x)$ has a unique fixed point $r \in I$ and the sequence of iterations $\{x_n\}$ generated by $x_{n+1} = g(x_n)$ converges to r.

Proof : Clearly, we first need to show there is a fixed point of g in I. Note that if $g(a) = a$ or $g(b) = b$ then it's trivial. However in the case when $g(a) \neq a$ and $g(b) \neq b$, first note that $g(a), g(b) \in I$, so we can say that $g(a) > a$ and $g(b) < b$.

With this, we then define the following,

$$h(x) = g(x) - x$$

Clearly, $h(x) \in [0, b-a]$, therefore $h(a) > 0$ and $h(b) < 0$ and since h is continuous, thus by Bolzano's theorem, there is a point $r \in I$ such that $h(r) = 0$ which simply means that $g(r) = r$. Hence, there exists a fixed point of g in I.

Now, to prove that the sequence of iterations generated by $x_{n+1} = g(x_n)$ converges to r, consider that at a particular instant of the iteration n, we are ϵ distance away from r, which is, consider,

$$r - x_n = \epsilon$$

If we now calculate ϵ,

$$\epsilon = r - x_n = g(r) - g(x_{n-1})$$

which can also be written via the first principle of derivative,

$$\epsilon = r - x_n = g'(\eta_n)(r - x_{n-1})$$

for some η_n between x_n and r.

Now, for iterations to converge,

$$\begin{aligned} |\epsilon_n| &\leq K|e_{n-1}| \\ &\leq K^2|e_{n-2}| \\ &\leq K^n|e_0| \end{aligned}$$

Therefore, in the limit $n \to \infty$, since $K < 1$, $e_n \to 0$. Which thus implies that $\{x_n\} \to r$.

Note that we also have to prove the *uniqueness* of the fixed point, for

that simply consider two points $r, r_1 \in I$ which are also fixed points, so, $g(r) = r$ and $g(r_1) = r_1$. If we set $x_0 = r$, then from the iteration,

$$x_1 = g(x_0) = g(r) = r$$

therefore, $e_0 = x_0 - r_1 = r - r_1$ and $e_1 = x_1 - r_1 = r - r_1$. But we know that $e_n \leq Ke_{n-1}$, therefore, we must have,

$$e_1 \leq Ke_0$$

Thus, we can conclude that $e_0 = e_1 = 0$, which simply implies that $r = r_1$.

6.2 Performance Analysis

We have the following theorem for the order of the convergence of the fixed point method.

Theorem 6.4 : *Let r be the root of $x = g(x)$. Assume that $g \in C^m(N_\delta(r))$ for $m \geq 2$ where $N_\delta(r)$ denotes a neighbourhood of r. Also assume that,*

$$g'(r) = \cdots = g^{(m-1)}(r) = 0$$

Then, if the initial guess $x_0 \in N_\delta(r)$, the iteration given by,

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

converges with order *m and*

$$\lim_{n\to\infty} \frac{(r - x_{n+1})}{(r - x_n)^m} = \underbrace{(-1)^{m-1} \cdot \frac{g^{(m)}(r)}{m!}}_{\textit{finite value !}}$$

Proof : Consider the iteration,

$$x_{n+1} = g(x_n)$$

We can expand it using Taylor's Theorem as follows,

$$x_{n+1} = g(x_n) = \sum_{k=0}^{m-1} \frac{(x_n - r)^k g^{(k)}(r)}{k!} + \frac{(x_n - r)^m g^{(m)}(\eta_n)}{m!}$$

Now, since $g'(r) = \cdots = g^{(m-1)}(r) = 0$, therefore,

$$x_{n+1} = r + \frac{(x_n - r)^m g^{(m)}(\eta_n)}{m!}$$

therefore,

$$\frac{x_n - r}{(x_n - r)^m} = \frac{g^{(m)}(\eta_n)}{m!}$$

finally, we get,

$$\frac{r - x_n}{(r - x_n)^m} = (-1)^{m-1} \frac{g^{(m)}(\eta_n)}{m!}$$

Chapter 7

Newton's Method from Fixed Point Method & Multiple Roots

We now analyze the Newton's method from Fixed Point method.

We know that the fixed point iteration algorithm for the iteration function of f converges with order m to a fixed point of the iteration function (refer to previous Theorem 6.4).

Now, recall the Newton's iteration,

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

If we want to use fixed point method here, then we should have the following recurrence relation,

$$x_{n+1} = g(x_n)$$

Comparing the above two equations, we thus get $g(x)$ as,

$$g(x) = x - \frac{f(x_n)}{f'(x_n)}$$

We can indeed verify that $g(x)$ has the fixed point as r (the root of $f(x) = 0$),

$$g(r) = r - \frac{f(r)}{f'(r)} = r$$

Thus, now we have the **iteration function for Newton's Method**. To, however prove the convergence of Newton's method and convergence of iteration, we invoke Theorem 6.4.

For this task, we would need to check derivative of the iteration function g, thus,

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

Again, for $x = r$, we get g' as,

$$g'(x) = 0$$

, thus we observe that,

$$g(r) = r \ , \ \ g'(r) = 0 \ , \ \ g''(r) = \frac{f''(r)}{f'(r)} \neq 0$$

Therefore **from Theorem 6.4, Newton's method has second order convergence**.

7.1 Newton's method on Multiple Roots

Definition : (Root with multiplicity) *If $r \in I$ is a root of $f(x) = 0$ with multiplicity $m > 1$, then*

$$f(x) = (x - r)^m h(x) \ , \ \ \textit{where } h(x) \neq 0$$

and $h(x)$ is continuous at $x = r$. Also, if $h \in C^m(I)$, then

$$f(r) = f'(r) = \cdots = f^{(m-1)} = 0 \textit{ and } f^m(r) \neq 0$$

Now, since we want to use Newton's method to find the root of $f(x) = 0$ where f has root with multiplicity > 1, thus we first need to find it's iteration function.

Note that since $f(x) = (x - r)^m h(x)$ and

$$f'(x) = (x - r)^m h'(x) + m(x - r)^{m-1} h(x)$$

thus, we get the iteration function (discussed previously for Newton's Iteration) $g(x)$ simply as,

$$g(x) = x - \frac{f(x)}{f'(x)} = x - \frac{(x - r)^m h(x)}{(x - r)^m h'(x) + m(x - r)^{m-1} h(x)} = \frac{(x - r)h(x)}{(x - r)h'(x) + mh(x)}$$

Again, you can check that $g(x)$ has the fixed point as r, $g(r) = r$.

Now, to find the order of convergence of Newton's Method for function f (which has root with higher multiplicity), we would again invoke the Theorem 6.4. Now to use this theorem, we need to find the amount of times we can differentiate g and still get a continuous function which is not zero at the point $x = r$, so to check it, we start differentiating g as follows,

$$g'(x) = 1 - \frac{h(x)}{(x-r)h'(x) + mh(x)} - (x-r)\frac{d}{dx}\left(\frac{h(x)}{(x-r)h'(x) + mh(x)}\right)$$

Now this implies the following,

$$g'(r) = 1 - \frac{1}{m} \neq 0$$

Hence **the convergence of Newton's method for functions with multiple roots is Linear (1^{st} Order).**

Now a natural question to ask is *whether we can make the convergence better?*

Now, **To regain the 2^{nd} order convergence**, notice in the above equation that $g'(x)$ can become 0 if we had m (the multiplicity of the root r) multiplied with 2^{nd} term of $g'(x)$. So, we can now **simply modify the Newton's Iteration formula** to the following,

$$\boxed{x_{n+1} = x_n - m\frac{f(x)}{f'(x)}} \tag{7.1}$$

which thus implies that we have the modified iteration function $g(x)$ as,

$$g(x) = x - m\frac{f(x)}{f'(x)}$$

Hence, we can easily verify now that,

$$g(r) = g'(r) = 0$$

Thus, using Theorem 6.4, we can certainly claim that the ***modified Newton's Iteration formula Eq. 7.1 for functions with multiple roots has second order convergence***.

Chapter 8

Interpolation

Approximation of functions with few values from it's range.

To see why we would need this, consider two points $P(x_0, f(x_0))$ and $Q(x_1, f(x_1))$. The linear curve joining PQ is,

$$y - f(x_0) = \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0)$$

If we set $y = p(x)$, then the above equation can be re-arranged to,

$$\begin{aligned} p(x) &= \frac{f(x_0)(x - x_1) - f(x_1)(x - x_0)}{x_0 - x_1} \\ &= f(x_0)L_0(x) + f(x_1)L_1(x) \end{aligned}$$

where $L_0(x) = \frac{x - x_1}{x_0 - x_1}$ and $L_1(x) = \frac{x - x_0}{x_0 - x_1}$.

Importantly observe that $p(x_0) = f(x_0)$ and $p(x_1) = f(x_1)$. Thus, we get the correct value of f at the two points at which we know the value of f. We in this chapter would extend this idea of fitting a linear curve between points of any function to fitting a polynomial curve between the points.

Definition: *(Interpolation) Given a set of* $(n+1)$ *data points* $(x_i, f(x_i))$, $i = 0, 1, \ldots, n$, *where* x_i*'s are distinct, find a function* $p(x)$ *such that*

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

The process of finding such a function $p(x)$ *is called Interpolation. If* $p(x)$ *is a polynomial then the process is called Polynomial Interpolation and* $p(x)$ *is called*

Interpolating Polynomial.

$\star$ Clearly, such a definition raises the following three questions which are important to answer,

1. Does such a $p(x)$ exists?
2. If $p(x)$ exists, then is it unique?
3. And if $p(x)$ exists, then how to construct it?

To answer the Question 1, we present the following theorem.

The Weierstrass Approximation Theorem for existence of $p(x)$

Theorem 8.5 : *(The WAT) Let $f \in C[a,b]$. Then for every $\epsilon > 0$, there is a **polynomial** $p(x)$ such that*

$$|p(x) - f(x)| < \epsilon \ \ \forall x \in [a,b]$$

This theorem thus guarantees the existence of a polynomial function p for approximating any continuous function f in a bounded interval. Hence the Question 1 is now answered in an affirmative manner.

To answer Question 2 & 3, we would need to look at particular methods for finding the polynomial $p(x)$, we thus look at the first method of such construction, the Lagrange's Interpolation.

8.1 Lagrange's Interpolating Polynomial

We here wish to find a polynomial $p(x)$

$$p(x) = a_0 + a_1 x + \cdots + a_n x^n$$

where $a_0, a_1, \ldots, a_n$ are to be determined **so that**,

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

The above condition thus forms a system of $n+1$ equations as follows

$$\begin{aligned} a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n &= f(x_0) \\ a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n &= f(x_1) \\ \vdots &= \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n &= f(x_n) \end{aligned}$$

In Matrix form, we can write it to be,

$$Xa = F$$

where,

$$X = \begin{bmatrix} x_0 & x_0^2 & \dots & x_0^n \\ x_1 & x_1^2 & \dots & x_1^n \\ \vdots & \vdots & \dots & \vdots \\ x_n & x_n^2 & \dots & x_n^n \end{bmatrix}$$
$$a = \begin{bmatrix} a_0 & a_1 & \dots & a_n \end{bmatrix}^T$$
$$F = \begin{bmatrix} f(x_0) & f(x_1) & \dots & f(x_n) \end{bmatrix}^T$$

Therefore, we can find the vector a quite easily. But, it can oly be the case if X is invertible (or non-singular), to that extent, we have the following definition.

Definition: *(Vandermonde Matrix) The matrix X is called Vandermonde Matrix and*

$$det(X) = \prod_{0 \le i < j \le n} (x_i - x_j) \neq 0$$

Hence it implies that $Xa = F$ has a unique solution, answering the Question 2 in the process!

BUT! The inverse of Vandermonde Matrix is EXTREMELY COSTLY to compute, we need to solve $n+1$ linear equations!. Thus we do not use this to construct Lagrangian Polynomial.

Now to answer **Question 3** in the context of Lagrange's Interpolation, we see now how to construct $p(x)$ in a much more efficient manner.

8.1.1 Construction of Lagrange's Interpolating Polynomial

Consider the special interpolating problem:

$$f(x_i) = 1\ ,\ \ f(x_j) = 0 \text{ for } j \neq i \text{ for some } i\ ,\ \ 0 \le i \le n$$

We wish to find a polynomial of degree $\le$ n with the given n zeros $x_j\ ,\ \ j \neq i$

As shown in the opening example of this chapter, we can define the following,

$$L_i(x) = c(x - x_0)\dots(x - x_{i-1})(x - x_{i+1})\dots(x - x_n)$$

where c is some constant.

Clearly, we want $L_i(x_i) = f(x_i) = 1$, therefore, we can find c as the following,

$$c = \frac{1}{(x_i - x_0)\dots(x_i - x_{i-1})(x_i - x_{i+1})\dots(x_i - x_n)}$$

We would hence get the following form of $L_i(x)$,

$$\boxed{L_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} \quad i = 0, 1, \dots, n} \tag{8.1}$$

Note that $L_i(x)$ is a product of n linear factors with $deg(L_i(x)) = n$. **The functions $L_i(x)$ are called Lagrange Basis Functions** with the following property satisfied for special inputs,

$$L_i(x_j) = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases}, \quad j = 0, \dots, n. \tag{8.2}$$

We can thus construct the Lagrange Interpolating Polynomial for any general problem now as,

$$\begin{aligned} p(x) &= \sum_{i=0}^{n} f(x_i) L_i(x) \\ &= f(x_0)L_0(x) + \dots + f(x_n)L_n(x) \end{aligned} \tag{8.3}$$

We can, once again, see that $p(x_j) = f(x_j)\ , \ \ j = 0, 1, \dots, n.$

We still however need to study whether this construction is unique or not (Question 2), again, to that extent, consider $q(x)$ is another polynomial of degree $\leq n$ such that,

$$q(x_i) = f(x_i)\ , \ \ i = 0, 1, \dots, n$$

Now, define $r(x) = p(x) - q(x)$. Since $deg(r(x)) \leq n$, therefore $r(x_i) = p(x_i) - q(x_i) = 0$ for all $i = 0, 1, \dots, n$. But since $r(x)$ is of degree $\leq n$ with $n + 1$ zeros, therefore, $r(x) = 0$ and thus $p(x) = q(x)$.

Finally the formula

$$\boxed{p_n(x) = \sum_{i=0}^{n} f(x_i) L_i(x)} \tag{8.4}$$

where $L_i(x)$ is as given in Eq. 8.1 is called the ***Lagrange's formula for Interpolating Polynomial.***

8.2 Error in Lagrange's Interpolation

Theorem 8.6 : *(Interpolation Error) Let $f \in C^{n+1}[a,b]$. If $p_n(x)$ is a polynomial of degree $\leq n$ with $p_n(x_i) = f(x_i)$, $i = 0, 1, \ldots, n$, where x_i's are distinct points in $[a,b]$, then for all $t \in [a,b]$, there exists $\zeta = \zeta(t) \in (a,b)$ such that,*

$$E(t) = f(t) - p_n(t) = f(t) - \sum_{j=0}^{n} f(x_j) L_j(t) = \boxed{\frac{f^{(n+1)}(\zeta)}{(n+1)!} \prod_{j=0}^{n} (t - x_j)}$$

Proof : If $t = x_i$, $i = 0, 1, \ldots, n$, then LHS = RHS trivially. However for $t \neq x_i$ for all i, define the following,

$$E(x) = f(x) - p_n(x)$$

where $p_n(x) = \sum_{j=0}^{n} f(x_j) L_j(x)$.

Now set,

$$G(x) = E(x) - \frac{\psi(x)}{\psi(t)} E(t) \quad \text{where } \psi(x) = \prod_{j=0}^{n} (x - x_j)$$

Clearly, $G(x) \in C^{n+1}[a,b]$.

Now, further notice that,

$$G(x_i) = E(x_i) - \frac{\psi(x_i)}{\psi(t)} E(t) = 0 \quad \text{where } i = 0, 1, \ldots, n$$

and

$$G(t) = E(t) - E(t) = 0$$

Therefore, G has $n+2$ zeros on $[a,b]$. By Mean Value Theorem, the G' has $n+1$ distinct zeros in $[a,b]$, G'' has n distinct zeros in $[a,b]$ and so on.

Hence the $G^{(n+1)}(x)$ has only one zero in $[a, b]$. Therefore this implies that there exists $\zeta \in [a, b]$ such that $G^{(n+1)}(\zeta) = 0$. Thus,

$$G^{(n+1)}(x) = E^{(n+1)}(x) - \frac{\psi^{(n+1)}(x)}{\psi(t)}E(t) = f^{(n+1)}(x) - \frac{(n+1)!}{\psi(t)}E(t)$$

Hence, putting $x = \zeta$ we get,

$$G^{(n+1)}(\zeta) = 0 = f^{(n+1)}(\zeta) - \frac{(n+1)!}{\psi(t)}E(t)$$

which implies,

$$E(t) = \frac{f^{(n+1)}(\zeta)}{(n+1)!}\psi(t) = \frac{f^{(n+1)}(\zeta)}{(n+1)!}\prod_{j=0}^{n}(t - x_j)$$

Remarks : Note that this error $E(t)$ is only of theoretic importance as we would need $f^{(n+1)}$ and we most of the time use interpolation when we only have the data points, not the generating function f itself.

8.2.1 Example use in Linear Interpolation Error

In Linear Interpolation, we have two points, say, $(x_0, f(x_0))$ and $(x_1, f(x_1))$. We construct the Interpolation Error for the Linear Interpolation through the use of Theorem 8.6.

$$f(t) - p_1(t) = \frac{f''(\zeta)}{2!}(t - x_0)(t - x_1)$$

Now, if we assume that f'' is upper bounded by M, i.e.,

$$\left|f''(x)\right| \leq M \ \ \forall x \in [x_0, x_1]$$

With this, we can now form the bound on the error $f(t) - p_1(t)$ as follows,

$$\begin{aligned} |f(t) - p_1(t)| &\leq \left|\frac{f''(\zeta)}{2!}(t - x_0)(t - x_1)\right| \\ &\leq |(t - x_0)(t - x_1)|\frac{M}{2} \end{aligned}$$

Now, to convert the bound in terms of x_0 and x_1 only, we should maximize it. In doing so, we would need to find the maximum value of $|(t - x_0)(t -$

$x_1)|$ over $t \in [x_0, x_1]$, which can easily be seen to be achieved when $t = \frac{x_0+x_1}{2}$, thus, we get the more general upper bound on the error as,

$$\boxed{|f(t) - p_1(t)| \leq (x_0 - x_1)^2 \frac{M}{8}} \tag{8.5}$$

Thus the Eq. 8.5 gives us the bound on the error that can be incurred in Linear Interpolation over $[x_0, x_1]$ when the $f'' \leq M$.

8.2.2 Example use in Rounding Error Analysis for Linear Interpolation

Due to Rounding-Off in computer arithmetic, we incur error in calaculation of function values, thus,

$$f(x_0) = f_0 + \epsilon_0 \quad f(x_1) = f_1 + \epsilon_1$$

where ϵ_0 and ϵ_1 are rounding error.

Now, define the error $\varepsilon(x)$ as,

$$\varepsilon(x) = f(x) - \frac{(x_1 - x)f_0 + (x - x_0)f_1}{(x_1 - x_0)}, \quad x_0 \leq x \leq x_1$$

Now let's substitute the approximation of $f_0 = f(x_0) - \epsilon_0, f_1 = f(x_1) - \epsilon_1$ instead to get the following,

$$\begin{aligned} \varepsilon(x) &= \underbrace{f(x) - \frac{(x_1 - x)f(x_0) + (x - x_0)f(x_1)}{(x_1 - x_0)}}_{\text{Interpolation Error}} + \underbrace{\frac{(x_1 - x)\epsilon_0 + (x - x_0)\epsilon_1}{(x_1 - x_0)}}_{\text{Rounding Error}} \\ &= E(x) + R(x) \end{aligned} \tag{8.6}$$

Note that for Linear Interpolation Error, we found the bound in Eq. 8.5 and Rounding Error are themselves bounded by $\max(|\epsilon_0|, |\epsilon_1|)$ as $x \in [x_0, x_1]$. Hence, **to find a bound on Rounding Error in Linear Interpolation**, we simply combine these two results,

$$\begin{aligned} |E(x)| &\leq (x_1 - x_0)^2 \frac{M}{8} \\ \max_{x_0 \leq x \leq x_1} |R(x)| &\leq \max(|\epsilon_0|, |\epsilon_1|) \end{aligned} \tag{8.7}$$

Now, if the difference between x_1 and x_0 is h, then,

$$\boxed{|\varepsilon(x)| \leq h^2 \frac{M}{8} + \max(|\epsilon_0|, |\epsilon_1|)} \tag{8.8}$$

8.2.3 Drawback in Lagrange Interpolation formula

The Eq. 8.4 shows the formula for Lagrangian Interpolation. Note that this formula would create a new Polynomial each time the value of n increases, that is, if we get more and more data, we cannot reuse the information contained in the previous polynomials.

In practie, one is **uncertain about how many interpolation points to use**. If one increases the Interpolation points then the functions $L_i(x)$ have to be recomputed and the previous computation of $L_i(x)$ is essentially wasted. In other words, in calculating $p_k(x)$, no obvious advanatage can be taken of the fact that one already has $p_{k-1}(x)$ available.

8.3 Newton's Interpolation

Let $(x_i, f(x_i))$, $i = 0, 1, \ldots, n$ be the set of $n+1$ data points, where x_i's are **distinct**. We know that **there exists** a unique polynomial $p_n(x)$ of degree $\leq n$ by Theorem 8.5 such that,

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

To see how **the Interpolating Polynomial in Newton's form** overcome the drawback of Lagrangian Interpolation, consider the following Interpolating Polynomial,

$$\begin{aligned} p_n(x) &= c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0)(x - x_1)\ldots(x - x_{n-1}) \\ &= \sum_{j=0}^{n} c_j q_j(x) \ , \text{ where } \boxed{q_j(x) = \prod_{k=0}^{j-1}(x - x_k)} \end{aligned} \tag{8.9}$$

Note that the Polynomials $q_j(x)$ forms a **basis** for the Newton's Interpolating Polynomial in Eq. 8.9,

$$\begin{aligned} q_0(x) &= 1 \\ q_1(x) &= (x - x_0) \\ \vdots &= \vdots \\ q_n(x) &= (x - x_0)\ldots(x - x_{n-1}) \end{aligned}$$

Now, from the **interpolating condition**, we should have the following,

$$p_n(x_i) = \sum_{j=0}^{n} c_j q_j(x_i) = f(x_i) \ , \quad \text{for all } i = 0, 1, \ldots, n \tag{8.10}$$

Representing this as a matrix formulation yields us,

$$\mathbf{A}\boldsymbol{c} = \boldsymbol{F} \tag{8.11}$$

where $\mathbf{A}_{ij} = q_j(x_i)$, $\boldsymbol{c} = \begin{bmatrix} c_0 & c_1 & \dots & c_n \end{bmatrix}^T$ and $\boldsymbol{F} = \begin{bmatrix} f(x_0) & f(x_1) & \dots & f(x_n) \end{bmatrix}^T$.

It can be seen very easily that **Matrix A is Lower Triangular** as,

$$q_j(x_i) = \begin{cases} \prod_{k=0}^{j-1}(x_i - x_k) & \text{if } i > j-1 \\ 0 & \text{if } i \leq j-1 \end{cases} \tag{8.12}$$

To see the advantage of Newton's Interpolation more closely, **consider the case when** $n = 2$, for that, the Interpolating Polynomial would be,

$$\begin{aligned} p_2(x) &= c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) \\ p_2(x_0) &= c_0 \\ p_2(x_1) &= c_0 + c_1(x_1 - x_0) \\ p_2(x_2) &= c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1) \end{aligned} \tag{8.13}$$

and for this case, the **equivalent matrix form** for interpolation condition is,

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & (x_1 - x_0) & 0 \\ 1 & (x_2 - x_0) & (x_2 - x_0)(x_2 - x_1) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \end{bmatrix} \tag{8.14}$$

$\boxed{\star}$ Now note that c_0 only depends of $f(x_0)$, x_1 depends only on $f(x_0)$ and $f(x_1)$ and so on. Thus, we conclude that **In general,** c_n **depends on** $f(x_0), \dots, f(x_n)$, which we write as the following,

$$\boxed{c_n = f[x_0, \dots, x_n]} \tag{8.15}$$

Hence we **define the symbol** $f[x_0, \dots, x_n]$ **as the coefficients of** q_n when $p(x) = \sum_{k=0}^{n} c_k q_k(x)$ interpolates f at $x_0, \dots, x_n$.

With the previous discussion, we arrive at the following form of **Newton's formula for the interpolating polynomial**,

$$\begin{aligned} p_n(x) = f[x_0] + f[x_0, x_1](x - x0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \dots \\ + f[x_0, \dots, x_n](x - x_0)(x - x_1)\dots(x - x_{n-1}) \end{aligned} \tag{8.16}$$

or in more compact form,

$$\boxed{p_n(x) = \sum_{k=0}^{n} f[x_0, \ldots, x_k] \underbrace{\prod_{j=0}^{k-1}(x - x_j)}_{q_k(x)}} \tag{8.17}$$

And now it should be clear how the Eq. 8.17 does not changes the p_k's when new point is added. Note that when indeed a new point $(n+2)$ is added, then all we need to calculate is $f[x_0, \ldots, x_{n+1}](x - x_0) \ldots (x - x_n)$ and **add** it to the $p_n(x)$, to yield us the required $p_{n+1}(x)$.

Now the next question that arises, is **how can we find $f[x_0, \ldots, x_k]$ which is the coefficient of $q_k(x)$ in Eq. 8.17?** Note that this coefficient is called *divided differences*, as explained next.

8.3.1 Explicit formula for computing Divided Differences

First note that,

$$p_n(x_0) = f(x_0) = c_0 = f[x_0]$$

Similarly,

$$p_n(x_1) = f(x_1) = c_0 + c_1(x_1 - x_0) = f(x_0) + f[x_0, x_1](x_1 - x_0)$$

which simply implies that,

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

Hopefully, now the motivation behind the name *divided differences* would be clear in reader's mind.

Theorem 8.7 : In general, we can represent higher order divided difference of a function f using the following formula,

$$f[x_0, x_1, \ldots, x_n] = \frac{f[x_1, x_2, \ldots, x_n] - f[x_0, x_1, \ldots, x_{n-1}]}{x_n - x_0} \tag{8.18}$$

Proof : Let $p_k(x)$ be the polynomial of degree $\leq k$ such that,

$$p_k(x_i) = f(x_i)\ , \quad i = 0, 1, \ldots, k$$

Also, let $q(x)$ be the polynomial of degree $\leq n-1$ such that such that,

$$q(x_i) = f(x_i)\ ,\ \ i = 1, 2, \ldots, n$$

With these two polynomials, we **construct polynomial** $p_n(x)$ as follows

$$p_n(x) = q(x) + \frac{x - x_0}{x_n - x_0}\left[q(x) - p_{n-1}(x)\right] \tag{8.19}$$

Note how the above polynomial manages to be of degree atmost n by the combination of $q(x)$ (degree $\leq n-1$) and $p_{n-1}(x)$ (degree $\leq n-1$), where a linear term is multiplied to the latter.

In Eq. 8.19, note that for $i = 1, 2, \ldots, n-1$, we have

$$\begin{aligned} p_n(x_i) &= q(x_i) + \frac{x_i - x_0}{x_n - x_0}\left[q(x_i) - p_{n-1}(x_i)\right] \\ &= f(x_i) \end{aligned} \tag{8.20}$$

Hence we are also complying to the interpolating condition with the polynomial $p_n(x)$ for $i = 1, \ldots, n-1$. But what about $i = 0$ and $i = n$?

$$\begin{aligned} p_n(x_0) &= q(x_0) - \left[q(x_0) - p_{n-1}(x_0)\right] = p_{n-1}(x_0) = f(x_0) \\ p_n(x_n) &= q(x_n) = f(x_n) \end{aligned} \tag{8.21}$$

So, we have p_n which obeys the interpolating condition. Thus LHS and RHS of Eq. 8.20 can be compared componentwise.

Now, in Eq. 8.19, consider the coefficients of x^n in LHS and both parts of RHS. Since both sides are equal, then so would be the coefficients,

$$\begin{aligned} &\text{Coefficient of } x^n \text{ in } p_n(x) = f[x_0, \ldots, x_n] \ \text{ (from Eq. 8.17)} \\ &\text{Coefficient of } x^n \text{ in } xq(x) = \frac{1}{x_n - x_0} f[x_1, \ldots, x_n] \ \text{ ($q(x)$ is only dependent on } x_1, \ldots, x_n) \\ &\text{Coefficient of } x^n \text{ in } xp_{n-1}(x) = \frac{1}{x_n - x_0} f[x_0, \ldots, x_{n-1}] \ \text{ ($p_n(x)$ is only dependent on } x_0, x_1, \ldots, x_n) \end{aligned} \tag{8.22}$$

we hence get

$$f[x_0, x_1, \ldots, x_n] = \frac{f[x_1, x_2, \ldots, x_n] - f[x_0, x_1, \ldots, x_{n-1}]}{x_n - x_0}$$

Small Example on Divided Differences

Using the Theorem 8.7, we can calculate the coefficients of Newton's Interpolating polynomial $p_n(x)$ as,

$$\begin{aligned} f[x_0] &= f(x_0) \\ f[x_0, x_1] &= \frac{f[x_1] - f[x_0]}{x_1 - x_0} = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \\ f[x_0, x_1, x_2] &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{\frac{f(x_2)-f(x_1)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{x_1-x_0}}{x_2 - x_0} \end{aligned} \tag{8.23}$$

8.4 Error in Newton's Interpolation

Theorem 8.8 : *Let $p_n(x)$ be the polynomial of degree $\leq n$ such that $p_n(x_i) = f(x_i)$ for all $i = 0, 1, \ldots, n$. If t is a point such that $t \neq x_0, x_1, \ldots, x_n$, then,*

$$f(t) - p(t) = f[x_0, x_1, \ldots, x_n, t] \prod_{j=0}^{n} (t - x_j) \tag{8.24}$$

Proof : Looking at Eq. 8.17, we see that $q_n(x)$ only depends on $x_0, \ldots, x_{n-1}$. Hence, let $q(x)$ be the polynomial of degree $\leq n + 1$

$$q(x_i) = f(x_i)\ , \quad i = 0, 1, \ldots, n \ \text{ and } \ q(t) = f(t)$$

which contains an extra interpolating point t. With this, we can obtain the $q(x)$ by simply adding a new term to get $p_{n+1}(x)$ term from $p(x)$, that is,

$$q(x) = p(x) + f[x_0, x_1, \ldots, x_n, t] \prod_{j=0}^{n} (x - x_j)$$

and since $q(t) = f(t)$,

$$f(t) = p(t) + f[x_0, x_1, \ldots, x_n, t] \prod_{j=0}^{n} (t - x_j)$$

8.5 Relation between Divided Differences and it's Derivatives

Theorem 8.8 gives us the error in Newton's Interpolation Method. However, note that **we cannot evaluate the term $f[x_0, x_1, \ldots, t]$ without knowing the value $f(t)$!**.

But as we will see in this section, the number $f[x_0, x_1, \ldots, x_n, t]$ is **closely related to the** $(n+1)^{th}$ **derivative of** $f(x)$.

Theorem 8.9 : *Let* $f \in C^k([a,b])$. *If* $x_0, x_1, \ldots, x_k$ *are* $k+1$ *distinct points in* $[a,b]$, *then* ***there exists*** $\xi \in (a,b)$ *such that,*

$$f[x_0, x_1, \ldots, x_k] = \frac{f^{(k)}(\xi)}{k!} \tag{8.25}$$

Proof : Clearly, the proof would involve the use of **Mean Value Theorem** as it *guarantees the existence of a point in the interval where* f *is continuously differentiable such that divided differnece on endpoints is equal to the tangent line at that point.*

With that motivation, for $k = 1$,

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

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

Now, consider the following polynomial,

$$e_k(x) = f(x) - p_k(x)$$

Note, that it has $k+1$ zeros $x_0, x_1, \ldots, x_k$ in $[a,b]$. We can thus use the **Rolle's Theorem** which says that *if a differntiable function in an open interval attains same value at two points in that interval, then there exists a point in the same open interval where the derivative is zero.*

However, using Rolle's Theorem, we can see that since e_k is a polynomial then $e_k'(x) = 0$ has atleast k zeros in (a,b), $e_k'' = 0$ has $k-1$ zeros in (a,b) and so on. Finally, $e_k^{(k)} = 0$ would only have one zero in (a,b). That is, there exists $\xi \in (a,b)$ such that

$$e_k^k(\xi) = 0 \implies f^{(k)}(\xi) = p_k^{(k)}(\xi)$$

Now, note from Eq. 8.17, that $p_k^{(k)}(x) = f[x_0, \ldots, x_k]k!$ We thus get the required result,

$$f[x_0, x_1, \ldots, x_k] = \frac{f^{(k)}(\xi)}{k!}$$

QED

With the Theorem 8.9 in hand which relates the k^{th} divided difference with the k^{th} derivative of f, we can reform the Theorem 8.8 to get a **new form of error in Newton's Interpolation** as follows,

Theorem 8.10 : *Let $f \in C^{n+1}([a,b])$. If $p_n(x)$ is a polynomial of degree $\leq n$ such that,*

$$p_n(x_i) = f(x_i)\ ,\ \ i = 0, 1, \ldots, n$$

Then, for all $t \in [a,b]$, there exists $\xi = \xi(t) \in (a,b)$ such that,

$$\boxed{f(t) - p_n(t) = \frac{f^{(n+1)(\xi)}}{(n+1)!} \prod_{j=0}^{n} (t - x_j)} \tag{8.26}$$

Proof : Just use Theorem 8.8 in view of Theorem 8.9, QED.

We now see some more interesting relations between divided differences and derivatives apart from Newton's Interpolation Error.

8.5.1 More Insights into Divided Difference

1. Note that

$$\lim_{x \to x_0} f[x_0, x] = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0} = f'(x_0)$$

   Which suggests that the **first divided difference at two identical points is the derivative of f at that point**, i.e.

$$f[x_0, x_0] = f'(x_0)$$

2. For any $n+1$ distinct points $x_0, x_1, \ldots, x_n$ in $[a,b]$ and any $f \in C^n[a,b]$, we have from Theorem 8.9 that

$$f[x_0, \ldots, x_n] = \frac{f^{(n)}(\xi)}{n!}$$

   Thus, if we let all $x_i \to x_0$ for all $i = 1, 2, \ldots, n$ with ξ being **constrained** to be in between them, then $\xi \to x_0$ too. We thus obtain,

$$\boxed{f[\underbrace{x_0, \ldots, x_0}_{n+1 \text{ times}}] = \frac{f^{(n)}(x_0)}{n!}} \tag{8.27}$$

8.6 Application of Newton's Interpolation : Inverse Interpolation

An interesting application of Newton's Interpolation is that of Inverse Interpolation, in which *the aim is to approximate the solution of the non-linear equation,*

$$f(x) = 0 \tag{8.28}$$

. Let $x_0 \approx \xi$ and $x_1 \approx \xi$ be two approximations of a root ξ of Eq. 8.28. Assuming that f is **monotone** near ξ, then,

$$y = f(x) \implies x = f^{-1}(y) = g(y) \tag{8.29}$$

Note that in view of Eq. 8.29, the root ξ is,

$$\xi = f^{-1}(0) = g(0)$$

Hence **our aim is to evaluate** $g(0)$.

Now, we compute

$$y_0 = f(x_0)$$
$$y_1 = f(x_1)$$

which implies that

$$x_0 = g(y_0)$$
$$x_1 = g(y_1)$$

We would now try to form an approximation of $g(y)$, the inverse of f, in order to find the value of $g(0)$. For this, the first divided difference is,

$$g[y_0, y_1] = \frac{g(y_0) - g(y_1)}{y_0 - y_1}$$

Using the above, **we can now improve the approximation of ξ by linear interpolation (refer to Eq. 8.17 with x = 0) as follows**,

$$\boxed{x_2 = x_0 + (0 - y_0)g[y_0, y_1] = x_0 - y_0 g[y_0, y_1]} \tag{8.30}$$

Note that **we can now compute** $y_2 = f(x_2)$. Hence, we can now compute next approximation which would be **quadratic** as follows,

$$\boxed{x_3 = x_2 + (0 - y_0)(0 - y_1)g[y_0, y_1, y_2] = x_2 + y_0 y_1 g[y_0, y_1, y_2]}$$

and then continue to find y_3 as,

$$y_3 = f(x_3) \implies x_3 = g(y_3)$$

Again, we can compute the **next term in Newton's Interpolation** as follows,

$$x_4 = x_3 - y_0 y_1 y_2 g[y_0, y_1, y_2, y_3] \ , \ \ y_4 = f(x_4) \implies x_4 = g(y_4)$$

and so on, we can continue in this way to generate new points for interpolation for inverse function g. Note that in all above iterations, we were keeping $y = 0$ as we need to find the value of interpolated polynomial at 0 only, we don't care about other values.

In general $x_k \to \xi$ with $k \to \infty$.

8.7 Hermite Interpolation

Definition: Given $n+1$ distinct points (nodes) $x_0, x_1, \ldots, x_n$ and the values $f(x_i) = y_i$ **and** $f'(x_i) = y_i'$ for $i = 0, 1, \ldots, n$. Our **Objective** is to *look for a polynomial $H(x)$ of degree at most $2n+1$ such that*,

$$\begin{gathered} H(x_i) = f(x_i) = y_i \\ \text{and} \\ H'(x_i) = f'(x_i) = y_i' \end{gathered} \tag{8.31}$$

for all $i = 0, 1, \ldots, n$. Therefore, we want to fit a polynomial of degree atmost $2n+1$ by using the above $2n+2$ conditions by also exploiting the conditions given on the derivatives of the function at the given points too.

In analogy with Lagrange's Formula Eq. 8.3, we write

$$H(x) = \sum_{i=0}^{n} y_i h_i(x) + \sum_{i=0}^{n} y_i' \tilde{h}_i(x)$$

where $h_i(x)$ and $\tilde{h}_i(x)$ are polynomials of degree $\leq 2n+1$ and **satisfies the following properties**,

$$\begin{aligned} h_i'(x_j) = \tilde{h}_i(x_j) = 0 \ , \ \ i, j = 0, 1, \ldots, n \\ h_i(x_j) = \tilde{h}_i'(x_j) = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases} \end{aligned} \tag{8.32}$$

Eq. 8.32 is needed for maintaining same functional and derivative values at all given points. We can verify it also that,

$$H(x_j) = y_j$$
$$H'(x_j) = y'_j$$

However, we still need to construct $h_i(x)$ and $\tilde{h}_i(x)$. For that, recall the **basis** for Lagrange Interpolating polynomial as given in Eq. 8.1 (repeated below for convenience),

$$L_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} \quad i = 0, 1, \ldots, n$$

Now simply define,

$$\boxed{h_i(x) = \left[1 - 2L'_i(x_i)(x - x_i)\right] \left(L_i(x)\right)^2 \ , \ \ 0 \leq i \leq n} \tag{8.33}$$

$$\boxed{\tilde{h}_i(x) = (x - x_i) \left(L_i(x)\right)^2} \tag{8.34}$$

It can be seen that $h_i(x)$ and $\tilde{h}(x)$ in Eq. 8.33 & 8.34 obey the constraints of Eq. 8.32. Hence, they are **valid** candidates for Hermite polynomials!

Therefore, the Hermite Polynomial in Lagrangian Form is,

$$\boxed{H(x) = \sum_{i=0}^{n} y_i h_i(x) + \sum_{i=0}^{n} y'_i \tilde{h}_i(x)} \tag{8.35}$$

8.7.1 Example with n=1

Let us take an example where we take $n = 1$ and construct the Hermite Interpolating polynomial in Lagrangian form with these two data points.

Therefore we have that,

$$f(x_0) = y_0$$
$$f(x_1) = y_1$$
$$f'(x_0) = y'_0$$
$$f'(x_1) = y'_1$$

Thus, we construct $h_0, h_1, \tilde{h}_0$ & $\tilde{h}_1$ as,

$$\begin{aligned} h_0(x) &= [1 - 2L_0'(x_0)(x - x_0)](L_0(x))^2 \\ h_1(x) &= [1 - 2L_1'(x_1)(x - x_1)](L_1(x))^2 \\ \tilde{h}_0(x) &= (x - x_0)(L_0(x))^2 \\ \tilde{h}_1(x) &= (x - x_1)(L_1(x))^2 \end{aligned}$$

where,

$$\begin{aligned} L_0(x) &= \frac{x - x_1}{x_0 - x_1} \\ L_1(x) &= \frac{x - x_0}{x_1 - x_0} \\ L_0'(x) &= \frac{1}{x_0 - x_1} \\ L_1'(x) &= \frac{1}{x_1 - x_0} \end{aligned}$$

8.7.2 Uniqueness of Hermite Interpolating Polynomial

To show the uniqueness, consider another Hermite interpolating polynomial $G(x)$ with degree $\leq 2n + 1$ with $G(x_i) = y_i$ and $G'(x_i) = y_i'$ for all $i = 0, 1, 2, \ldots, n$.

Now, simply **construct**,

$$R(x) = G(x) - H(x)$$

Clearly, $R(x_i) = R'(x_i) = 0 \; \forall \; i = 0, 1, \ldots, n$.

For points other than x_i, note that $R(x)$ has $n + 1$ double roots at $x = x_0, x_1, \ldots, x_n$. Therefore, we can write it as,

$$R(x) = q(x)(x - x_0)^2(x - x_1)^2 \ldots (x - x_n)^2$$

for $q(x) \neq 0$ is a polynomial.

Now, critically note that $(x - x_0)^2(x - x_1)^2 \ldots (x - x_n)^2$ is of degree $2n + 2$!! Which is a **contradiction** as both $G(x)$ and $H(x)$ are of degree atmost $2n + 1$.

Thus we are forced to conclude that $R(x) = 0$. QED.

8.8 Hermite Interpolation based on Divided Differences

Clearly, the divided differences form is much more accessible from computablility standpoint as compared to Lagrangian form as discussed earlier.

Consider that we are given functional values of $f(x)$ at points $z_1, z_2, \ldots, z_{2n}$. Writing this in Newton divided differences form yields us the following,

$$\begin{aligned} H(x) = f(z_1) &+ (x - z_1) f[z_1, z_2] \\ &+ (x - z_1)(x - z_2) f[z_1, z_2, z_3] \\ &+ \ldots \\ &+ (x - z_1)(x - z_2) \ldots (x - z_{2n-1}) f[z_1, z_2, \ldots, z_{2n}] \end{aligned} \tag{8.36}$$

The error in the above approximation can be written as the following with the help of Theorem 8.8 and a new interpolating point $x \neq z_i$ for all i,

$$f(x) - H(x) = (x - z_1)(x - z_2) \ldots (x - z_{2n}) f[z_1, \ldots, z_{2n}, x] \tag{8.37}$$

Now, let's relax the condition of disjoint points $z_1, \ldots, z_{2n}$, in particular, we set,

$$z_1, z_2 \to x_1 \ , \ \ z_3, z_4 \to x_2 \ , \ \ \ldots \ , \ \ z_{2n-1}, z_{2n} \to x_n$$

which enables us to write,

$$\begin{aligned} H(x) = f(x_1) &+ (x - x_1) f[x_1, x_1] \\ &+ (x - x_1)^2 f[x_1, x_1, x_2] \\ &+ \ldots \\ &+ (x - x_1)^2 \ldots (x - x_{n-1})^2 (x - x_n) f[x_1, x_1, \ldots, x_n, x_n] \end{aligned} \tag{8.38}$$

Now, from the above equation we see that the $H(x)$ is a polynomial of degree $\leq 2n - 1$.

Let's see how the error term $f(x) - H(x)$ has changed,

$$f(x) - H(x) = (x - x_1)^2 \ldots (x - x_{n-1})^2 (x - x_n)^2 f[x_1, x_1, \ldots, x_n, x_n, x]$$

To check whether the interpolation condition is satisfied or not,

$$\boxed{f(x_i) - H(x_i) = 0 \text{ for all } \ i = 1, 2, \ldots, n}$$

Hence we are good so far!

Now, for the derivative condition of Hermite Interpolation, we see that,

$$f'(x) - H'(x) = (x - x_1)^2 \dots (x - x_{n-1})^2 (x - x_n)^2 \frac{d}{dx} f[x_1, x_1, \dots, x_n, x_n, x] + 2f[x_1, x_1, \dots, x_n, x_n, x] \sum_{i=1}^{n} \left\{ (x - x_i) \prod_{j \neq i} (x - x_j)^2 \right\} \tag{8.39}$$

and thus it's easy to see that,

$$\boxed{f'(x_i) - H'(x_i) = 0 \text{ for all } \ i = 1, 2, \dots, n}$$

$\boxed{\star}$ So it satisfies all the criterion for the Definition of Hermite Polynomial in Eq. 8.31, and by the uniqueness of Hermite Interpolating Polynomial, we conclude that $H(x)$ **is unique**.

Hence, **the desired Hermite Interpolating Polynomial for functional and derivative data given for n points for a function f based on divided differences is given by**

$$\begin{aligned} H(x) = f(x_1) &+ (x - x_1) f[x_1, x_1] \\ &+ (x - x_1)^2 f[x_1, x_1, x_2] \\ &+ \dots \\ &+ (x - x_1)^2 \dots (x - x_{n-1})^2 (x - x_n) f[x_1, x_1, \dots, x_n, x_n] \end{aligned} \tag{8.40}$$

and the **associated Error Formula** is,

$$\boxed{f(x) - H(x) = \prod_{j=1}^{n} (x - x_j)^2 f[x_1, x_1, \dots, x_n, x_n, x]} \tag{8.41}$$

$\boxed{\star}$ **Note that for finding $f[x_1, x_1]$, we will use Eq. 8.27.**

Refer to the slides for a nice and simple example.

8.9 Spline Interpolation

The word *Spline* appears from an instrument with which craftsmen used to draw smooth curves.

Spline Interpolation can be explained as a piecewise interpolation technique.

Definition: A spline is a function defined on a sequence of intervals by a corresponding sequence of polynomials. At the points where the intervals touch, the function and some of it's derivatives are required to be continuous.

Definition: *(Linear Spline) Given a set of* $(n+1)$ *data points*

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$$

with $x_0 < x_1 \cdots < x_n$. *These points* x_i*'s are called* ***knots*** *and* y_i*'s are called* ***ordinates***. *These* ***may*** *be values of a function* f *at the points* x_i, *i.e.* $f(x_i) = y_i$. *A function* $S(x)$ *that satisfies the following conditions:*

1. $S(x)$ *is a* ***linear*** *function in the intervals* $[x_i, x_{i+1}]$.
2. $S(x)$ *is* ***continuous*** *in* $[x_0, x_n]$.
3. $S(x_i) = y_i$ *for* $i = 0, 1, \ldots, n$.

is called ***Linear Interpolating Spline***.

A very natural question to ask now is that *whether such a function even exists or not?*

To see that it exists, consider $S_i(x)$ as

$$S_i(x) = y_i + \frac{u_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i)$$

Now to check whether it follows the three conditions for a linear spline, first note that $S_i(x)$ is clearly linear for some i. Defining $S(x)$ as

$$S(x) = S_i(x) \text{ for } x \in [x_i, x_{i+1}]$$

then $S(x)$ is piece-wise linear function which is continuous in the whole interval while satisfying the interpolating condition, i.e. $S(x_i) = y_i$.

Therefore such functions exists!

However, more useful method of Spline Interpolation can be achieved by fitting higher order polynomials in between the points, which is what we consider now.

8.9.1 Cubic Spline Interpolation

Definition: *(Cubic Spline) A function $S(x)$ is a cubic spline if*

1. *$S(x)$ is a polynomial of degree ≤ 3 on each $[x_i, x_{i+1}]$.*
2. *$S', S'' \in C([x_0, x_n])$*
3. *$S(x_i) = y_i$ for all $i = 0, 1, \ldots, n$*
4. *First and Second derivative has to be continuous on each interval boundary.*

Now suppose we are given $n+1$ points $(x_0, y_0), \ldots, (x_n, y_n)$ as usual. Let S_i be the cubic polynomial on $[x_i, x_{i+1}]$ representing $S(x)$. Then (note the closed intervals!),

$$S(x) = \begin{cases} S_0(x) & x \in [x_0, x_1] \\ S_1(x) & x \in [x_1, x_2] \\ \vdots & \vdots \\ S_{n-1}(x) & x \in [x_{n-1}, x_n] \end{cases}$$

additionally with

$$S_{i-1}(x_i) = y_i = S_i(x_i) \text{ for } i = 1, \ldots, n-1$$

and combined with the **assumed continuity of S' and S''**, we also get the following conditions,

$$S'_{i-1}(x_i) = S'_i(x_i)$$
$$S''_{i-1}(x_i) = S''_i(x_i)$$

for $i = 1, \ldots, n-1$.

Can we find such a function?

One interesting question at this point can be to ask if we can even form such a function $S(x)$ with the conditions given **so-far**.

First, it's easy to see that we need $4n$ conditions to make $S(x)$ (the coefficients of third degree polynomial on each interval). However, a quick observation tells us that we have total $2n$ (Boundary values of each subinterval) $+(n-1)$ (Boundary values for continuity of S_i') $+(n-1)$ (Boundary values for continuity of S_i'').

So in total, we have $4n-2$ total conditions, **which is 2 short than the required** $4n$ **conditions!!** Thus we still need 2 more conditions to make this endeavour possible. We will see these two conditions as we will try to derive the cubic spline in the next section.

8.9.2 Derivation of Cubic Spline

Let us proceed to derive equation for $S_i(x)$ on $[x_i, x_{i+1}]$ for $0 \leq i \leq n-1$. For this, let us define,

$$m_i = S''(x_i) \text{ for all } 0 \leq i \leq n$$

Clearly, m_i also satisfies (**the continuity of** S''),

$$\lim_{x \to x_i^-} S''(x) = m_i = \lim_{x \to x_i^+} S''(x) \text{ for all } 1 \leq i \leq n-1$$

Also since S_i is an atmost cubic polynomial on $[x_i, x_{i+1}]$, it directly implies that S_i'' is atmost linear and satisfies,

$$S_i''(x_i) = m_i \text{ and } S_i''(x_{i+1}) = m_{i+1} \text{ for all } 1 \leq i \leq n-1$$

Thus, S_i''**, which is linear, is given by** (Lagrange's Interpolation Eq 8.1,8.4 on the interval $[x_i, x_{i+1}]$),

$$\begin{aligned} S_i''(x) &= m_i \frac{x_{i+1} - x}{x_{i+1} - x_i} + m_{i+1} \frac{x - x_i}{x_{i+1} - x_i} \\ &= \frac{m_i}{h_i}(x_{i+1} - x) + \frac{m_{i+1}}{h_i}(x - x_i) \end{aligned} \tag{8.42}$$

where $\boxed{h_i = x_{i+1} - x_i}$. Note the reversed order of subtraction in two terms.

Eq. 8.42 has given the S_i'' for each interval $[x_i, x_{i+1}]$. Since S_i'' is **linear**, to get $S(x)$, we can simply **integrate Eq. 8.42 twice**, the results of which are,

$$S_i(x) = \frac{m_i}{6h_i}(x_{i+1} - x)^3 + \frac{m_{i+1}}{6h_i}(x - x_i)^3 + C(x - x_i) + D(x_{i+1} - x) \tag{8.43}$$

where C & D are constants of integration.

The conditions $S_i(x_i) = y_i$ and $S_i(x_{i+1}) = y_{i+1}$ (**Interpolating Conditions**) can be used to solve for C and D, the final results are thus,

$$\boxed{\begin{aligned} S_i(x) = {} & \frac{m_i}{6h_i}(x_{i+1} - x)^3 + \frac{m_{i+1}}{6h_i}(x - x_i)^3 \\ & + \left(\frac{y_{i+1}}{h_i} - \frac{m_{i+1}h_i}{6}\right)(x - x_i) + \left(\frac{y_i}{h_i} - \frac{m_i h_i}{6}\right)(x_{i+1} - x) \end{aligned}} \tag{8.44}$$

Now, we will use the final condition, the **Continuity of** S'. First of all, the S_i' is,

$$S_i'(x) = -\frac{m_i}{2h_i}(x_{i+1} - x)^2 + \frac{m_{i+1}}{2h_i}(x - x_i)^2 + \left(\frac{y_{i+1}}{h_i} - \frac{m_{i+1}h_i}{6}\right) - \left(\frac{y_i}{h_i} - \frac{m_i h_i}{6}\right) \tag{8.45}$$

Now, since,

$$S_{i-1}'(x_i) = S_i'(x_i) \quad \text{for all} \quad i = 1, 2, \ldots, n-1$$

Therefore,

$$\begin{aligned} S_i'(x_i) &= -\frac{m_i h_i}{3} - \frac{h_i m_{i+1}}{6} - \frac{y_i}{h_i} + \frac{y_{i+1}}{h_i} \\ S_{i-1}'(x_i) &= \frac{h_{i-1}m_{i-1}}{6} + \frac{h_{i-1}m_i}{3} - \frac{y_{i-1}}{h_{i-1}} + \frac{y_i}{h_{i-1}} \end{aligned} \tag{8.46}$$

Equating them, we get,

$$\boxed{h_{i-1}m_{i-1} + 2(h_i + h_{i-1})m_i + h_i m_{i+1} = \frac{6}{h_i}(y_{i+1} - y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1})} \tag{8.47}$$

The Eq. 8.47 leads to $n-1$ linear equations with $n+1$ unknown in $m_0, m_1, \ldots, m_n$. If we select m_0 **and** m_n **arbitrarily**, we can solve it for $n-1$ variables $m_1, \ldots, m_{n-1}$.

Doing the above calculation for $m_0 = S_0''(x_0) = m_n = S_n''(x_n) = 0$ leads us to the definition of **Natural Cubic Spline**.

Natural Cubic Spline in Matrix Form

With $m_0 = m_n = 0$, the resulting linear system can be represented as the following matrix equation,

$$\begin{bmatrix} u_1 & h_1 & & & & \\ h_1 & u_2 & h_2 & & & \\ & h_2 & u_3 & h_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{n-3} & u_{n-2} & h_{n-2} \\ & & & & h_{n-2} & u_{n-1} \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ \vdots \\ m_{n-2} \\ m_{n-1} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix} \quad \text{with} \quad m_0 = m_n = 0 \tag{8.48}$$

where,

$$\begin{aligned} h_i &= x_{i+1} - x_i \\ u_i &= 2(h_i + h_{i-1}) \\ b_i &= \frac{6}{h_i}(y_{i+1} - y_i) \\ v_i &= b_i - b_{i-1} \end{aligned} \tag{8.49}$$

for $1 \leq i \leq n-1$. Hence, we obtain a **tridiagonal and diagonally dominant system**, which can be solved by Gaussian Elimination for $m_1, \ldots, m_{n-1}$.

$\boxed{\star}$ Now after finding $m_1, m_2, \ldots, m_n$, we can **simply put those values in Eq. 8.44 to get the Natural Cubic Spline Interpolating Function**.

8.10 Minimizing Property of Cubic Splines

We saw the constraints under which Cubic Splines are defined in as discussed in Section 8.9.1. Now with this first remember that the curvature of a curve $y = f(x)$ is given by,

$$\left|f''(x)\right| \left[1 + (f(x))^2\right]^{-\frac{3}{2}}$$

However, **if we drop the nonlinear $(f(x))^2$ term, we would still get a good enough approximation to the curvature as**

$$\left|f''(x)\right| \approx \text{curvature}$$

And as we saw from previous section, in **natural** cubic spline interpolation we are finding the **curve with minimal (approximate) curvature over an interval**, i.e.

$$\int_a^b \left[f''(x)\right]^2 dx$$

is being minimized, as visible by the process from Eq. 8.42 to 8.43.

Now, we formalize the above discussion in the following important theorem in Functional Analysis:

Theorem 8.11 : *(Minimal Curvature of Natural Cubic Spline) Let $f \in C^2([a,b])$ and let $a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$. If S is the natural cubic spline interpolating f at x_i, i.e. $S(x_i) = f(x_i)$ for $i = 0, 1, \ldots, n$, then*

$$\int_a^b \left[S''(x)\right]^2 dx \leq \int_a^b \left[f''(x)\right]^2 dx$$

Proof : Take $h(x) = f(x) - S(x)$. Clearly, $h(x_i) = 0$ for all $i = 0, 1, \ldots, n$. Now, note the following

$$\int_a^b \left[f''(x)\right]^2 dx = \underbrace{\int_a^b \left[S''(x)\right]^2 dx}_{\geq 0} + \underbrace{\int_a^b \left[h''(x)\right]^2 dx}_{\geq 0} + \underbrace{2\int_a^b S''(x)h''(x)dx}_{?}$$

The last term can be seen to be the deciding factor, in the sense that if it is greater than zero than we are done, however if it is less than zero, that can become problematic.

However, as we will show now, this **third term is indeed equal to zero**.

By using Integration by Parts, we get,

$$\begin{aligned}
\int_a^b S''(x)h''(x)dx &= \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} S''h''dx \\
&= \sum_{i=1}^{n} \left[\underbrace{(S''h')(x_i) - (S''h')(x_{i-1})}_{=0, \text{ Continuity constraints}} - \int_{x_{i-1}}^{x_i} S'''h'dx \right] \\
&= -\sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} \underbrace{S'''}_{\text{const.}} h'dx \\
&= -\sum_{i=1}^{n} c_i(h(x_i) - h(x_{i-1})) = 0
\end{aligned} \tag{8.50}$$

QED.

Chapter 9

Numerical Differentiation

We know that from past experience that differentiating a function is easier than integrating the function. However, to approximate the derivative is a much harder task then integrating the same function numerically!

Let us pose the question of numerical differentiate first in a concrete manner.

Definition: *(Numerical Differentiation & Integration Problem) Suppose we are giventhe values of a function $f(x)$ at a few points, say $x_0, x_1, \ldots, x_n$, can that information be used to estimate*

$$f'(c) \text{ (a } \textbf{\textit{derivative}}\text{)} \quad \text{or} \quad \int_a^b f(x)dx \text{ (an } \textbf{\textit{integral}}\text{)}$$

The answer to this problem is **yes**, we can using the interpolating polynomial which we just discussed!

Thus, in numerical differentiation and integration, we approximate the number

$$L(f(x)) \approx L(p(x))$$

where, L is a **linear operator** and $L(f)$ denotes one of these operations (derivatice of integral),

$$L(f) = f'(x) \quad \text{or} \quad L(f) = \int_a^b f(x)dx$$

and the $p(x)$ is the polynomial approximating to $f(x)$.

9.1 Differentiation via Polynomial Interpolation

We now discuss the process of differentiation through polynomial interpolation.

Let $f(x) \in C^{k+2}[c,d]$. If $x_0, x_1, \ldots, x_k$ are **distinct** points in $[c,d]$, we can write using the **Newton's Interpolating polynomial** the $f(x)$ as,

$$f(x) = p_k(x) + f[x_0, x_1, \ldots, x_k, x]\psi_k(x) \tag{9.1}$$

where $p_k(x)$ is a polynomial of degree $\leq k$ which satisfies the **interpolation conditions**

$$p_k(x_i) = f(x_i) \;\; i = 0, 1, \ldots, k$$

with ψ_k being,

$$\psi_k(x) = \prod_{j=0}^{k}(x - x_j)$$

(see Eq. 8.17).

Now, since we can write the derivative of the divided difference as the following (refer Sec. 8.5.1, 1^{st} point),

$$\frac{d}{dx}f[x_0, x_1, \ldots, x_k, x] = f[x_0, x_1, \ldots, x_k, x, x]$$

With the help of above, we can **differentiate** Eq. 9.1 to get the following,

$$f'(x) = p_k'(x) + f[x_0, x_1, \ldots, x_k, x, x]\psi_k(x) + f[x_0, x_1, \ldots, x_k, x]\psi_k'(x) \tag{9.2}$$

Now, let us define the derivative as the operator D to allow us to write $D(f) = f'(a)$ for any $a \in [c,d]$.

Now, **if** $D(f) \approx D(p_k)$, then we can write the error $E(f)$ as,

$$\begin{aligned} E(f) &= D(f) - D(p_k) \\ &= f[x_0, x_1, \ldots, x_k, a, a]\psi_k(a) + f[x_0, x_1, \ldots, x_k, a]\psi_k'(a) \\ &= \boxed{\frac{f^{(k+2)}(\xi)\psi_k(a)}{(k+2)!} + \frac{f^{(k+1)}(\eta)\psi'(a)}{(k+1)!}} \quad \text{(Theorem 8.9)} \end{aligned} \tag{9.3}$$

where $\xi, \eta \in [c,d]$.

9.1.1 Simplification of Error Eq. 9.3

To simplify the error expression 9.3, we analyze it under the following cases.

- **CASE I** : *When $a = x_i$ for some i.*

  In such a case, we see that $\psi_k(a)$ would be zero (Eq. 8.17) as it contains $(x - x_i)$ as a factor. Thus,

  $$\psi_k(a) = 0$$

  Coming to $\psi'(x)$, we can easily see that,

  $$\psi'_k(a) = (a - x_0)(a - x_1)\dots(a - x_{i-1})(a - x_{i+1})\dots(a - x_k)$$

  Hence, if we choose $a = x_i$, then we would get the error $E(f)$ simply as the following

  $$E(f) = \frac{f^{(k+1)}(\eta)}{(k+1)!} \prod_{j=0, j\neq i}^{k} (x_i - x_j) \quad \text{for } \eta \in [c, d] \tag{9.4}$$

- **CASE II** : *When a is such that $\psi'_k(a) = 0$.*

  We first discuss how this case can arise, i.e. for which a is $\psi'_k(a) = 0$.

  If it so happens that k is odd, then we can choose a to be the midpoint of the whole interval of interpolating points, i.e.

  $$x_{k-j} - a = a - x_j \ , \quad j = 0, \dots, \frac{k-1}{2}$$

  Then, we have that,

  $$\begin{aligned}(x - x_j)(x - x_{k-j}) &= (x - a + a - x_j)(x - a + a - x_{k-j}) \\ &= (x - a)^2 - (a - x_j)^2 \ , \quad j = 0, \dots, \frac{k-1}{2}\end{aligned}$$

  Hence, we get the $\psi_k(x)$ as,

  $$\psi_k(x) = \prod_{j=0}^{\frac{k-1}{2}} \left[(x - a)^2 - (a - x_j)^2\right]$$

Now since,

$$\frac{d}{dx}\left[(x-a)^2-(a-x_j)^2\right]\Big|_{x=a} = 2(x-a)|_{x=a} = 0 \;\; \forall\, j$$

Thus, it follows easily the required condition which we were trying to prove for a particular choice of a,

$$\psi_k'(a) = 0$$

And it's not difficult to see that the error term thus becomes,

$$E(f) = \frac{f^{(k+2)}(\xi)}{(k+2)!}\prod_{j=0}^{\frac{k-1}{2}}\left[-(a-x_j)^2\right] \quad \text{for some } \xi \in (c,d) \tag{9.5}$$

Also note that in this case the derivative is one orde higher than that in Eq. 9.4.

We now demonstrate few examples which portray the usage of above cases in determining the derivatives at the given points.

Exampe 1 : For $k = 1$,

$$p_1(x) = f(x_0) + f[x_0, x_1](x - x_0)$$

Thus,

$$D(p_1) = f[x_0, x_1]$$

regardless of a. If we set $a = x_0$, we thus arrive at **CASE I**.

Even more, if we set $h = x_1 - x_0$ then we obtain the following **forward difference formula**

$$f'(a) \approx f[a, a+h] = \frac{f(a+h) - f(a)}{h}$$

and then the error is simply given by Eq. 9.4 as,

$$E(f) = -\frac{1}{2}hf''(\eta)$$

However, if we choose a symmetrically around x_1 and x_0, i.e. $a = \frac{1}{2}(x_1 - x_0)$, we get the **CASE II**. With $x_0 = a - h$ and $x_1 = a + h$ and $h = \frac{1}{2}(x_1 - x_0)$, we obtain the **central-difference formula**,

$$f'(a) \approx f[a-h, a+h] = \frac{f(a+h) - f(a-h)}{2h}$$

and the associated error through the Eq. 9.5 is,

$$E(f) = -\frac{h^2}{6} f'''(\eta)$$

Example 2 : For $k = 2$ (three interpolation points), we would have,

$$p_k(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1)$$

whose derivative would be,

$$p'_k(x) = f[x_0, x_1] + f[x_0, x_1, x_2](2x - x_0 - x_1)$$

Again, at $x_0 = a$ (**CASE I**), we would get the derivative $f'(a)$ as (Eq. 9.4 and $D(f) = D(p_k) + E(f)$),

$$f'(a) = f[a, x_1] + f[a, x_1, x_2](a - x_1) + \underbrace{\frac{1}{6}(a - x_1)(a - x_2) f'''(\eta)}_{E(f)}$$

In a more representable way, we can write $x_1 = a + h$ and $x_2 = a + 2h$ to obtain (after opening all the finite differences),

$$f'(a) \approx \frac{-3f(a) + 4f(a + h) - f(a + 2h)}{2h}$$

and the error as (Eq. 9.4),

$$E(f) = \frac{h^2}{3} f'''(\xi) \quad \text{, for some} \quad \xi \in (a, a, +2h)$$

On the other hand, if we choose $x_1 = a - h$ and $x_2 = a + h$ (**This is NOT the CASE II**, we just assume a different distribution of points), then,

$$f'(a) \approx \frac{f(a + h) - f(a - h)}{2h}$$

and the error becomes,

$$E(f) = -\frac{h^2}{6} f'''(\xi) \quad \text{with} \quad |\xi - a| < |h|$$

9.1.2 Discretization

Observe in above examples that the formulas for the derivative of f were always of the form,

$$D(f) = D(p_k) + Ch^r f^{(r+1)}(\xi) \tag{9.6}$$

with $D(f) = f'(a)$ and h is the **spacing** of the points used for interpolation.

Definition: *(Discretization) The process of replacing $D(f)$ by $D(p_k)$ is known as **discretization** and the error term $Ch^r f^{(r+1)}(\xi)$ is called the **discretization error**.*

Derivative on a computer

Theorem 9.12 : *For a computer with fixed word length and for a given function f, there is an optimum value of h below which the approximation of the derivative with the derivative of p_k will become worse. The optimum value of h is that value for which the sum of the magnitudes of the round-off error and the discretization error is minimized.*

More concretely, consider the formula as discussed in example 2 above,

$$f'(a) = \frac{f(a+h) - f(a-h)}{2h} - \frac{h^2}{6} f'''(\xi)$$

In a computer, during calculation, **to model the roundoff**, we use,

$$\begin{array}{ll} f(a+h) + R_+ & \text{instead of } f(a+h) \\ f(a-h) + R_- & \text{instead of } f(a-h) \end{array} \tag{9.7}$$

Therefore,

$$\begin{aligned} f'_{comp} &= \frac{f(a+h) + R_+ - f(a-h) - R_-}{2h} \\ &= \frac{f(a+h) - f(a-h)}{2h} + \frac{R_+ - R_-}{2h} \end{aligned} \tag{9.8}$$

Hence,

$$\boxed{f'(a) = f'_{comp} - \underbrace{\frac{R_+ - R_-}{2h}}_{\text{Round-Off Error}} - \underbrace{\frac{h^2}{6} f'''(\xi)}_{\text{Discretization Error}}} \tag{9.9}$$

$\boxed{\star}$ Hence if we want to find optimum h, just simply find the value of h which minimizes the |Round-Off Error| + |Discretization Error|.

Now, if $f'''(x)$ is bounded then the discretization error goes to zero as $h \to 0$, but the round-off grows if we assume that $R_+ - R_-$ does not decrease.

9.2 Approximation of Derivatives via Taylor's Theorem

In this section we will focus on approximating the derivatives for a given function by the use of the Taylor's theorem.

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(\xi)$$

where $\xi \in (x, x+h)$ and $f'' \in C^2[(x, x+h)]$. We can rewrite the above equation as the following,

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2}f''(\xi)$$

Now note that the first term in the above equation is just the **forward-difference equation** (Example 1)! Thus if we **neglect the higher order term**, thus we get an approximation of f' as,

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \tag{9.10}$$

$\boxed{\star}$ The neglected higher order term $-\frac{h}{2}f''(\xi)$ is called the **Truncation Error (T.E.)**.

We can use the Taylor's theorem on $f(x-h)$ as the following,

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(\xi)$$

Thus, we get the following formula for approximating $f(x)$,

$$f'(x) \approx \frac{f(x) - f(x-h)}{h} \tag{9.11}$$

which is called **backward-difference formula**.

Similarly, we can **approximate to higher-order derivatives** through the Taylor's Theorem as follows,

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(\xi_1) \\ f(x-h) &= f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f'''(\xi_2) \end{aligned} \tag{9.12}$$

where $\xi_1 \in (x, x+h)$ and $\xi_2 \in (x-h, x)$. If we **subtract the above two equations**, we get,

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{12}[f'''(\xi_1) + f'''(\xi_2)] \tag{9.13}$$

Assuming $f''' \in C([x-h, x+h])$, there exists some point $\xi \in [x-h, x+h]$ such that,

$$f'''(\xi) = \frac{1}{2}[f'''(\xi_1) + f'''(\xi_2)]$$

With the above $f'''(\xi)$, we can write Eq. 9.13 as,

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6}f'''(\xi) \tag{9.14}$$

Thus, if we neglect the higher-order term, then,

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{9.15}$$

which is the **central-difference formula** as in Example 1.

Note that in this case, **the Truncation Error is** $O(h^2)$

It should be clear now that we can extend the above calculation to even more higher order derivatives. For example, if we extend the taylor's theorem to fourth order derivatives,

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \frac{h^4}{24}f^{(4)}(\xi_1) \\ f(x-h) &= f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f'''(x) + \frac{h^4}{24}f^{(4)}(\xi_2) \end{aligned} \tag{9.16}$$

Again, **adding the above two equations** leads us to,

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - \frac{h^2}{24}f^{(4)}(\xi) \tag{9.17}$$

for some $\xi \in (x-h, x+h)$. Then again, the **approximate formula of** $f''(x)$ is,

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \tag{9.18}$$

and the **Truncation Error is** $O(h^2)$.

Chapter 10

Numerical Integration

Seeing the difficulty of approximating the derivatives of a function, we now turn our attention to approximating the integral of a function. As we will see, the task of finding the integral is in general, easier than that of it's counterpart.

In particular, we wish to find the value of,

$$\int_a^b f(x)dx.$$

This can be alternatively stated as *the task of finding a function $F(x)$ such that $F'(x) = f(x)$*. Then, we would have,

$$\int_a^b f(x)dx = F(b) - F(a)$$

Note that there are a lots of functions whose anti-derivative are very difficult to compute, for example the following,

$$\int_0^2 e^{-x^2}dx\ ,\quad \int_0^1 \sin(xe^x)dx\ ,\quad \int_0^\pi \cos(3\cos\theta)d\theta$$

are **not amenable to the techniques learned in elementary calculus**.

A very basic question that may arise in the reader's mind is that ***if we are given values of a function $f(x)$ at a few points, say $x_0, x_1, \ldots, x_k$, then can that information be used to estimate***

$$\int_a^b f(x)dx \quad \textbf{\textit{an integral ?}}$$

Surprisingly, the answer is yes! It comes from the same idea that we used in numerical differentiation, of that of **approximating the function f by p_k**, i.e.

$$I(f) \approx I(p_k) = \int_a^b p_k(x)dx \tag{10.1}$$

where $p_k(x)$ is the polynomial of degree $\leq k$ with

$$p_k(x_i) = f(x_i)\ ,\ \ i = 0, \ldots, k \tag{10.2}$$

The error in the approximation is then given by,

$$E(f) = I(f) - I(p_k)$$

This **approximation is usually written as a weighted sum** as follows,

$$I(p_k) = A_0 f(x_0) + A_1 f(x_1) + \cdots + A_k f(x_k) \tag{10.3}$$

where A_i are the weights and $f(x_i)$'s are the functional values.

By Newton's Divided Differences, we can write the interpolation polynomial **using Theorem 8.8** as follows,

$$f(x) = p_k(x) + \underbrace{f[x_0, \ldots, x_k, x]\psi_k(x)}_{\text{Error Term}}$$

where we know that,

$$\begin{aligned} p_k(x) = f(x_0) &+ f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \ldots \\ &+ f[x_0, x_1, \ldots, x_k](x - x_0)(x - x_1)\ldots(x - x_{k-1}) \end{aligned}$$

$$\psi_k(x) = \prod_{j=0}^{k}(x - x_j)$$

10.1 Many rules of choosing the interpolating points

We now discuss the various ways one can choose the interpolating points and thus, the various ways we could thus find the integral.

10.1.1 Rectangle Rule

Let $k = 0$. Then,

$$f(x) \approx p_0(x) = f(x_0)$$

Hence, the approximate integral would be,

$$I(p_k) = \int_a^b f(x_0)dx = (b-a)f(x_0)$$

If $x_0 = a$, then this approximation becomes

$$I(f) \approx R = (b-a)f(a) \tag{10.4}$$

which is called the **Rectangle Rule**.

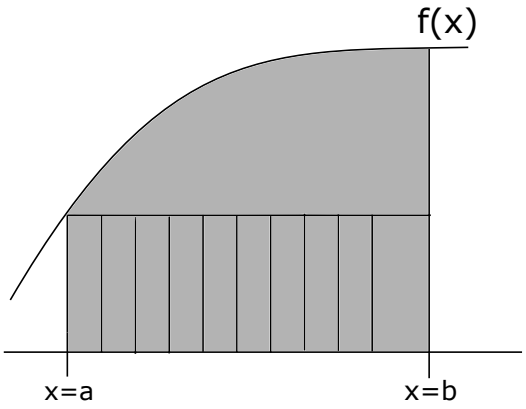


Figure 10.1: Rectangle Rule

10.1.2 Mid-point Rule

If $x_0 = \frac{a+b}{2}$, then the approximation would be,

$$I(f) \approx M = (b-a)f\left(\frac{a+b}{2}\right) \tag{10.5}$$

is known as the **mid-point rule**.

10.1.3 Trapezoidal Rule

Let $k = 1$. Then

$$f(x) \approx f(x_0) + f[x_0, x_1](x - x_0)$$

Choosing $x_0 = a$ **and** $x_1 = b$ to have,

$$I(p_1) = \int_a^b \left(f(a) + f[a,b](x-a)\right) dx$$

or,

$$I(f) \approx T = \frac{1}{2}(b-a)[f(a) + f(b)] \tag{10.6}$$

which yields the **trapezoidal rule**.

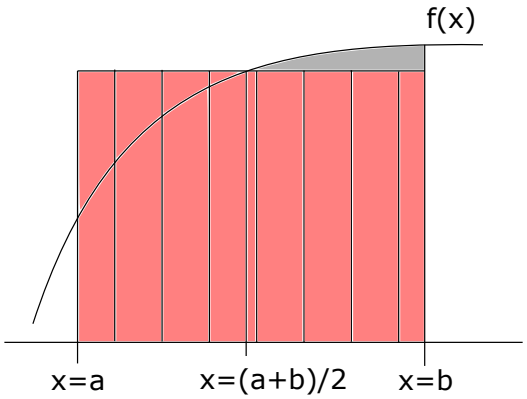


Figure 10.2: Mid-Point Rule

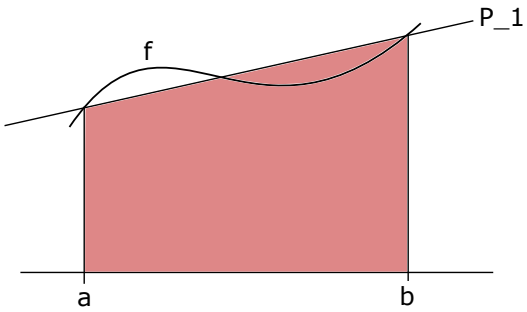


Figure 10.3: Trapezoidal Rule

10.1.4 Simpson's Rule

Let $k = 2$. Then

$$f(x) \approx p_2(x)$$

If we choose $x_0 = a$, $x_1 = \frac{a+b}{2}$ **and** $x_2 = b$, then the interpolating polynomial $p_2(x)$ is in the form,

$$p_2(x) = f(a) + f[a,b](x-a) + f\left[a, b, \frac{a+b}{2}\right](x-a)(x-b)$$

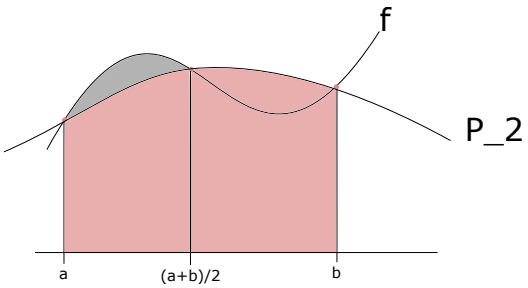


Figure 10.4: Simpson's Rule

Then, the approximate integral would be,

$$\begin{aligned} I(f) \approx I(p_2) &= f(a)(b-a) + f[a,b]\frac{(b-a)^2}{2} + f\left[a,b,\frac{a+b}{2}\right]\int_a^b (x-a)(x-b)dx \\ &= f(a)(b-a) + f[a,b]\frac{(b-a)^2}{2} + f\left[a,b,\frac{a+b}{2}\right]\frac{(b-a)^3}{6} \\ &= (b-a)\left\{f(a) + \frac{f(b)-f(a)}{2} - 2\frac{\left(f(b) - f\left(\frac{a+b}{2}\right) + f(a)\right)}{6}\right\} \\ &= \frac{b-a}{6}\left\{f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right\} \end{aligned} \tag{10.7}$$

Thus, the quadrature formula becomes,

$$I(f) \approx S = \frac{b-a}{6}\left\{f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right\} \tag{10.8}$$

which is called the **Simpson's Rule**.

10.1.5 Corrected Trapezoidal Rule

Suppose we have $k = 3$. Then, as usual,

$$f(x) \approx p_3(x)$$

Choosing $x_0 = x_1 = a$ **and** $x_2 = x_3 = b$, we notice that,

$$p_3(x) = f(a) + f[a,a](x-a) + f[a,a,b](x-a)^2 + f[a,a,b,b](x-a)^2(x-b)$$

Now, integrating $p_3(x)$ leads to the following approximation,

$$I(f) \approx I(p_3) = \int_a^b p_3(x)dx = \frac{b-a}{2}[f(a)+f(b)] + \frac{(b-a)^2}{12}[f'(a)-f'(b)] \tag{10.9}$$

which is known as **corrected trapezoidal rule**.

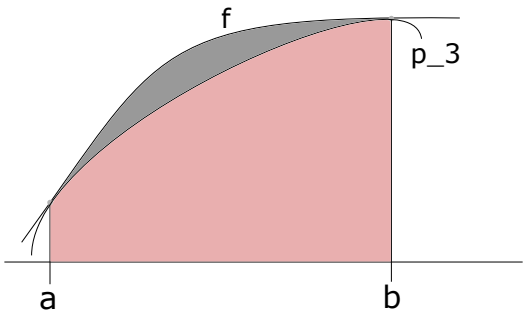


Figure 10.5: Corrected Trapezoidal Rule

10.2 Error Analysis

So far, we've seen the various methods to approximate the integral of a function f by approximating it via Newton's Interpolating method. A very obvious question that arises in mind after seeing all the methods is **the amount of error we are incurring in approximating the integral via Newton's interpolating polynomial**. Note that if we are able to quantify the error, then it would be much easier to judge which rule is most suitable for a given function f. Hence we now do exactly that, i.e. quantify the error.

By Newton's divided difference, we can write,

$$f(x) = p_k(x) + f[x_0, \ldots, x_k, x]\psi_k(x)$$

where $\psi_k(x) = \prod_{j=0}^{k}(x - x_j)$. The error $E(f)$ is given by,

$$E(f) = I(f) - I(p_k) = \int_a^b f[x_0, \ldots, x_k, x]\psi_k(x)dx$$

$\boxed{\star}$ **If $\psi_k(x)$ is only of one sign in interval $[a, b]$**, then by Mean Value Theorem

$$\int_a^b f[x_0, \ldots, x_k, x]\psi_k(x)dx = f[x_0, \ldots, x_k, \xi]\int_a^b \psi_k(x)dx$$

for some $\xi \in (a,b)$. If $f(x) \in C^{k+1}(a,b)$, then it follows that the error by Theorem 8.9 is,

$$\boxed{E(f) = \frac{1}{(k+1)!} f^{(k+1)}(\eta) \int_a^b \psi_k(x)dx \;\text{ for some }\; \eta \in (a,b)} \tag{10.10}$$

$\boxed{\star}$ **Even if $\psi_k(x)$ is not of one sign, certain simplifications in the error $E(f)$ are possible.** One particular desirable instance of this kind occurs when,

$$\int_a^b \psi_k(x)dx = 0 \tag{10.11}$$

For this case, we can use the identity,

$$f[x_0,\ldots,x_k,x] = f[x_0,\ldots,x_k,x_{k+1}] + f[x_0,\ldots,x_{k+1},x](x - x_{k+1}) \tag{10.12}$$

which is true for arbitrary x_{k+1}. Then the error $E(f)$ becomes

$$\begin{aligned} E(f) &= \int_a^b f[x_0,\ldots,x_{k+1}]\psi_k(x)dx + \int_a^b f[x_0,\ldots,x_{k+1},x](x - x_{k+1})dx \\ &= \int_a^b f[x_0,\ldots,x_{k+1},x]\psi_{k+1}(x)dx \end{aligned} \tag{10.13}$$

where $\psi_{k+1}(x)dx = (x - x_{k+1})\psi_k(x)$.

Now **choose** x_{k+1} in such a way that $\psi_{k+1}(x) = (x - x_{k+1})\psi_k(x)$ **is of one sign on** (a,b). Thus, if $f(x) \in C^{k+2}(a,b)$ then,

$$\boxed{E(f) = \frac{1}{(k+2)!} f^{(k+2)}(\eta) \int_a^b \psi_{k+1}(x)dx \;\text{ for some }\; \eta \in (a,b)} \tag{10.14}$$

10.2.1 Error in Rectangle Rule

In this case, we only have one point $x_0 = a$, hence,

$$\psi_0(x) = (x - a)$$

Thus, the error becomes,

$$E_R = E(f) = \int_a^b f[x_0,x](x - a)$$

$\boxed{\star}$ Now since $\psi_0(x)$ *is of one sign in* (a,b), hence we would use Eq. 10.10 to get,

$$E_R = f'(\eta) \int_a^b (x - a)dx = \boxed{\frac{f'(\eta)(b-a)^2}{2}} \tag{10.15}$$

10.2.2 Error in Mid-Point Rule

In this case, we still have only one point, however, that point is the mid-point of the interval (a, b), thus $x_0 = \frac{a+b}{2}$.

Note that in this case,

$$\psi_0(x) = (x - \frac{a+b}{2})$$

which is of NOT the same sign for $x \in (a, b)$. But note that,

$$\int_a^b \psi_0(x)dx = \int_a^b \left(x - \frac{a+b}{2}\right) dx = 0$$

Thus we can now simply use the second method discussed above.

For that, if we introduce another point $x_1 = \frac{a+b}{2}$, then

$$\psi_1(x) = (x - x_1)\psi_0(x) = (x - \frac{a+b}{2})^2$$

is of one sign. Thus, by Eq. 10.14, we get,

$$\boxed{E_M = \frac{f''(\eta)(b-a)^3}{24} \text{ for some } \eta \in (a, b)} \tag{10.16}$$

10.2.3 Error in Trapezoidal Rule

In this case, we have two points $x_0 = a$ and $x_1 = b$. Therefore,

$$\psi_1(x) = (x - a)(x - b)$$

As $\psi_1(x)$ is of one sign on (a, b), thus, we can simply use the Eq. 10.10 so that the error in Trapezoidal Rule E_T becomes,

$$\boxed{E_T = \frac{f''(\eta)}{2} \int_a^b (x-a)(x-b)dx = \frac{f''(\eta)(b-a)^3}{12} \text{ for some } \eta \in (a, b)} \tag{10.17}$$

10.2.4 Error in Simpson's Rule

In this case, we have 3 points as $x_0 = a$, $x_1 = \frac{a+b}{2}$ and $x_2 = b$. Thus, the $\psi_2(x)$ is,

$$\psi_2(x) = (x - a)(x - b)\left(x - \frac{a+b}{2}\right)$$

Now note that in (a, b), the sign of $\psi_2(x)$ is NOT same. But still, note that,

$$\int_a^b \psi_2(x)dx = \int_a^b (x-a)(x-b)\left(x - \frac{a+b}{2}\right)dx = 0$$

Hence we can **use the second method** in this case! For that all, we need to find is another point x_3 such that $\psi_3(x)$ **should be of same sign in** (a, b).

For this, first see that

$$\psi_4(x) = (x-a)(x-b)(x-x_3)\left(x - \frac{a+b}{2}\right)$$

and note that it can be of same sign on (a, b) only when $x_3 = \frac{a+b}{2}$. Thus, the error for Simpson's Rule can now can be written with the help of Eq. 10.14 as,

$$\begin{aligned} E_S &= \frac{f^{(4)}(\eta)}{24}\int_a^b (x-a)(x-b)\left(x - \frac{a+b}{2}\right)^2 dx \\ &= \frac{f^{(4)}(\eta)}{24} \times \frac{-4}{15}\left(\frac{b-a}{2}\right)^5 \\ &\boxed{= -\frac{f^{(4)}(\eta)}{90}\left(\frac{b-a}{2}\right)^5 \quad , \ \eta \in (a,b)}. \end{aligned} \tag{10.18}$$

10.2.5 Error in Corrected Trapezoidal Rule

We have three points here, $x_0 = x_1 = a$ and $x_2 = x_3 = b$. Now note that,

$$\psi_3(x) = (x-a)^2(x-b)^2$$

is of one sign on (a,b) and hence we can simply use Eq. 10.10 to get the error E_{CT} as,

$$\begin{aligned} E_{CT} &= \frac{f^{(4)}(\eta)}{4!}\int_a^b (x-a)^2(x-b)^2 dx \\ &= \boxed{\frac{f^{(4)}(\eta)(b-a)^5}{720} \ \eta \in (a,b)} \end{aligned} \tag{10.19}$$

10.3 Composite Methods

Let's take a look at Figure 10.1. The boxed one represents the area actual approximation and the whole grey region is the actual value of $\int_a^b f(x)dx$.

We can clearly see intuitively that this would contain heavy amount of error. This was also confirmed by error analysis and Eq. 10.15. It grows quadratically with the size of interval (a, b).

One way to overcome this might be to use different length rectangles in each subinterval after dividing the original interval. This would bring us somewhat closer to the Darboux's definition of Riemann integral. Thus, our aim now is to find a scheme to fit the length of rectangle which we should make dependent on the subinterval of (a, b). This is called composite rule where we use the previously discussed rules in the context of each subinterval of (a, b) instead of using it on whole of (a, b) at once.

10.3.1 Composite Rectangle Rule

Divide $[a, b]$ into N subintervals to get,

$$a = x_0 < x_1 < x_2 < \cdots < x_N = b$$

where $x_i = a + ih$, $i = 0, \ldots, N$ with **uniform step-size** $h = \frac{b-a}{N}$.

Set,

$$f_i = f(x_i) = f(a + ih) \ , \ \ i = 0, \ldots, N$$

and,

$$f_{i-\frac{1}{2}} = f(x_i - \frac{h}{2}).$$

Now, simply apply the **Rectangular Rule on each subinterval** to get,

$$\int_{x_{i-1}}^{x_i} f(x)dx = (x_i - x_{i-1})f(x_{i-1}) + \underbrace{\frac{f'(\eta_i)(x_i - x_{i-1})^2}{2}}_{\text{Error Term (Eq. 10.15)}}$$

$$= hf(x_{i-1}) + \frac{f'(\eta_i)h^2}{2}$$

Summing from $i = 1$ to $i = N$, we obtain,

$$\boxed{I(f) = \int_a^b f(x)dx = h\sum_{i=1}^{N} f_{i-1} + \sum_{i=1}^{N} \frac{f'(\eta_i)h^2}{2} = R_N + E_N^R} \qquad (10.20)$$

where $\eta_i \in (x_{i-1}, x_i)$ and R_N is called the **Composite Rectangle Rule**.

Note that if $f'(x)$ is continuous, then we can write,

$$\sum_{i=1}^{N} \frac{f'(\eta_i)h^2}{N} = f'(\eta) \sum_{i=1}^{N} \frac{h^2}{2} = \frac{f'(\eta)Nh^2}{2}$$

with $Nh = b - a$, it becomes,

$$E_N^R = \frac{f'(\eta)(b-a)h}{2} \quad \text{for some} \quad \eta \in (a,b) \tag{10.21}$$

10.3.2 Composite Mid-Point Rule

Continuing as in the previous subsection for the rectangle rule, we can do the same now for the Mid-Point rule,

$$\begin{aligned} I(f) \int_a^b f(x)dx &= \sum_{i=1}^{N} \int_{x_{i-1}}^{x_i} f(x)dx \\ &\approx h \sum_{i=1}^{N} f(x_{i-\frac{1}{2}}) \\ &= \boxed{h \sum_{i=1}^{N} f_{i-\frac{1}{2}}} \end{aligned} \tag{10.22}$$

where the error can be similarly found to be,

$$E_N^M = \frac{f''(\xi)(b-a)h^2}{24} \tag{10.23}$$

10.3.3 Composite Trapezoid Rule

Using Eq. 10.6,

$$\begin{aligned} I(f) = \int_a^b f(x)dx &= \sum_{i=1}^{N} \int_{x_{i-1}}^{x_i} f(x)dx \\ &\approx \frac{h}{2} \sum_{i=1}^{N} [f(x_{i-1}) + f(x_i)] \\ T_N &= \boxed{\frac{h}{2} \left[f_0 + 2 \sum_{i=1}^{N-1} f_i + f_N \right]} \end{aligned} \tag{10.24}$$

and the error E_N^T follows from Eq. 10.17,

$$E_N^T = -\frac{f''(\eta)(b-a)h^2}{12} \tag{10.25}$$

10.3.4 Composite Simpson's Rule

As usual, letting $a = x_{i-1}$, $b = x_i$ and $x_i - x_{i-1} = h$, apply Simpson's Rule over a single subinterval (x_{i-1}, x_i) in conjunction with result in Eq. 10.7 to obtain,

$$\int_{x_{i-1}}^{x_i} f(x)dx = \frac{h}{6}\left[f_{i-1} + 4f_{i-\frac{1}{2}} + f_i\right] - \frac{f^{(4)}(\eta_i)\left(\frac{h}{2}\right)^5}{90}, \quad x_{i-1} < \eta_i < x_i$$

Now simply summing over all subintervals $i = 1, \ldots, N$ to obtain,

$$\begin{aligned} I(f) &= \sum_{i=1}^{N}\int_{x_{i-1}}^{x_i} f(x)dx = \frac{h}{6}\sum_{i=1}^{N}\left[f_{i-1} + 4f_{i-\frac{1}{2}} + f_i\right] - \sum_{i=1}^{N}\frac{f^{(4)}(\eta_i)\left(\frac{h}{2}\right)^5}{90} \\ &= S_N + E_N^S \end{aligned} \tag{10.26}$$

The composite Simpson approximation S_N can be simplified to have,

$$\boxed{S_N = \frac{h}{6}\left[f_0 + f_N + 2\sum_{i=1}^{N-1} f_i + 4\sum_{i=1}^{N} f_{i-\frac{1}{2}}\right]} \tag{10.27}$$

with the error term E_N^S being,

$$E_N^S = \frac{f^{(4)}(\xi)(b-a)\left(\frac{h}{2}\right)^4}{180} \quad a < \xi < b \tag{10.28}$$

10.3.5 Composite Corrected Trapezoidal Rule

Following from Eq. 10.9,

$$\begin{aligned} I(f) &= \sum_{i=1}^{N}\int_{x_{i-1}}^{x_i} f(x)dx = \int_{x_0}^{x_1} f(x)dx + \cdots + \int_{x_{N-1}}^{x_N} f(x)dx \\ &\approx \frac{h}{2}[f(x_0) + f(x_1)] + \frac{h^2}{12}[f'(x_0) - f'(x_1)] \\ &+ \cdots + \frac{h}{2}[f(x_{N-1}) + f(x_N)] + \frac{h^2}{12}[f'(x_{N-1}) - f'(x_N)] \end{aligned} \tag{10.29}$$

Now note that all the interior derivatives form a *telescopic series*, thus cancelling each other except the first and the last terms. Thus, the **Composite**

Corrected Trapezoidal Rule is,

$$\boxed{I(f) \approx CT_N = \frac{h}{2}\left(f_0 + f_N + 2\sum_{i=1}^{N-1} f_i\right) + \frac{h^2}{12}[f'(a) - f'(b)]} \tag{10.30}$$

and the error term E_N^{CT} can be seen from Eq. 10.19 to be,

$$E_N^{CT} = \frac{f^{(4)}(\eta)(b-a)(h)^4}{720} \quad , a < \eta < b \tag{10.31}$$

10.4 Gauss Quadrature Rule

Quadrature is a historic word which means *the area* under a curve, thus, an integral. In fact, we have seen three quadrature rules already, where any quadrature rule has the form,

$$I(f) \approx A_0 f(x_0) + A_1 f(x_1) + \cdots + A_k f(x_k) \tag{10.32}$$

with **nodes** $x_0, \ldots, x_k$ and **weights** $A_0, \ldots, A_k$.

Note that this encompasses all the following rules,

$$I(f) \approx (b-a)f(x) \quad \text{(Rectangle Rule)}$$
$$I(f) \approx \frac{(b-a)}{2}[f(a) + f(b)] \quad \text{(Trapezoidal Rule)}$$
$$I(f) \approx \frac{(b-a)}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] \quad \text{(Simpson's Rule)}$$

$\boxed{\star}$ Now note that there is no apparent freedom in choosing the nodes $x_0, x_1, \ldots, x_k$. The difficulty with the Composite Rules is that the nodes x_i **must be evenly spread**.

Thus a natural question to ask is ***whether it is possible to make a rule for polynomial of degree*** $\leq 2k+1$ ***by choosing the nodes appropriately?***

To discuss the Gaussian Rules in the more general context, we write the integral as

$$\int_a^b f(x)dx = \int_a^b w(x)g(x)dx$$

where the **weight function** $w(x)$ is assumed to be *non-negative* and *integrable* on $[a, b]$ and,

$$g(x) = \frac{f(x)}{w(x)}$$

Consider approximating the weighted integral,

$$\begin{aligned} I(g) &= \int_a^b g(x)w(x)dx \\ &\approx A_0 g(x_0) + A_1 g(x_1) + \cdots + A_k g(x_k) \\ &= \sum_{j=0}^{k} A_j g(x_j) = I_k(g) \end{aligned}$$

We can clearly see in the above equations that the nodes x_j and weights A_j are to be chosen so that $I_k(g)$ equals $I(g)$ exactly for polynomials $g(x)$ of *as higher degree as possible*. This is indeed the basic idea behind the **Gaussian Rule**.

10.4.1 Special case when $w(x) = 1$ and $\int_{-1}^{1} g(x)dx \approx \sum_{j=0}^{k} A_j g(x_j)$

In this case, we define the **Error** as,

$$\boxed{E_k(g) = \int_{-1}^{1} g(x)dx - \sum_{j=0}^{k} A_j g(x_j)} \tag{10.33}$$

Now the task is to **determine the weights A_j and nodes x_j to make the error 0**, i.e.

$$E_k(g) = 0 \tag{10.34}$$

for as high a degree polynomial $g(x)$ as possible.

Note that,

$$E_k(a_0 + a_1 x + \cdots + a_n x^m) = a_0 E_k(1) + a_1 E_k(x) + \cdots + a_m E_k(x^m) \tag{10.35}$$

Also, note that,

$$\begin{aligned} & E_k(g) = 0 \ \text{ for every polynomial of degree } \ \leq m \\ \iff \ & E_k(x^i) = 0, \ i = 0, 1, 2, \ldots, m. \end{aligned} \tag{10.36}$$

Hence, to show that $E_k(g) = 0$ is equivalent to showing that each $E_k(x^i) = 0$ for all $i = 0, 1, \ldots, m$. We now use exactly this equivalence.

For $k = 0$

We have,

$$\int_{-1}^{1} g(x)dx \approx A_0 g(x_0)$$

Due to the conditions, **we require** $E_k(1) = 0$ **and** $E_k(x) = 0$.

$$\begin{aligned} E_k(1) = 0 &\implies \int_{-1}^{1} dx - A_0 \implies A_0 = 2 \\ E_k(x) = 0 &\implies \int_{-1}^{1} x dx - A_0 x_0 \implies x_0 = 0 \end{aligned} \tag{10.37}$$

Therefore, we would be lead to the following formula,

$$\boxed{\int_{-1}^{1} g(x)dx \approx 2g(0)}$$

which is the **Mid-Point Rule!**

For $k = 1$

We have,

$$\int_{-1}^{1} g(x)dx \approx A_0 g(x_0) + A_1 g(x_1)$$

Note there are **four unknown parameters** to be determined.

Thus we require,

$$\begin{aligned} E_k(1) = 0 &\implies \int_{-1}^{1} dx - A_0 - A_1 = 0 \implies A_0 + A_1 = 2 \\ E_k(x) = 0 &\implies \int_{-1}^{1} x dx - (A_0 x_0 + A_1 x_1) = 0 \implies A_0 x_0 + A_1 x_1 = 0 \\ E_k(x^2) = 0 &\implies \int_{-1}^{1} x^2 dx - (A_0 x_0^2 + A_1 x_1^2) = 0 \implies A_0 x_0^2 + A_1 x_1^2 = \frac{2}{3} \\ E_k(x^3) = 0 &\implies \int_{-1}^{1} x^3 dx - (A_0 x_0^3 + A_1 x_1^3) = 0 \implies A_0 x_0^3 + A_1 x_1^3 = 0 \end{aligned} \tag{10.38}$$

On solving, we get,

$$\begin{aligned} A_0 &= 1 \\ A_1 &= 1 \\ x_0 &= \frac{1}{\sqrt{3}} \\ x_1 &= -\frac{1}{\sqrt{3}} \end{aligned}$$

This leads to the formula,

$$\boxed{\int_{-1}^{1} g(x)dx \approx g\left(\frac{1}{\sqrt{3}}\right) + g\left(-\frac{1}{\sqrt{3}}\right)}$$

which has a degree of precision 3.

For a general k

There are $2k+2$ parameters, weights $\{w_i\}_{i=0}^{k}$ and nodes $\{x_i\}_{i=0}^{k}$, to be determined.

The equations to be solved are,

$$E_k(x^i) = 0\,, \quad i = 0, 1, \ldots, 2k+1 \tag{10.39}$$

This leads to the following set of equations:

$$\boxed{\sum_{j=0}^{k} A_j x_j^i = \begin{cases} 0 & i = 1, 3, \ldots, 2k+1. \\ \frac{2}{i+1} & i = 0, 2, \ldots, 2k. \end{cases}} \tag{10.40}$$

However, note that the equations in Eq. 10.40 are non-linear equations and their solvability is not at all obvious. For that, we **may** have the following alternative

10.4.2 An Alternate Approach through Newton's Interpolation

First, choose $x_0, \ldots, x_k$ in (a, b) and write the Newton's Interpolating Polynomial (Theorem 8,8),

$$g(x) = p_k(x) + g[x_0, x_1, \ldots, x_k, x]\psi_k(x)$$

where $p_k(x)$ is the polynomial of degree $\leq k$ which interpolates $g(x)$ at $x_0, \ldots, x_k$ with,

$$\psi_k(x) = (x - x_0) \ldots (x - x_k)$$

This now leads to, after integrating both sides,

$$I(g) = I(p_k) + \underbrace{\int_a^b g[x_0, \ldots, x_k, x]\psi_k(x)w(x)dx}_{\text{Error Term}} \tag{10.41}$$

Now, if we write $p_k(x)$ in Lagrange form (Eq. 8.3) then,

$$p_k(x) = g(x_0)L_0(x) + g(x_1)L_1(x) + \cdots + g(x_k)L_k(x)$$

with,

$$L_i(x) = \prod_{j=0, i\neq j}^{k} \frac{(x - x_j)}{(x_i - x_j)}\ , \quad i = 0, \ldots, k$$

Thus, expanding $I(p_k)$ in Eq. 10.41 now yields,

$$\begin{aligned} I(p_k) &= \int_a^b p_k(x)w(x)dx \\ &= g(x_0)\int_a^b L_0(x)w(x)dx + \cdots + g(x_k)\int_a^b L_k(x)w(x)dx \end{aligned}$$

Hence we can write $I(p_k)$ simply as,

$$I(p_k) = A_0 g(x_0) + A_1 g(x_1) + \cdots + A_k g(x_k) \tag{10.42}$$

where coefficients A_i are simply,

$$A_i = \int_a^b L_i(x)w(x)dx\ , \quad i = 0, \ldots, k \tag{10.43}$$

Now, for the error term, **using the property of orthogonal polynomials**, for many $w(x)$, **we can find a polynomial** $P_{k+1}(x)$ such that,

$$\int_a^b P_{k+1}(x)q(x)w(x)dx = 0 \tag{10.44}$$

for all polynomial $q(x)$ of degree $\leq k$. Moreover, since $P_{k+1}(x)$ is a polynomial, thus,

$$P_{k+1}(x) = \alpha_{k+1}(x - \xi_0)(x - \xi_1) \ldots (x - \xi_k)$$

where $\xi_0, \ldots, \xi_k$ are the $k+1$ distinct points in the interval (a, b) at which P_{k+1} vanishes.

Now, **simply set,**

$$x_j = \xi_j \,, \quad j = 0, \ldots, k \tag{10.45}$$

to get

$$I(f) = I(p_k)$$

as the error term becomes zero. But the difficulty being finding the right P_{k+1}'s.

$\boxed{\star}$ Hence, if we choose the points $x_0, \ldots, x_k$ as the zeros of the polynomial $P_{k+1}(x)$ of degree $k+1$, which is orthogonal to the weight function $w(x)$ over (a, b) to any polynomial of degree $\leq k$, and if the coefficients A_i for $i = 0, \ldots, k$ are chosen in accordance with Eq. 10.43, then **the resulting Gaussian Quadrature Formula will then be EXACT for all polynomials of degree** $\leq 2k+1$.

But we see that determining $P_{k+1}(x)$ is not at all obvious. One thing we can do, however, is to choose the orthogonal polynomials P_{k+1} to be **Legendre Polynomials**! But the Legendre polynomials are only defined over $(-1, 1)$. Thus, one needs to transform original integral $\int_a^b f(x)dx$ over the limits -1 and 1 to write $\int_{-1}^{1} g(\tau)d\tau$. This can be done using the following change of variable,

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

Now, we can use the argument above to, for any k, construct the Orthogonal Legendre Polynomials, find it's roots ξ_i, subsequently exactly calculate the integral by Eq. 10.42, 10.43 and $x_i = \xi_i$.

Refer to Example in Lecture 20 for more rigidity.

Chapter 11

Numerical Solutions to IVPs for ODEs : One-Step Method

We first consider the first-order ordinary differential equations as we can use the same idea to extend these ideas to higher order ODEs as they can be written as a system of first order ODEs.

Consider a first-order IVP of the form

$$y' = f(x, y) \ , \ \ y(x_0) = y_0 \tag{11.1}$$

where the function f may be linear or non-linear.

Clearly, before jumping into the numerical techniques, we must first make sure concretely that the question we are trying to solve indeed has a solution and that the solution is unique. To that extent, for IVP in Eq. 11.1, we have the following theorem,

Theorem 11.13 : **(Unique Solution of Eq. 11.1)** *If $f(x, y)$ is continuous in a rectangle*

$$R = \{(x, y) \ : \ |x - x_0| \leq a \ , \ \ |y - y_0| \leq b\}$$

satisfies a Lipschitz condition in the second variable

$$|f(x, y_1) - f(x, y_2)| \leq L|y_1 - y_2| \ \ \forall (x, y_1), (x, y_2) \in \mathbb{R}^2$$

*Then the IVP in Eq. 11.1 has **a unique solution** in the interval $|x - x_0| \leq \alpha$ where,*

$$\alpha = \min\left\{a, \frac{b}{M}\right\} \ \ \textit{and} \ \ M = \max_R |f(x, y)|$$

11.1 Taylor-Series Method

Note that we can expand the $y(x)$ into it's Taylor Series to get,

$$\begin{aligned} y(x) &= y(x_0) + (x - x_0)y'(x_0) + \frac{(x - x_0)^2}{2!}y''(x_0) + \dots \\ &= y_0 + hy'(x_0) + \frac{h^2}{2!}y''(x_0) + \dots \end{aligned}$$

where $x - x_0 = h$. **If f is sufficiently differentiable with respect to x and y**, we can compute,

$$\begin{aligned} y' &= f(x, y) \\ y'' &= f' = f_x + f_y y' = f_x + f_y f \\ y''' &= f'' = f_{xx} + f_{xy}f + f_{yx}f + f_{yy}f^2 + f_y f_x + f_y^2 f \\ &\text{and so on} \dots \end{aligned}$$

Now let y_n **be approximations to the true solutions** $y(x_n)$ at the points $x_n = x_0 + nh$, $n = 0, 1, 2, \dots$. That is,

$$y_n \approx y(x_n)$$

We thus arrive at the *Taylor's Algorithm of Order k.*

11.1.1 Taylor's Algorithm of Order k

Algorithm: To find an approximate solution of the IVP

$$y' = f(x, y) \ , \ \ y(x_0) = y_0$$

over an interval $[a, b]$.

Step 1 : Choose a step size $h = \frac{b-a}{N}$. Set,

$$x_n = x_0 + nh \ , \ \ n = 0, 1, \dots, N$$

Step 2 : Generate approximations y_n to $y(x_n)$ from the recursion

$$y_{n+1} = y_n + hT_k(x_n, y_n) \ , \ \ n = 0, 1, \dots, N - 1$$

where T_k is,

$$T_k(x, y) = f(x, y) + \frac{h}{2!}f'(x, y) + \dots \frac{h^{k-1}}{k!}f^{(k-1)}(x, y) \ , \ \ k = 1, 2, \dots$$

Here, $f^{(j)}$ denotes the j^{th} derivative of the function $f(x, y(x))$ with respect to x.

So we are basically truncating the Taylor series to k^{th} order and take the h common out from them as $y' = f$.

Note that we calculate y at $x = x_{n+1}$ by using only the information about y and it's derivatives at the previous step $x = x_n$. Such a method is thus called **One-Step/Single-Step Method**.

Error in Taylor's Algorithm

Taylor's Theorem with remainder term shows the local error of Taylor's Algorithm of order k can be written as,

$$E = \frac{h^{k+1}}{(k+1)!} f^{(k)}(\xi, y(\xi)) = \frac{h^{k+1}}{(k+1)!} y^{(k+1)}(\xi)\ , \quad x_n < \xi < x_n + h \tag{11.2}$$

The Taylor algorithm is said to be of **order** k **if the local error** E **is** $O(h^{(k+1)})$.

11.2 Euler's Method

Setting $k = 1$ (Taylor's Algorithm of Order 1), we obtain,

$$y_{n+1} = y_n + hf(x_n, y_n) \tag{11.3}$$

which is known as **Euler's Method**. Note that the x_n is still calculated as per the Taylor's Algorithm. The **Local Error** is simply

$$E = \frac{h^2}{2} y''(\xi) \tag{11.4}$$

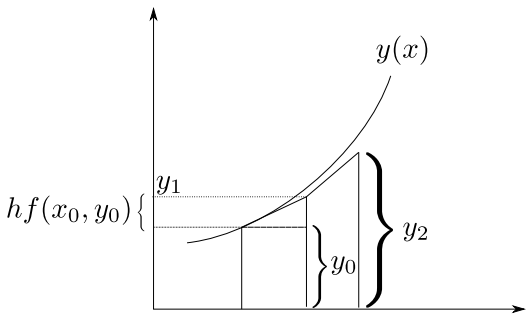


Figure 11.1: Geometric Interpretation of Euler's Method.

Error in Euler's Method

Theorem 11.14 : **(Error Estimate)** *Let y_n be the approximate solution of*

$$y' = f(x, y)\ ,\ \ y(x_0) = y_0$$

as generated by Euler's Method. If $y'' \in C([x_0, b])$ and

$$\left|f_y(x, y)\right| < L\ ,\ \ \left|y''(x)\right| < M\ , x \in [x_0, b]$$

*for fixed positive constants L and M, then, the **error** $e_n = y(x_n) - y_n$ **of Euler's method at a point** $x_n = x_0 + nh$ **is bounded** as follows,*

$$|e_n| \leq \frac{hM}{2L}\left[e^{(x_n - x_0)L} - 1\right]$$

This theorem shows that the error is $O(h)$ and $e_n \to 0$ as $h \to 0$.

Refer to Lecture 21 for a small example.

11.3 Runge-Kutta Method

We previously saw that in Taylor's Algorithm of order k and that in Euler's Method, one needs higher order derivatives of a function. But we definitely don't have that information with us. Moreover, the approximations done by such methods only produce results with low error when h is low enough. Thus, we try to look at the methods which only require function calculations instead of requiring to calculate an explicit form of other functions.

To that extent, the Runge-Kutta Method plays a major step. In this method, we seek a formula of the form,

$$\begin{aligned} y_{n+1} &= y_n + ak_1 + bk_2 \\ k_1 &= hf(x_n, y_n)\ ,\ \ k_2 = hf(x_n + \alpha h, y_n + \beta k_1) \end{aligned} \tag{11.5}$$

where the **constants a, b, α, β are to be determined so that Eq. 11.5 will agree with the Taylor Algorithm of as high an order as possible.**

Expanding $y(x_{n+1})$ in a Taylor's Series yields us (keep the original problem, Eq. 11.1, in mind),

$$\begin{aligned} y(x_{n+1}) = y(x_n + h) &= y(x_n) + hy'(x_n) + \frac{h^2}{2}y''(x_n) + \frac{h^3}{6}y'''(x_n) + \dots \\ &= y(x_n) + hf(x_n, y_n) + \frac{h^2}{2}(f_x + f_y f)_n \\ &\quad + \frac{h^3}{6}(f_{xx} + 2f_{xy}f + f_{yy}f^2 + f_y f_x + f_y^2 f)_n + O(h^4) \end{aligned} \tag{11.6}$$

On the other hand, expanding $f(x_n + \alpha h, y_n + \beta k_1)$ as in Eq. 11.5 by using Taylor's Expansion for functions of two variables, we have,

$$\begin{aligned} \frac{k_2}{h} = f(x_n + \alpha h, y_n + \beta k_1) =& f(x_n, y_n) + \alpha h f_x + \beta k_1 f_y + \frac{\alpha^2 h^2}{2} f_{xx} \\ &+ (\alpha h \beta k_1) f_{xy} + \frac{\beta^2 k_1^2}{2} f_{yy} + O(h^3) \end{aligned} \tag{11.7}$$

where all derivatives are evaluated at (x_n, y_n). Substituting the expression for k_2 above in Eq. 11.5 and using $k_1 = hf(x_n, y_n)$, we get upon rearrangement in powers of h that,

$$\begin{aligned} y_{n+1} &= y_n + (a+b)hf + bh^2(\alpha f_x + \beta f f_y) \\ &\quad + bh^3\left(\frac{\alpha^2}{2}f_{xx} + \alpha\beta f f_{xy} + \frac{\beta^2}{2}f^2 f_{yy}\right) + O(h^4) \end{aligned} \tag{11.8}$$

Now, simply **comparing the corresponding powers of H and h^2 from Eq. 11.6 and Eq. 11.8**, we must have,

$$\begin{aligned} a + b &= 1 \\ b\alpha = b\beta &= \frac{1}{2} \end{aligned}$$

Note that there are many solutions to the above system, the simplest perhaps being,

$$a = b = \frac{1}{2}$$
$$\alpha = \beta = 1$$

and we use this in the following algorithm.

Runge-Kutta Method of Order 2

Algorithm: For the equation,

$$y' = f(x, y)\ ,\ \ y(x_0) = y_0$$

generate approximations y_n to the exact solution $y(x_0 + nh)$, for fixed h, using the recursion formula,

$$\begin{aligned} y_{n+1} &= y_n + \frac{1}{2}(k_1 + k_2) \\ k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + h, y_n + k_1) \end{aligned} \tag{11.9}$$

Also, the local error is of the form (refer Eq. 11.6 and Eq. 11.8)

$$\begin{aligned} y(x_{n+1}) - y_{n+1} &= \frac{h^3}{12}(f_{xx} + 2f_{xy}f + f_{yy}f^2 + f_y f_x + f_y^2 f) + O(h^4) \\ &= O(h^3) \end{aligned} \tag{11.10}$$

Note that the this method thus takes average slope between two points as the quantity to add to current estimate at the previous point h distance away.

Runge-Kutta Method of Order 4

Algorithm: For the equation,

$$y' = f(x, y)\ ,\ \ y(x_0) = y_0$$

generate approximations y_n to the exact solution $y(x_0 + nh)$, for fixed h and $n = 0, 1, \ldots$ using the following recursion formula,

$$\begin{aligned} y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\ k_3 &= hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\ k_4 &= hf(x_n + h, y_n + k_3) \end{aligned} \tag{11.11}$$

Also, the **Local Discretization Error** would be of order $O(h^5)$ (Eq. 11.6 & Eq. 11.8).

11.4 Systems of Differential Equations

Now we would use the previous methods for solving first order IVPs to solve an N^{th} order equation of the form,

$$y^{(N)}(x) = f(x, y(x), y'(x), \ldots, y^{(N-1)}(x)) \tag{11.12}$$

can be written as a system of N first order equations through substitution. With $y_1 = y$, set,

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= y_3 \\ y_3' &= y_4 \\ &\vdots \\ y_{N-1}' &= y_N \\ y_N' &= f(x, y_1, y_2, \ldots, y_N) \end{aligned} \tag{11.13}$$

More generally, a system of N first-order equations will have the form,

$$\begin{aligned} y_1' &= f_1(x, y_1, y_2, \ldots, y_N) \\ y_2' &= f_2(x, y_1, y_2, \ldots, y_N) \\ &\vdots \\ y_N' &= f_N(x, y_1, y_2, \ldots, y_N) \end{aligned} \tag{11.14}$$

Fourth-order Runge-Kutta Method for the system of two equations

For simplicity, we now illustrate Fourth-order Runge-Kutta Method for the system of two equations of the form

$$\begin{aligned} y' &= f(x, y, z) \\ z' &= g(x, y, z) \end{aligned} \tag{11.15}$$

with the initial conditions $y(x_0) = y_0$ and $z(x_0) = z_0$.

Algorithm:

1. Set $x_n = x_0 + nh$ where $n = 0, 1, \ldots$.

2. Generate approximations y_n and z_n to the exact solutions $y(x_n)$ and $z(x_n)$, using the recursion formula,

$$\begin{aligned} y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ z_{n+1} &= z_n + \frac{1}{6}(l_1 + 2l_2 + 2l_3 + l_4) \end{aligned} \tag{11.16}$$

where,

$$\begin{aligned} k_1 &= hf(x_n, y_n, z_n) \\ l_1 &= hg(x_n, y_n, z_n) \\ k_2 &= hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}, z_n + \frac{l_1}{2}\right) \\ l_2 &= hg\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}, z_n + \frac{l_1}{2}\right) \\ k_3 &= hf\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}, z_n + \frac{l_2}{2}\right) \\ l_3 &= hg\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}, z_n + \frac{l_2}{2}\right) \\ k_4 &= hf\left(x_n + h, y_n + k_3, z_n + l_3\right) \\ l_4 &= hg\left(x_n + h, y_n + k_3, z_n + l_3\right) \end{aligned} \tag{11.17}$$

Chapter 12

Numerical Solutions to IVPs for ODEs : Multi-Step Methods

The One-Step methods discussed in previous chapter (Taylor's Method, Euler Method, Runge-Kutta Methods) use the information on the solution of the previous step to compute the solution at the next step. We now discuss Mutli-Step Methods, where we make use of the information about the solution at more than one point.

Consier the IVP of the form

$$y' = f(x, y) \ , \ \ y(x_0) = y_0 \tag{12.1}$$

Now, let us **assume that we have already obtained approximations to y' and y at a number of equally spaced points, say $x_0, x_1, \ldots, x_n$.** Integrating the differential equation 12.1 from x_n to x_{n+1}, we get,

$$\begin{aligned} &\int_{x_n}^{x_{n+1}} y' dx = \int_{x_n}^{x_{n+1}} f(x, y(x)) dx \\ \implies \quad &y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} f(x, y(x)) dx \end{aligned} \tag{12.2}$$

as we don't have any information about the RHS integral, thus we **now simply approximate $f(x, y(x))$ by a polynomial $p_m(x)$ which interpolates $f(x, y(x))$ at the $m + 1$ points $x_n, x_{n-1}, x_{n-2}, \ldots, x_{n-m}$** seen previously. Set,

$$f(x_k, y(x_k)) = f_k$$

12.0.1 Adams-Bashford Method

Using Newton's backward formula of degree m for this purpose, we form $p_m(x)$ as the following,

$$p_m(x) = \sum_{k=0}^{m} \binom{-s}{k} \Delta^k f_{n-k} \text{ where } s = \frac{x - x_n}{h} \tag{12.3}$$

where Δ **is the forward difference operator** and $\Delta f_i = f_{i+1} - f_i$ and $\Delta^2 f_i = \Delta(\Delta f_i) = \Delta(f_{i+1} - f_i) = f_{i+2} - 2f_{i+1} + f_i$.

Inserting this equation into the integral in Eq. 12.2 and noting that $hds = dx$ and also appropriately changing the limits, we get,

$$\begin{aligned} y_{n+1} &= y_n + h \int_0^1 \sum_{k=0}^{m} \binom{-s}{k} \Delta^k f_{n-k} \\ &= y_n + h \left[\gamma_0 f_n + \gamma_1 \Delta f_{n-1} + \cdots + \gamma_m \Delta^m\right] \\ &\text{where,} \\ \gamma_k &= (-1)^k \int_0^1 \binom{-s}{k} ds \end{aligned} \tag{12.4}$$

The Eq. 12.4 is called the **Adams-Bashford Method**.

For example, consider $m = 3$. Thus, from Eq. 12.4, we have,

$$y_{n+1} = y_n + h\left(f_n + \frac{1}{2}\Delta f_{n-1} + \frac{5}{12}\Delta^2 f_{n-2} + \frac{3}{8}\Delta^3 f_{n-3}\right)$$

Now, expanding the forward-difference formulas yields us,

$$\begin{aligned} \Delta f_{n-1} &= f_n - f_{n-1} \\ \Delta^2 f_{n-2} &= f_n - 2f_{n-1} + f_{n-2} \\ \Delta^3 f_{n-3} &= f_n - 3f_{n-1} + 3f_{n-2} - f_{n-3} \end{aligned}$$

Substituting them, we get,

$$y_{n+1} = y_n + \frac{h}{24}(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3})$$

Note that the local discretization error would be,

$$E_{AB} = h^5 y^{(5)}(\xi)\frac{251}{720} = O(h^5)$$

Remarks:

- One major disadvantage of Multi-Step formulas is that they are **not self-starting**. These starting values must be obtained by some independent method like the Single-Step methods discussed in the chapter earlier.
- Another disadvantage of the Adams-Bashford method is that, although the local discretization error is $O(h^5)$, the **coefficient in the error term is somewhat larger than for formulas of the Runge-Kutta** type of the same order.
- On the other hand, the **Multi-Step formula require only one derivative evaluation per step**, compared with four evaluations per step with Runge-Kutta methods and is therefore considered faster and requires less computational load.

12.1 Predictor-Corrector Methods

We now see another type of method for solving ODEs as a member of Multi-Step methods. we Continue from Eq. 12.2, which we write here again for brevity,

$$\int_{x_n}^{x_{n+1}} y' dx = \int_{x_n}^{x_{n+1}} f(x, y(x)) dx$$
$$\implies \quad y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} f(x, y(x)) dx$$

$\boxed{\star}$ Now, **approximating the integral by the trapezoidal rule**, we obtain,

$$y_{n+1} = y_n + \frac{h}{2} \left[f(x_n, y_n) + f(x_{n+1}, y_{n+1}) \right] \ , \ \ n = 0, 1, 2, \ldots \tag{12.5}$$

Note that the error of this formula is simply $E = -\frac{h^3}{12} y'''$ (Chapter 10, Eq. 10.17). Thus, the formula Eq. 12.5 is known as **Improved Euler Method**.

Note that the Eq. 12.5 is an implicit equation for y_{n+1}.

$\boxed{\star}$ If $f(x, y)$ is a non-linear function, then it is difficult to solve Eq. 12.5 for y_{n+1} exactly. However, one can attempt to obtain y_{n+1} by means of iteration. Thus, **keeping x_n fixed, we obtain a first approximation to y_{n+1} by means of Euler's Formula**,

$$y_{n+1}^{(0)} = y_n + h f(x_n, y_n)$$

Then, **compute** $f(x_{n+1}, y_{n+1}^{(0)})$ and obtain the following approximation,

$$y_{n+1}^{(1)} = y_n + \frac{h}{2}\left[f(x_n, y_n) + f(x_{n+1}, y_{n+1}^{(0)})\right]$$

Continue the iteration to calculate $f(x_{n+1}, y_{n+1}^{(1)})$ and obtain $y_{n+1}^{(2)}$ as,

$$y_{n+1}^{(2)} = y_n + \frac{h}{2}\left[f(x_n, y_n) + f(x_{n+1}, y_{n+1}^{(1)})\right]$$

In general, the iteration is given by,

$$y_{n+1}^{(k)} = y_n + \frac{h}{2}\left[f(x_n, y_n) + f(x_{n+1}, y_{n+1}^{(k-1)})\right] \ , \ \ k = 1, 2, \ldots$$

This leads us to the following algorithm.

12.1.1 Second-Order Predictor-Corrector Method

Algorithm: For the differential equation

$$y' = f(x, y) \ , \ \ y(x_0) = y_0$$

with h given and $x_n = x_0 + nh$, for each fixed $n = 0, 1, \ldots$. Now,

1. Compute $y_{n+1}^{(0)}$ using,

$$y_{n+1}^{(0)} = y_n + hf(x_n, y_n)$$

2. Compute $y_{n+1}^{(k)}$ where $k = 1, 2, \ldots$, using,

$$y_{n+1}^{(k)} = y_n + \frac{h}{2}\left[f(x_n, y_n) + f(x_{n+1}, y_{n+2}^{(k-1)})\right]$$

iterating on k untill

$$\frac{\left|y_{n+1}^{(k)} - y_{n+1}^{(k-1)}\right|}{\left|y_{n+1}^{(k)}\right|} < \epsilon$$

for a prescribed tolerance.

Remarks:

- It is customary to call an explicit formula such as Euler's Formula an **Open Type** formula, while an implicit formula such as 12.5 is said to be of **Closed Type**.
- When they are used as a pair of formulas, the open-type formula is called a **predictor**, while the closed-type formula is called a **corrector**.

A natural question to ask now would be that ***what are the conditions on which the inner iteration on*** k ***will converge?***

To that extent we have the following theorem.

Theorem 12.15 : *Let* $f(x,y)$, $\frac{\partial f}{\partial y} \in C(R)$. *Then, the inner iteration defined by,*

$$y_{n+1}^{(k)} = y_n + \frac{h}{2}\left[f(x_n, y_n) + f(x_{n+1}, y_{n+1}^{(k-1)})\right] \ , \ k = 1, 2, \ldots$$

will converge*, provided* h ***is chosen small enough so that****, for* $x = x_n$ *and for all* y *with* $|y - y_{n+1}| < |y_{n+1}^{(0)} - y_{n+1}|$,

$$\left|\frac{\partial f}{\partial y}\right| h < 2$$

Proof : Observe that with x_n fixed and setting $y_{n+1}^{(k)} = Y^{(k)}$, we can write the iteration in Eq. 12.5 in the form,

$$Y^{(k)} = F\left(Y^{(k-1)}\right)$$

where $F(Y) = \frac{h}{2} f(x_{n+1}, Y) + C$. Here C depends on n not on Y.

Now, consider the above equation as the fixed point iteration with iteration function $F(Y)$. Now, since we know that this would converge when,

$$\left|F'(Y)\right| < 1 \ \forall \ Y \text{ with } |Y - y_{n+1}| < |Y^{(0)} - y_{n+1}|$$

where y_{n+1} is the fixed point of $F(Y)$.

Thus, iteration would converge if,

$$|F'(Y)| \leq \left|\frac{h}{2}\frac{\partial f}{\partial y}\right| < 1 \implies h < \left|\frac{2}{\frac{\partial f}{\partial y}}\right|$$

QED.

12.1.2 The Adams-Moulton Method

Continuing from the following recurrence relation,

$$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} f(x, y(x))dx$$

for the IVP in Eq. 12.1. We now approximate $f(x, y(x))$ by a polynomial which interpolates at $x_{n+1}, x_n, \ldots, x_{n-m}$ for an integer $m > 0$. Then, use of **Newton's backward interpolation formula which interpolates f at these $\mathbf{m+2}$ points** (instead of $m+1$) in terms of $s = \frac{x-x_n}{h}$ yields,

$$p_{m+1}(s) = \sum_{k=0}^{m+1} (-1)^k \binom{1-s}{k} \Delta^k f_{n+1-k}$$

These differences are based on the $m+2$ values $f_{n+1}, f_n, \ldots, f_{n-m}$. Replacing f by p_{m+1} in Eq. 12.2, we get the following,

$$\begin{aligned} y_{n+1} &= y_n + h \int_0^1 \sum_{k=0}^{m+1} (-1)^k \binom{1-s}{k} \Delta^k f_{n+1-k} ds \\ &= y_n + h \left[\gamma_0' f_{n+1} + \gamma_1' \Delta f_n + \cdots + \gamma_{m+1}' \Delta^{m+1} f_{n-m} \right] \end{aligned} \tag{12.6}$$

where,

$$\gamma_k' = (-1)^k \int_0^1 \sum_{k=0}^{m+1} \binom{1-s}{k} ds \ , \quad k = 0, 1, \ldots, m+1$$

This formula is known as **Adams-Moulton Multi-Step Formula**.

For Example, consider $m = 2$ and the values of coefficients are $\gamma_0' = 1, \gamma_1' = -\frac{1}{2}, \gamma_2' = -\frac{1}{12}$ and $\gamma_3' = -\frac{1}{24}$, we obtain the formula,

$$y_{n+1} = y_n + h \left(f_{n+1} - \frac{1}{2}\Delta f_n - \frac{1}{12}\Delta^2 f_{n-1} - \frac{1}{24}\Delta^3 f_{n-2} \right)$$

Putting in the values of the forward difference operators, we simply obtain,

$$y_{n+1} = y_n + \frac{h}{24}(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}) \tag{12.7}$$

and the error in this example would be,

$$E_{AM} = -\frac{19}{720} h^5 y^{(iv)}(\xi)$$

This result in this example is used in demonstrating the following algorithm

The Adams-Moulton Predictor-Corrector Method

Algorithm: For the IVP,

$$y' = f(x,y)\ ,\ \ y(x_0) = y_0$$

and with fixed h, $x_n = x_0 + nh$ and given $(y_0, f_0), (y_1, f_1), (y_2, f_2), (y_3, f_3)$, where $f_i = f(x_i, y_i)$. Note that we have $m = 3$

1. Compute $y^{(0)}_{n+1}$ using the **Adams-Bashforth Formula** as in the example in it's section,

$$y^{(0)}_{n+1} = y_n + \frac{h}{24}(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3})\ ,\ \ n = 3,4,\ldots$$

2. Compute $f^{(0)}_{n+1} = f(x_{n+1}, y^{(0)}_{n+1})$.

3. Improve the value of y_{n+1} using the **Adams-Moulton Muti-Step Formula** (Eq. 12.7),

$$y^{(k)}_{n+1} = y_n + \frac{h}{24}\left(9f(x_{n+1}, y^{(k-1)}_{n+1}) + 19f_n - 5f_{n-1} + f_{n-2}\right)\ \ k = 1,2,\ldots$$

4. Iterate on k until,

$$\frac{\left|y^{(k)}_{n+1} - y^{(k-1)}_{n+1}\right|}{\left|y^{(k)}_{n+1}\right|} < \epsilon\ ,\ \ \text{for prescribed tolerance } \epsilon$$

12.2 Stability of Multi-Step Methods [1]

Consider the IVP as discussed earlier,

$$y' = f(x,y)\ ,\ \ y(x_0) = y_0$$

Integrating from x_{n-1} to x_{n+1} yields,

$$\int_{x_{n-1}}^{x_{n+1}} y'dx = \int_{x_{n-1}}^{x_{n+1}} f(x,y)dx$$

[1] This section contains some details which are rigorously explained in the following chapter.

Now, **using the mid-point rule for the integral on RHS** leads to the following multi-step method,

$$y_{n+1} = y_{n-1} + 2hf_n \tag{12.8}$$

Since we want to focus on the stability of this multistep method, thus, for the instance, let us **consider the following problem as an example**,

$$y' = -2y + 1\ ,\ \ y(0) = 1 \tag{12.9}$$

The exact solution of this can be found to be simply,

$$y(x) = \frac{1}{2}e^{-2x} + \frac{1}{2} \tag{12.10}$$

Applying the multi-step method with $f_n = f(x_n, y_n) = -2y + 1$ (Eq. 12.8), we get,

$$y_{n+1} + 4hy_n - y_{n-1} = 2h\ ,\ \ y_0 = 1$$

$\boxed{\checkmark}$ Following the method to solve a Linear Difference Equation in the next chapter, we find the characteristics polynomial is (for general solution),

$$\beta^2 + 4h\beta - 1 = 0$$

which gives us,

$$\beta_1 = -2h + \sqrt{1 + 4h^2}\ \ \beta_2 = -2h - \sqrt{1 + 4h^2}$$

Now, expanding $\sqrt{1 + 4h^2}$ using Taylor Series gets us,

$$\beta_1 \approx 1 - 2h + O(h^2)\ \text{ and }\ \beta_2 = -1 - 2h + O(h^2)$$

Note the particular solution is simply $y_n^{PS} = \frac{1}{2}$, thus the general solution is simply (refer Chapter 13 for more details),

$$y_n = C_1(1 - 2h + O(h^2))^n + C_2(-1 - 2h + O(h^2))^n + \frac{1}{2} \tag{12.11}$$

Note that $n = \frac{x_n}{h}$ as $x_0 = 0$. Thus **for fixed** x_n, we have this *important observation*,

$$\lim_{h\to\infty}(1 + 2h)^n = \lim_{h\to\infty}(1 + 2h)^{\frac{1}{2h}2x_n} = e^{2x_n}$$

$$\text{Similarly, } \lim_{h\to\infty}(1 - 2h)^n = e^{-2x_n}$$

Hence, **in the limit** $h \to 0$**, the solution of Eq. 12.9 as written in eq. 12.11 becomes**,

$$y_n = \left(C_1e^{-2x_n} + \frac{1}{2}\right) + C_2(-1)^n e^{2x_n} \tag{12.12}$$

Remarks:-

- Note that **the first term tends to the true solution (Eq. 12.11) of the differential equation**. The **second term is extraneous and arises only because we have replaced a first-order difference equation by a second-order difference equation!**

- The error term induced from extraneous solution will eventually dominate the true solution and lead to completely incorrect results!!

Now, for a general case, suppose that a multi-step method leads to a difference equation of order k whose characteristics equation is,

$$\beta^k + a_{k-1}\beta^{k-1} + \cdots + a_0 = 0 \tag{12.13}$$

If Eq. 12.13 has k distinct zeros, say $\beta_1, \beta_2, \ldots, \beta_k$. The GS of the corresponding homogeneous difference equation is then,

$$y_n = c_1\beta_1^n + c_2\beta_2^n + \cdots + c_k\beta_n^k \;\forall\; n$$

$\boxed{\star}$ Now, **one of these solutions, say β_1^n, will tend to the exact solution of the differential equation as $h \to 0$. All the other solutions $(\beta_2^n, \ldots, \beta_k^n)$ are thus extraneous.** We hence arrive at the following definition.

Definition: *A multi-step method is said to be* ***strongly stable*** *if the extraneous roots satisfy the condition*

$$|\beta_i| < 1\ ,\quad i = 2, 3, \ldots, k$$

It's trivial to see that if $|\beta_i| > 1$ for any $i = 2, 3, \ldots, k$, then the errors will grow exponentially.

Note : First example in Lecture 25 shows that **Adams-Bashforth method is *strongly-stable***. We demonstrate below the stability of Milne's Method.

12.2.1 Example : Stability of Milne's Method

Consider the IVP,

$$y\prime = \lambda y\ ,\quad y(0) = 1$$

The Milne's method is given by,

$$y_{n+1} = y_{n-1} + \frac{h}{3}(f_{n+1} + 4f_n + f_{n-1})$$

Since $f(x,y) = \lambda y$, it follows that,

$$y_{n+1} - y_{n-1} - \frac{h\lambda}{3}(y_{n+1} + 4y_n + y_{n-1}) = 0$$

It's characteristics equation is thus given by,

$$\begin{aligned} p(\beta) + h\lambda q(\beta) = 0 \ \text{ where,} \\ p(\beta) = \beta^2 - 1 \\ q(\beta) = -\frac{1}{3}(\beta^2 + 4\beta + 1) \end{aligned} \tag{12.14}$$

Now, as $h \to 0$, we would have the following,

$$p(\beta) = \beta^2 - 1 = 0$$

which has two roots, $\beta_1 = 1$, $\beta_2 = -1$. By definition, hence, **Milne's method is not strongly stable**!

Moreover, for small h, the roots of the stability polynomial Eq. 12.14 is,

$$\begin{aligned} \beta_1 &= 1 + \lambda h + O(h^2) \\ \beta_2 &= -\left(1 - \frac{\lambda h}{3}\right) + O(h^2) \end{aligned}$$

The general solution is thus,

$$y_n = c_1(1 + \lambda h + O(h^2))^n + c_2(-1)^n \left(1 - \frac{\lambda h}{3} + O(h^2)\right)^n \tag{12.15}$$

Set $n = \frac{x_n}{h}$ and let $h \to 0$, the solution thus approaches,

$$y_n = \underbrace{c_1 e^{\lambda x_n}}_{y_{d,n}(x)} + \underbrace{c_2(-1)^n e^{-\frac{\lambda x_n}{3}}}_{y_{e,n}(x)}$$

$\boxed{\star}$ **Remarks :**

- If $\lambda > 0$, the **desired solution** $y_{d,n}(x)$ is exponentially increasing and the **extraneous solution** $y_{e,n}(x)$ is exponentially decreasing. In that case, the Milne's method would be **stable**, NOT strongly stable though.
- On the other hand, if $\lambda < 0$, then Milne's method would be **unstable** as the extraneous solution $y_{e,n}(x)$ would be increasing exponentially.
- Methods of this type whose stability depends on the the sign of λ for the test equation $y' = \lambda y$ are said to be **weakly stable**.

Chapter 13

Solutions of Linear Difference Equations

We now turn our attention to finding solutions of the linear difference equations (LDE). An LDE of **order** N is of the form,

$$y_{n+N} + a_{N-1}y_{n+N-1} + a_{N-2}y_{n+N-2} + \cdots + a_0 y_n = b_n \tag{13.1}$$

where $a_{N-1}, a_{N-2}, \ldots, a_0$ are constants.

$\boxed{\star}$ If $b_n = 0$, then Eq. 13.1 is called a **Homogeneous LDE**.

13.1 Solutions to homogeneous LDE

We seek solutions of the form $\boxed{y_n = \beta^n}$ for all n. Thus, substituting, we get,

$$y_{n+N} = \beta^{n+N} \ , \ \ y_{n+N-1} = \beta^{n+N-1} \ , \ \ \ldots$$

into Eq. 13.1 with $b_n = 0$ (homogeneous), to get,

$$\beta^{n+N} + a_{N-1}\beta^{n+N-1} + \cdots + a_0\beta^n = 0$$

Thus,

$$\boxed{p(\beta) = \beta^N + a_{N-1}\beta^{N-1} + \cdots + a_0 = 0} \tag{13.2}$$

which is known as **characteristics equation**.

Now with regards to the characteristics equation in Eq. 13.2, we can have following cases,

CASE I : *Assume that Eq. 13.2 has N distinct zeros* $\beta_1, \beta_2, \ldots, \beta_N$.

Thus, we have that

$$\beta_1^n, \beta_2^n, \ldots, \beta_N^n$$

are all solutions to Eq. 13.1.

Thus, by linearity, **we can write the general solution (GS) as**,

$$\boxed{y_n = c_1\beta_1^n + c_2\beta_2^n + \cdots + c_N\beta_N^n \quad \forall n} \tag{13.3}$$

where $c_1, c_2, \ldots, c_N$ are arbitrary constants.

Remark : If the first $N-1$ values of y_n are given, the resulting initial-value difference equation can be solved explicitly for all succeeding values of n. That is to say that if we are given y_i for $i = 0, \ldots, N-1$, then we can solve for the constants c_i that arise in the general solution.

CASE II : *If the characteristic equation (Eq. 13.2) has a pair of conjugate-complex roots, the solution can still be expressed in real form.*

Thus, if $\beta_1 = \alpha + i\beta$ and $\beta_2 = \alpha - i\beta$, then we write,

$$\begin{aligned} \beta_1 &= re^{i\theta} \\ \beta_2 &= re^{-i\theta} \end{aligned} \tag{13.4}$$

where $r = \sqrt{\alpha^2 + \beta^2}$ and $\theta = \tan^{-1}\left(\frac{\beta}{\alpha}\right)$. Thus, **the general solution (GS) corresponding to this pair is** simply,

$$\begin{aligned} y_n = c_1\beta_1^n + c_2\beta_2^n &= c_1 r^n e^{in\theta} + c_2 r^n e^{-in\theta} \\ &= r^n\left[c_1(\cos n\theta + i\sin n\theta) + c_2(\cos n\theta - i\sin n\theta)\right] \\ &\boxed{= r^n(C_1\cos n\theta + C_2\sin n\theta)} \end{aligned} \tag{13.5}$$

CASE III : *If* β_1 *is a double root of the characteristic equation Eq. 13.2, then a second solution of the homogeneous LDE is* $n\beta_1^n$.

Thus, if we substitute $y_n = n\beta_1^n$ in the homogeneous LDE, we get,

$$\begin{aligned}(n+N)\beta_1^{n+N} + a_{N-1}(n+N-1)\beta_1^{n+N-1} + \cdots + a_0 n\beta_1^n \\ = \beta_1^n[n(\beta_1^N + a_{N-1}\beta_1^{N-1} + \cdots + a_0) + \beta_1(N\beta_1^{N-1} + a_{N-1}(N-1)\beta_1^{N-2} \\ + \cdots + a_1)] \\ = \beta_1^n\left[np(\beta_1) + \beta_1'(\beta_1)\right] = 0\end{aligned}$$

13.2 Solution to Non-homogeneous LDE with constant coefficients

The General Solution of the equation

$$y_{n+N} + a_{N-1}y_{n+N-1} + \cdots + a_0 y_n = b_n \tag{13.6}$$

can be written in the form,

$$y_n = y_n^{GS} + y_n^{PS}$$

where y_n^{GS} **is the General Solution of the Homogeneous LDE (i.e. for** $b_n = 0$**) and** y_n^{PS} **is the particular solution of the Eq. 13.6.**

Note that in the **special case that** $b_n = b$ **is a constant**, one can find particular solution by simply substituting $y_n^{PS} = A$ (a constant). This constant would be simply,

$$A = \frac{b}{1 + a_{N-1} + \cdots + a_0}$$

provided, of-course, $1 + a_{N-1} + \cdots + a_0 \neq 0$.

Chapter 14

Finite Difference Approximations to Derivatives

Consider the two-dimensional second-order PDE:

$$a\frac{\partial^2 U}{\partial x^2} + b\frac{\partial^2 U}{\partial x \partial y} + c\frac{\partial^2 U}{\partial y^2} + d\frac{\partial U}{\partial x} + e\frac{\partial U}{\partial y} + fU + g = 0 \qquad (14.1)$$

where a, b, c, d, e, f, g **may** be functions of the independent variables x and y and the dependent variable $U = U(x, y)$.

We can classify the PDE 14.1 on the basis of the coefficients as follows,

- **Elliptic** : when $b^2 - 4ac < 0$.
- **Parabolic** : when $b^2 - 4ac = 0$.
- **Hyperbolic** : when $b^2 - 4ac > 0$.

The above classification occur many times in real world as the following examples show.

Elliptic Equations : These problems are generally associated with equilibrium or steady-state problems. For example,

- Laplace's Equation,
$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0$$
- Poisson's Equation,
$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = f(x, y)$$

Parabolic Equations : The heat equation,

$$\frac{\partial U}{\partial t} = \kappa \frac{\partial^2 U}{\partial x^2}$$

Hyperbolic Equations : One-Dimensional Wave Equation,

$$\frac{\partial^2 U}{\partial t^2} = c^2 \frac{\partial^2 U}{\partial x^2}$$

14.1 Finite Difference Approximations to Derivatives

Now to approximate the derivatives using finite differences, we use the composite rules.

14.1.1 Functions of one-variable

Let $U : [a, b] \to \mathbb{R}$ be sufficiently differentiable function. Let

$$a = x_o < x_1 < x_2 < \cdots < x_n = b$$

be a partition of $[a, b]$ such that $x_n = x_0 + nh$, where $h = \frac{x_n - x_0}{n}$ is the **discretization parameter**.

Set $x_i = x_0 + ih$, $i = 0, 1, \ldots, n$ and $U_i = U(x_i)$.

Now, by Taylor's Theorem,

$$U(x+h) = U(x) + hU'(x) + \frac{h^2}{2}U''(x) + \frac{h^3}{6}U'''(x) + \ldots \tag{14.2}$$

$$U(x-h) = U(x) - hU'(x) + \frac{h^2}{2}U''(x) - \frac{h^3}{6}U'''(x) + \ldots \tag{14.3}$$

Now,

$$\begin{aligned}
\left.\frac{dU}{dx}\right|_{x=x_i} &= \frac{U(x_i+h)-U(x_i)}{h} + O(h) \quad \text{(From 14.2)} \\
&\approx \frac{U_{i+1}-U_i}{h} \quad \text{(Forward difference formula)} \\
&= \frac{U(x_i)-U(x_i-h)}{h} + O(h) \quad \text{(From 14.3)} \\
&\approx \frac{U_i-U_{i-1}}{h} \quad \text{(Backward difference formula)} \\
&= \frac{U(x_i+h)-U(x_i-h)}{2h} + O(h^2) \quad \text{(From Eq. 14.2 - 14.3)} \\
&\approx \frac{U_{i+1}-U_{i-1}}{2h} \quad \text{(Central difference formula)}
\end{aligned} \tag{14.4}$$

And for the second-derivative,

$$\begin{aligned}
\left.\frac{d^2U}{dx^2}\right|_{x=x_i} &= \frac{U(x_i+h)-2U(x_i)+U(x_i-h)}{h^2} + O(h^2) \quad \text{(From Eq. 14.2 + 14.3)} \\
&\approx \frac{U_{i+1}-2U_i+U_{i-1}}{h^2}
\end{aligned} \tag{14.5}$$

14.1.2 Functions of two-variables

Let $U : [0,a] \times [0,b] \to \mathbb{R}$ be a differentiable function of x and t. Introduce the **mesh parameters** h and k in the directions of x and t respectively. Denote,

$$\begin{aligned}
x_i &= ih\,, \;\; i = 0,1,2,\ldots,N \text{ with } x_0 = 0\,, \;\; x_N = a \\
t_j &= jk\,, \;\; i = 0,1,2,\ldots,J \text{ with } t_0 = 0\,, \;\; t_J = b
\end{aligned} \tag{14.6}$$

Before, the partial derivatives, let's first define the following,

$$\begin{aligned}
U_{i,j} &= U(x_i,t_j) = U(ih,jk) \\
U_{i+1,j} &= U(x_i+h,t_j) = U((i+1)h,jk) \\
U_{i,j+1} &= U(x_i,t_j+k) = U(ih,(j+1)k)
\end{aligned} \tag{14.7}$$

With the above, we can find the partial derivative as,

$$\begin{aligned}
\left.\frac{\partial U}{\partial x}\right|_{(x_i,t_j)} &= \frac{U_{i+1,j}-U_{i,j}}{h}+O(h)\\
&= \frac{U_{i,j}-U_{i-1,j}}{h}+O(h)\\
&= \frac{U_{i+1,j}-U_{i-1,j}}{2h}+O(h^2)\\
\left.\frac{\partial U}{\partial t}\right|_{(x_i,t_j)} &= \frac{U_{i,j+1}-U_{i,j}}{k}+O(k)\\
&= \frac{U_{i,j}-U_{i,j-1}}{k}+O(k)\\
&= \frac{U_{i,j+1}-U_{i,j-1}}{2k}+O(k^2)\\
\left.\frac{\partial^2 U}{\partial x^2}\right|_{(x_i,t_j)} &= \frac{U_{i+1,j}-2U_{i,j}+U_{i-1,j}}{h^2}+O(h^2)\\
\left.\frac{\partial^2 U}{\partial t^2}\right|_{(x_i,t_j)} &= \frac{U_{i,j+1}-2U_{i,j}+U_{i,j-1}}{k^2}+O(k^2)
\end{aligned} \tag{14.8}$$

We now use the above technique for solving some application based problems in the next few sections.

14.2 The Heat Equation

Consider one-dimensional heat equation of the form

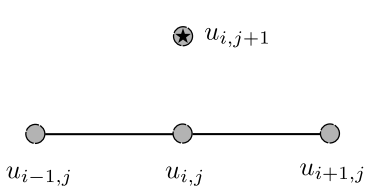
$$\begin{aligned}
\frac{\partial U}{\partial t} &= \frac{\partial^2 U}{\partial x^2} \quad x\in(0,1)\ ,\quad t>0\\
U(0,t) &= U(1,t)=0\ ,\quad t>0 \ \text{(Boundary Condition)}\\
U(x,0) &= f(x)\ ,\quad 0\le x\le 1 \ \text{(Initial Condition)}
\end{aligned} \tag{14.9}$$

Set $x_i = ih$, $i = 0,1,2,\ldots$ and $t_j = jk$, $j = 0,1,2,\ldots$.

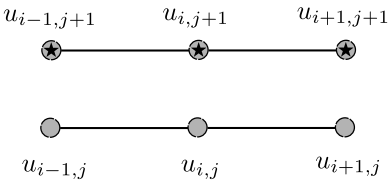
Let U_{ij} be the **true value of the solution** at the grid point (x_i, t_j).

Let u_{ij} be the **finite difference approximation to the true solution** at (x_i, t_j). Thus,

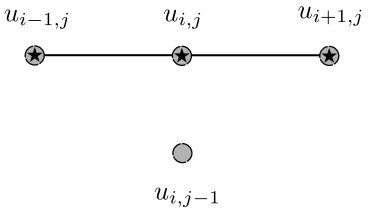
$$u_{ij} \approx U_{ij} = U(x_i, t_j)$$



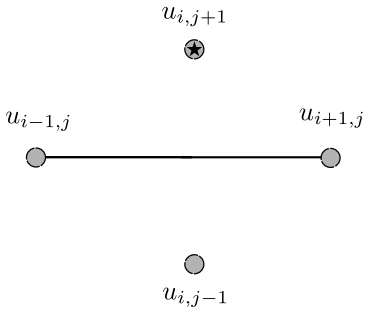
Schmidt's Scheme



Crank-Nicolson Scheme



Euler's Scheme



DuFort-Frankel Scheme

Figure 14.1: Computational Stencils for various schemes for Heat Equation.

with the basic notations in hand, we now explain various schemes by which one can approximate the partial derivative with the help of finite differences.

14.2.1 Schmidt's Explicit Scheme

Technique : Use the **forward-time and central-space (FTCS)** schemes to approximate $\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2}$ at the grid point (x_i, t_j).

Hence, we would get the partial derivatives as (from Eq. 14.8),

$$\begin{aligned} \left(\frac{\partial U}{\partial t}\right)_{(x_i,t_j)} &\approx \frac{u_{i,j+1} - u_{i,j}}{k} \\ \left(\frac{\partial^2 U}{\partial x^2}\right)_{(x_i,t_j)} &\approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} \end{aligned} \tag{14.10}$$

With the above approximations to the derivatives, the heat equation 14.9 would thus yield to the following,

$$\frac{u_{i,j+1} - u_{i,j}}{k} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$$

which on simplification gives,

$$\begin{aligned} u_{i,j+1} &= ru_{i-1,j} + (1-2r)u_{i,j} + ru_{i+1,j} \\ &\text{where,} \\ r &= \frac{k}{h^2} \end{aligned} \tag{14.11}$$

Note that in this equation, we want values of u at different values of t as the values of u at different values of x for a fixed t can be found by recursively using Eq. 14.11, the Boundary conditions and Initial conditions.

Remarks:

- It is a two level **explicit system** as the solution at the j^{th} level is only computed by values at the $j-1^{th}$ level.
- This scheme is **conditionally stable** when $\left(0 < r \leq \frac{1}{2}\right)$ (will be proved soon).
- The local truncation error is $O(h^2) + O(k)$ as seen from Eq. 14.9.

14.2.2 Euler's Implicit Scheme

Technique : Use the **backward-time and central-space (BTCS)** schemes at the point (x_i, t_j).

The approximations thus are,

$$\begin{aligned} \left(\frac{\partial U}{\partial t}\right)_{(x_i,t_j)} &\approx \frac{u_{i,j} - u_{i,j-1}}{k} \\ \left(\frac{\partial^2 U}{\partial x^2}\right)_{(x_i,t_j)} &\approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} \end{aligned} \tag{14.12}$$

Thus the heat equation is,

$$\frac{u_{i,j} - u_{i,j-1}}{k} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$$

and on simplification, it yields,

$$\begin{aligned} u_{i,j-1} &= -ru_{i-1,j} + (1+2r)u_{i,j} - ru_{i+1,j} \\ &\text{where,} \\ r &= \frac{k}{h^2} \end{aligned} \tag{14.13}$$

Remarks:

- It is a two-level **implicit scheme**.
- At each time level, we are required to solve a linear system.
- This scheme is **unconditionally stable**, that is, it's stable for all values of r.
- The local truncation error is $O(h^2) + O(k)$.

14.2.3 Crank-Nicolson Scheme

Technique : Use the central difference formula for both time and space at the mid point of on time scale.

$$\begin{aligned} \left(\frac{\partial U}{\partial t}\right)_{\left(x_i, t_{j+\frac{1}{2}}\right)} &\approx \frac{u_{i,j+1} - u_{i,j}}{k} \quad \text{(From Eq. 9.15)} \\ \left(\frac{\partial^2 U}{\partial x^2}\right)_{\left(x_i, t_{j+\frac{1}{2}}\right)} &\approx \frac{1}{2}\left[\frac{u_{i-1,j+1} - 2u_{i,j+1} + u_{i+1,j+1}}{h^2} + \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2}\right] \end{aligned} \tag{14.14}$$

Thus the Heat equation gives us,

$$\frac{u_{i,j+1} - u_{i,j}}{k} = \frac{1}{2}\left[\frac{u_{i-1,j+1} - 2u_{i,j+1} + u_{i+1,j+1}}{h^2} + \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2}\right]$$

Simplifying this finally gives us,

$$-ru_{i-1,j+1} + (2+2r)u_{i,j+1} - ru_{i+1,j+1} = ru_{i-1,j} + (2-2r)u_{i,j} + ru_{i+1,j} \tag{14.15}$$

where $r = \frac{k}{h^2}$.

Remarks:

- As visible, this is a two-level **implicit scheme** as the value at level $j+1$ and level j are interlinked in the different levels of i. In-short, the $j+1$ level is in multiple places with different levels of i.
- At each time level, we are required to solve a linear system.
- This scheme is also **unconditionally stable**.
- The local truncation error is $O(h^2) + O(k^2)$ (From Eq. 9.15).

14.2.4 Richardson's Scheme

Technique : Use only the central-time and central-space (CTCS) approximation.

The CTCS approximations are (follows from Eq. 9.15),

$$\begin{aligned} \left(\frac{\partial U}{\partial t}\right)_{(x_i,t_j)} &\approx \frac{u_{i,j+1} - u_{i,j-1}}{2k} \\ \left(\frac{\partial^2 U}{\partial x^2}\right)_{(x_i,t_j)} &\approx \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2} \end{aligned} \tag{14.16}$$

the heat equation thus gives us,

$$\frac{u_{i,j+1} - u_{i,j-1}}{2k} = \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2} \tag{14.17}$$

Remarks:

- It is a three level **explicit scheme** .
- This scheme is **unstable**, thus not used practically.
- Local truncation error is $O(h^2) + O(k^2)$.

14.2.5 DuFort-Frankel Explicit Scheme

Technique : Modification of Richardson's scheme as follows,

$$u_{i,j} = \frac{u_{i,j-1} + u_{i,j+1}}{2} \tag{14.18}$$

Thus, substituting in the Heat equation and further simplification yields,

$$u_{i,j+1} = \frac{1-2r}{1+2r} u_{i,j-1} + \frac{2r}{1+2r}(u_{i-1,j} + u_{i+1,j}) \tag{14.19}$$

Remarks:

- It is a three level **explicit scheme**.
- This scheme is **unconditionally stable**.
- The local truncation error is $O(h^2) + O(k^2)$.

14.2.6 Weighted Average Approximation

A more **general finite-difference approximation to the heat equation** is given by

$$\frac{u_{i,j+1} - u_{i,j}}{k} = \frac{1}{h^2}\left[\theta(u_{i-1,j+1} - 2u_{i,j+1} + u_{i+1,j+1}) + (1-\theta)(u_{i-1,j} - 2u_{i,j} + u_{i+1,j})\right] \tag{14.20}$$

where θ is a parameter such that $0 \leq \theta \leq 1$.

Now observe that it generalises the previous methods,

$$\begin{aligned} \theta = 0 &\implies \text{Scheme's Explicit Scheme} \\ \theta = 1 &\implies \text{Euler's Implicit Scheme} \\ \theta = \frac{1}{2} &\implies \text{Crank-Nicolson Scheme} \end{aligned} \tag{14.21}$$

Remarks:

- This scheme is **unconditionally stable** for $\frac{1}{2} \leq \theta \leq 1$.
- Also, this scheme is **conditionally stable** for $0 \leq \theta < \frac{1}{2}$. The condition for stability is,
$$r = \frac{k}{h^2} \leq \frac{1}{2(1-2\theta)}$$

Now, define the central-difference operators δ_x and δ_t as:

$$\begin{aligned} \delta_x \phi_{i,j} &= \phi_{i+\frac{1}{2},j} - \phi_{i-\frac{1}{2},j} \\ \delta_t \phi_{i,j} &= \phi_{i,j+\frac{1}{2}} - \phi_{i,j-\frac{1}{2}} \end{aligned} \tag{14.22}$$

Thus, we can write the explicit scheme for heat equation as,

$$\begin{aligned} \delta_t u_{i,j+\frac{1}{2}} &= u_{i,j+1} - u_{i,j} \\ \delta_x(\delta_x u_{i,j}) &= \delta_x(u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}) \\ &= u_{i+1,j} - 2u_{i,j} + u_{i-1,j} \end{aligned} \tag{14.23}$$

and finally,

$$\frac{1}{k}\delta_t u_{i,j+\frac{1}{2}} = \frac{1}{h^2}\delta_x^2 u_{i,j} \tag{14.24}$$

Note that using the definitions in Eq. 14.23, you can write the **Euler's Implicit scheme** as,

$$\frac{1}{k}\delta_t u_{i,j-\frac{1}{2}} = \frac{1}{h^2}\delta_x^2 u_{i,j} \tag{14.25}$$

the **Crank-Nicolson scheme**,

$$\frac{1}{k}\delta_t u_{i,j+\frac{1}{2}} = \frac{1}{2h^2}\left[\delta_x^2 u_{i,j+1} + \delta_x^2 u_{i,j}\right] \tag{14.26}$$

and finally, the **Weighted Average scheme**,

$$\frac{1}{k}\delta_t u_{i,j+\frac{1}{2}} = \frac{1}{h^2}\left[\theta\delta_x^2 u_{i,j+1} + (1-\theta)\delta_x^2 u_{i,j}\right] \tag{14.27}$$

14.3 Consistency & Stability of Finite Difference Equations

To discuss the stability of the schemes discussed earlier, we now formulate the definitions needed for calculating that.

Let

$$L(U) = 0 \tag{14.28}$$

represent the partial differential equation in the independent variables x and t with exact solution U. Let,

$$F_{i,j}(u) = 0 \tag{14.29}$$

represent the finite difference equation approximation for the PDE at $(i,j)^{th}$ mesh point with exact solution u.

The local truncation error $T_{i,j}(U)$ at the point (ih, jk) is defined by

$$T_{i,j}(U) = F_{i,j}(U) - L(U_{i,j}) = F_{i,j}(U)$$

as $L(U_{i,j}) = 0$. Thus, $T_{i,j}$ gives an indication of the error resulting from the replacement of $L(U_{i,j})$ by $F_{i,j}(U)$. We hence arrive at the following definition,

Definition : *If $T_{i,j}(U) \to 0$ as $h \to 0$ and $k \to 0$, then the difference equation in Eq. 14.29 is said to be* ***consistent*** *or* ***compartible*** *with the PDE Eq. 14.28.*

Example. To illustrate the use of this definition, let us, for example, consider the local truncation error $T_{i,j}$ of the two level explicit scheme approximating $U_t = U_{xx}$ at the point (ih, jk).

The finite difference equation in this case will be (Eq. 14.11),

$$F_{i,j}(u) = \frac{u_{i,j+1} - u_{i,j}}{k} - \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2} = 0$$

thus the Truncation error $T_{i,j}$,

$$T_{i,j}(U) = F_{i,j}(U) = \frac{U_{i,j+1} - U_{i,j}}{k} - \frac{U_{i-1,j} - 2U_{i,j} + U_{i+1,j}}{h^2} \tag{14.30}$$

By Taylor's Expansion,

$$\begin{aligned}
U_{i+1,j} &= U((i+1)h, j) = U(x_i + h, t_j) \\
&= U_{i,j} + h\left(\frac{\partial U}{\partial x}\right)_{i,j} + \frac{h^2}{2}\left(\frac{\partial^2 U}{\partial x^2}\right)_{i,j} + \frac{h^3}{3!}\left(\frac{\partial^3 U}{\partial x^3}\right)_{i,j} + \dots \\
U_{i-1,j} &= U(x_i - h, t_j) \\
&= U_{i,j} - h\left(\frac{\partial U}{\partial x}\right)_{i,j} + \frac{h^2}{2}\left(\frac{\partial^2 U}{\partial x^2}\right)_{i,j} - \frac{h^3}{3!}\left(\frac{\partial^3 U}{\partial x^3}\right)_{i,j} + \dots \\
U_{i,j+1} &= U(x_i, t_j + k) \\
&= U_{i,j} + k\left(\frac{\partial U}{\partial t}\right)_{i,j} + \frac{k^2}{2}\left(\frac{\partial^2 U}{\partial t^2}\right)_{i,j} + \frac{k^3}{3!}\left(\frac{\partial^3 U}{\partial t^3}\right)_{i,j} + \dots
\end{aligned} \tag{14.31}$$

Substituting these approximations in the Finite Difference Eq. 14.30, we get,

$$T_{i,j} = \left(\frac{\partial U}{\partial t} - \frac{\partial^2 U}{\partial x^2}\right)_{i,j} + \frac{1}{2}k\left(\frac{\partial^2 U}{\partial t^2}\right)_{i,j} - \frac{1}{12}h^2\left(\frac{\partial^4 U}{\partial x^4}\right)_{i,j} + \frac{1}{6}k^2\left(\frac{\partial^3 U}{\partial t^3}\right)_{i,j} - \frac{1}{360}h^4\left(\frac{\partial^6 U}{\partial x^6}\right)_{i,j} + \dots$$

Note that since $\left(\frac{\partial U}{\partial t} - \frac{\partial^2 U}{\partial x^2}\right)_{i,j} = 0$, the major contribution to the local truncation error happen only through the following term,

$$\frac{1}{2}k\left(\frac{\partial^2 U}{\partial t^2}\right)_{i,j} - \frac{1}{12}h^2\left(\frac{\partial^4 U}{\partial x^4}\right)_{i,j}$$

Therefore, $T_{i,j} = O(k) + O(h^2)$. Also note that,

$$\boxed{T_{i,j} \to 0 \text{ as } h \to 0 \text{ and } k \to 0}$$

Thus, the explicit scheme approximating the heat equation in Eq. 14.11 is **consistent** with the heat equation.

14.4 Stability

Definition : *Stability is related to the round-off error. We say that the scheme is **stable** if the round-off error in the numerical process is bounded.*

Let $R = \{(x,t)|0 \leq x \leq 1\ ,\ \ 0 \leq t \leq T\}$ be a rectangle. Consider the PDE,

$$L(U) = 0 \text{ in } R$$

with prescribed initial and boundary conditions.

Let h and k be the mesh parameters such that,

$$\begin{aligned} x_i &= ih\ ,\ \ i = 0, 1, \ldots, N \text{ with } Nh = 1 \\ t_j &= jk\ ,\ \ j = 0, 1, \ldots, N \text{ with } Jk = T \end{aligned}$$

Assume that h is related to k as follows: As $h \to 0$, then $k \to 0$ (or $k = O(h)$ or $k = O(h^2)$).

Consider the **general finite difference approximation of the form**:

$$b_{i-1}u_{i-1,j+1} + b_i u_{i,j+1} + b_{i+1}u_{i+1,j+1} = c_{i-1}u_{i-1,j} + c_i u_{i,j} + c_{i+1}u_{i+1,j} \tag{14.32}$$

where b_i's and c_i's are constants.

Now, **suppose that the boundary values $u_{0,j}$ and $u_{N,j}$ for $j > 0$ are known**.

Then for remaining $i = 1, \ldots, N-1$, we can write,

$$\begin{bmatrix} b_1 & b_2 & & & & \\ b_1 & b_2 & b_3 & & & \\ \ddots & \ddots & \ddots & & & \\ & & b_{N-3} & b_{N-2} & b_{N-1} \\ & & & b_{N-2} & b_{N-1} \end{bmatrix} \begin{bmatrix} u_{1,j+1} \\ u_{2,j+1} \\ \vdots \\ u_{N-2,j+1} \\ u_{N-1,j+1} \end{bmatrix}$$

$$= \begin{bmatrix} c_1 & c_2 & & & & \\ c_1 & c_2 & c_3 & & & \\ \ddots & \ddots & \ddots & & & \\ & & c_{N-3} & c_{N-2} & c_{N-1} \\ & & & c_{N-2} & c_{N-1} \end{bmatrix} \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{N-2,j} \\ u_{N-1,j} \end{bmatrix} + \begin{bmatrix} c_0 u_{0,j} - b_0 u_{0,j+1} \\ 0 \\ \vdots \\ 0 \\ c_N u_{N,j} - b_N u_{N,j+1} \end{bmatrix} \tag{14.33}$$

In matrix notation,

$$\begin{aligned} \mathbf{B}\boldsymbol{u}_{j+1} &= \mathbf{C}\boldsymbol{u}_j + \boldsymbol{d}_j \\ \implies \boldsymbol{u}_{j+1} &= \mathbf{B}^{-1}\mathbf{C}\boldsymbol{u}_j + \mathbf{B}^{-1}\boldsymbol{d}_j \\ \implies \boldsymbol{u}_{j+1} &= \mathbf{A}\boldsymbol{u}_j + \boldsymbol{f}_j \end{aligned} \tag{14.34}$$

where $\mathbf{A} = \mathbf{B}^{-1}\mathbf{C}$ and $\boldsymbol{f}_j = \mathbf{B}^{-1}\boldsymbol{d}_j$.

We can apply the Eq. 14.34 recursively to obtain

$$\begin{aligned} \boldsymbol{u}_j &= \mathbf{A}\boldsymbol{u}_{j-1} + \boldsymbol{f}_{j-1} \\ &= \mathbf{A}(\mathbf{A}\boldsymbol{u}_{j-2} + \boldsymbol{f}_{j-2}) + \boldsymbol{f}_{j-1} = \mathbf{A}^2\boldsymbol{u}_{j-2} + \mathbf{A}\boldsymbol{f}_{j-2} + \boldsymbol{f}_{j-1} \\ &= \ldots \\ &= \mathbf{A}^j\boldsymbol{u}_0 + \mathbf{A}^{j-1}\boldsymbol{f}_0 + \mathbf{A}^{j-2}\boldsymbol{f}_1 + \cdots + \boldsymbol{f}_{j-1} \end{aligned} \tag{14.35}$$

$\boxed{\star}$ where $\boldsymbol{u}_0$ is the **vector of initial values** and $\boldsymbol{f}_0, \boldsymbol{f}_1, \ldots, \boldsymbol{f}_{j-1}$ are the **vectors of known boundary values**.

Since we want to find the stability of the finite difference method, so we check the error in the resultant vector after **perturbing the initial value from $\boldsymbol{u}_0$ to $\boldsymbol{u}_0^*$**. Thus, the exact solution at the j^{th} time level will be,

$$\boldsymbol{u}_j^* = \mathbf{A}^j\boldsymbol{u}_0^* + \mathbf{A}^{j-1}\boldsymbol{f}_0 + \mathbf{A}^{j-2}\boldsymbol{f}_1 + \cdots + \boldsymbol{f}_{j-1} \tag{14.36}$$

Now, define the **perturbation error** by

$$\boldsymbol{e}^* = \boldsymbol{u}^* - \boldsymbol{u}$$

then it follows from Eq. 14.35 and 14.36 that,

$$\boldsymbol{e}_j = \boldsymbol{u}_j^* - \boldsymbol{u}_j = \mathbf{A}(\boldsymbol{u}_0^* - \boldsymbol{u}_0) = \mathbf{A}^j \boldsymbol{e}_0$$

where $\boldsymbol{e}_0$ is the amount by which the initial value $\boldsymbol{u}_0$ is perturbed by.

Note that,

$$\|\boldsymbol{e}_j\| = \|\mathbf{A}^j \boldsymbol{e}_0\| \leq \|\mathbf{A}^j\| \|\boldsymbol{e}_0\| \tag{14.37}$$

Now if there exists a positive number M, independent of j, h and k such that $\|\mathbf{A}^j\| \leq M$, then it's easy to see that,

$$\|\boldsymbol{e}_j\| \leq M \|\boldsymbol{e}_0\|$$

Now to use this, first note that,

$$\|\mathbf{A}^j\| = \|\mathbf{A}^{j-1}\mathbf{A}\| \leq \|\mathbf{A}^{j-1}\| \|\mathbf{A}\| \leq \cdots \leq \|\mathbf{A}\|^j$$

Thus, the bound M on $\|\mathbf{A}^j\|$ would imply that,

$$\|\boldsymbol{e}_j\| \leq \|\mathbf{A}\|^j \|\boldsymbol{e}_0\|$$

The **Lax & Richtmeyer definition of stability** thus requires that

$$\boxed{\|\mathbf{A}\| \leq 1} \tag{14.38}$$

This is the **necessary and sufficient condition** for the difference equations to be **stable** when the solution of the PDE does not increase as t increases.

14.5 Stability Analysis by Matrix Method

Now that we have converted our problem to equivalent matrix formulation, we can now use a lot of techniques for checking convergence and stability available from Matrix Theory!

In particular, we would explore stability of finite difference equations by two methods, namely **Matrix method** and **von Neumann's**[1] **method**.

Before that, we present brief review of matrix theory.

[1] *Of-course!*

14.5.1 Basic Elements of Matrix Theory

- Let $\mathbf{A}$ be an $n \times n$ matrix. Define,

$$\begin{aligned} \|\mathbf{A}\|_1 &= \max_{1\le j\le n} \sum_{i=1}^{n} |a_{ij}| \\ \|\mathbf{A}\|_\infty &= \max_{1\le i\le n} \sum_{j=1}^{n} |a_{ij}| \\ \|\mathbf{A}\| &= \max_{x\neq 0} \frac{\|\mathbf{A}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \max_{\|x\|=1} \|\mathbf{A}\boldsymbol{x}\| \end{aligned} \tag{14.39}$$

- Let $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ be the eigenvectors corresponding to the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ of $\mathbf{A}$. Then,

$$\begin{aligned} & \mathbf{A}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i \\ \implies \quad & \|\mathbf{A}\boldsymbol{v}_i\| = \|\lambda_i \boldsymbol{v}_i\| = |\lambda_i| \|\boldsymbol{v}_i\| \\ \implies \quad & |\lambda_i| \|\boldsymbol{v}_i\| = \|\mathbf{A}\boldsymbol{v}_i\| \le \|\mathbf{A}\| \|\boldsymbol{v}_i\| \\ \implies \quad & |\lambda_i| \le \|\mathbf{A}\| \ , \quad i = 1, \ldots, N \\ \implies \quad & \max_i |\lambda_i| \le \|\mathbf{A}\| \\ & \text{or, } \ \rho(\mathbf{A}) \le \|\mathbf{A}\| \end{aligned} \tag{14.40}$$

  where $\rho(\mathbf{A})$ is called the **spectral radius** and is the largest eigenvalue of matrix $\mathbf{A}$.

- The eigenvalues of the following $n \times n$ matrix

$$\begin{bmatrix} a & b & & & \\ c & a & b & & \\ \ddots & \ddots & \ddots & & \\ & & c & a & b \\ & & & a & b \end{bmatrix}$$

  are given by,

$$\lambda_s = a + 2\sqrt{bc} \cos\left(\frac{s\pi}{n+1}\right) \ , \quad s = 1, \ldots, n \tag{14.41}$$

  where $a, b, c \in \mathbb{R}$.

Lax-Richtmeyer Stability condition

As in Eq. 14.38, the Lax-Richtemeyer Condition that $\|\mathbf{A}\| \leq 1$ implies that,

$$\boxed{\|\mathbf{A}\| \leq 1 \implies \rho(\mathbf{A}) \leq 1} \tag{14.42}$$

Note that the converse is not true, that is,

$$\rho(\mathbf{A}) \leq 1 \not\implies \|\mathbf{A}\| \leq 1$$

However, if $\mathbf{A}$ is real and symmetric, then

$$\|\mathbf{A}\| = \sqrt{\rho(\mathbf{A}^T\mathbf{A})} = \sqrt{\rho(\mathbf{A}^2)} = \sqrt{\rho^2(\mathbf{A})} = \rho(\mathbf{A}) \tag{14.43}$$

Stability of the Schmidt's Explicit Scheme

$$u_{i,j+1} = ru_{i-1,j} + (1-2r)u_{i,j} + ru_{i+1,j}\,, \quad i = 1, \ldots, N-1$$

As usual, assuming that the boundary values $u_{0,j}$ and $u_{N,j}$ are known for $j = 1, 2, \ldots$, thus the equivalent matrix formulation becomes,

$$\begin{bmatrix} u_{1,j+1} \\ u_{2,j+1} \\ \vdots \\ u_{N-2,j+1} \\ u_{N-1,j+1} \end{bmatrix} = \underbrace{\begin{bmatrix} (1-2r) & r & & & \\ r & (1-2r) & r & & \\ \ddots & \ddots & \ddots & & \\ & & r & (1-2r) & r \\ & & & r & (1-2r) \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{N-2,j} \\ u_{N-1,j} \end{bmatrix} + \begin{bmatrix} ru_{0,j} \\ 0 \\ \vdots \\ 0 \\ ru_{N,j} \end{bmatrix}$$

Clearly, the amplification matrix is $\mathbf{A}$.

Now, note that

$$\|\mathbf{A}\|_\infty = 2|r| + |1-2r|$$

Now,

CASE I : $1 - 2r \geq 0 \implies 0 \leq r \leq \frac{1}{2}$.

Note that,

$$\|\mathbf{A}\|_\infty = r + 1 - 2r + r = 1$$

CASE II : $1 - 2r < 0 \implies r > \frac{1}{2}$.

Therefore, $|1-2r| = 2r - 1$, and

$$\|\mathbf{A}\|_\infty = r + 2r - 1 + r = 4r - 1 > 1$$

Thus, the scheme is stable only for $0 < r \leq \frac{1}{2}$. Note that since $\mathbf{A}$ is real and symmetric,

$$\|\mathbf{A}\| = \rho(\mathbf{A}) = \max_s |\mu_s|$$

where μ_s is the s^{th} eigenvalue of $\mathbf{A}$.

We can do the same calculation in a more *eigenvalue oriented* one, i.e., deriving a general form of eigenvalue and using Lax-Richtemeyer condition of stability in Eq. 14.42.

For this, first note that the matrix $\mathbf{A}$ can be rewritten as,

$$\mathbf{A} = \mathbf{I}_{N-1} + r\mathbf{T}_{N-1}$$

where,

$$\begin{aligned} \mathbf{I}_{N-1} &= \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \\ \mathbf{T}_{N-1} &= \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & & \ddots & & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix} \end{aligned} \tag{14.44}$$

Now, a particular eigenvalue of $\mathbf{A}$ is just the sum of the corresponding eigenvalues of $\mathbf{I}_{N-1}$ and $\mathbf{T}_{N-1}$. Thus, we first need to find the eigenvalues λ_s of $\mathbf{T}_{N-1}$ as the eigenvalues of $\mathbf{I}_{N-1}$ are trivially all 1.

$$\begin{aligned} \lambda_s &= -2 + 2\cos\left(\frac{s\pi}{N}\right) \quad \text{(Eq. 14.41)} \\ &= -4\sin^2\left(\frac{s\pi}{2N}\right) \end{aligned}$$

for $s = 1, \ldots, N-1$. Therefore, the eigenvalues of $\mathbf{A}$ are,

$$\mu_s = 1 - 4r\sin^2\left(\frac{s\pi}{2N}\right) \tag{14.45}$$

Now, for the stability, $\rho(|\mu_s|) \leq 1$,

$$\rho(|\mu_s|) = \max_s \left|1 - 4r\sin^2\left(\frac{s\pi}{2N}\right)\right| \leq 1 \tag{14.46}$$

which leads to,

$$\begin{aligned} -1 \leq 1 - 4r\sin^2\left(\frac{s\pi}{2N}\right) \leq 1 \\ \implies r \leq \frac{1}{2\sin^2\left(\frac{s\pi}{2N}\right)} \end{aligned}$$

which implies that (for $s = N-1$),

$$r \leq \frac{1}{2\sin^2\left(\frac{(N-1)\pi}{2N}\right)} \tag{14.47}$$

Hence, for $h \to 0$, then $N \to \infty$, which then implies that $\sin^2\left(\frac{(N-1)\pi}{2N}\right) \to 1$. Which implies that,

$$r \leq \frac{1}{2} \tag{14.48}$$

is the required condition for stability (conditionally).

14.6 Stability Analysis by von Neumann Method & Lax Equivalence Theorem

We previously saw the basic definition of stability by Lax-Richtmeyer condition. Now, we shift our focus on extending this criterion for checking the stability of any given finite difference approximation of a certain IVP through a method called *von Neumann stability method*.

Consider the IVP,

$$\begin{aligned} U_t = U_{xx}\,, \;\; 0 \leq x \leq L\,, \;\; 0 < t \leq T \\ U(x,0) = f(x)\,, \;\; 0 \leq x \leq L \end{aligned} \tag{14.49}$$

The basic idea now followed to find an approximate solution to this is two-fold:

- ⋆ Express initial data by finite Fourier series along the initial mesh-point at $t = 0$.
- ⋆ Study the growth of a function that reduced to this series at $t = 0$ by variable separation method.

Hence, the Fourier series in complex exponential form can be represented as,

$$U(x,0) = f(x) = \sum A_n e^{\frac{i\pi n x}{L}}$$

Now, let us introduce the commonly used notations for this method before moving into it's peculiarities,

Define,

$$\begin{aligned} x_p &= ph\ , \quad p = 0,1,2,\ldots,N \text{ with } Nh = L \\ t_q &= qk\ , \quad q = 0,1,2,\ldots,J \text{ with } Jk = T \\ u(x_p,t_q) &= u_{p,q} = u(ph,qk) \end{aligned} \tag{14.50}$$

With the above, first note that the Fourier Series terms at particular values of x_p are

$$A_n e^{\frac{i\pi n x_p}{L}} = A_n e^{\frac{i\pi n ph}{Nh}} = A_n e^{i\beta_n ph} \text{ where } \beta_n = \frac{\pi n}{Nh}$$

Now, at $q = 0$, i.e. $t = t_0 = 0$,

$$u_{p,0} = u(ph,0) = \sum_{n=0}^{N} A_n e^{i\beta_n ph}$$

Using the given $u_{p,0}$ at $p = 0,1,2,\ldots,N$, **the constants A_n can be uniquely determined**! But since this problem is a linear one, thus it is enough to investigate the propagation of only one initial value, such as $e^{i\beta ph}$ as separate solutions are additive. The coefficient A_n is a constant and can be neglected for the growth analysis. Thus, we study,

$$u_{p,q} = e^{i\beta ph} \cdot e^{\alpha qk} = e^{i\beta ph} \cdot \xi^q \tag{14.51}$$

where $\boxed{\xi = e^{\alpha k}}$ and α is a constant. Note that this turns to $e^{i\beta ph}$ at $k = 0$ which corresponds to the $t = 0$ criterion.

Now, **by Lax-Richtemeyer definition of stability**, we have,

$$\left|u_{p,q}\right| \leq C \ \ \forall q \leq J$$

as $h \to 0$, $k \to 0$ and for all values of β needed to satisfy the initial conditions. This inequality directly translate to,

$$\begin{aligned} \left|u_{p,q}\right| &\leq C \\ \left|e^{i\beta ph} \cdot \xi^p\right| &\leq C \quad \text{(Eq. 14.51)} \\ \left|e^{i\beta ph}\right| |\xi^p| &\leq C \\ |\xi^p| &\leq C \end{aligned} \tag{14.52}$$

i.e. $|\xi^p|$ should remain bounded, which is only possible if

$$\boxed{\xi \leq 1} \tag{14.53}$$

which is the **necessary and sufficient condition for stability**, or also called the **von Neumann stability** condition.

Remarks:

- **Consistency** is related to the truncation error. We say a finite difference scheme is consistent with given PDE if the truncation error $T_{i,j} \to 0$ as $h \to 0$ and $k \to 0$.
- Whereas **Stability** is related to the round-off error. We say that a finite difference scheme is stable if the exact solution of the finite difference equation does not grow with time.

We finally note the major theme of the whole discussion. Since from the beginning, we wanted to find an approximation as *close* as possible to the true solution of an PDE, we hence discuss the *convergence* of the finite difference approximation to the true solution, starting from the following definition.

Definition : *Let $U_{i,j}$ be the exact solution of the PDE at the mesh point (ih, jk) and let $u_{i,j}$ be it's finite difference approximation at the same point. Then, we say that $u_{i,j}$* ***converges*** *to $U_{i,j}$ if*

$$\left|u_{i,j} - U_{i,j}\right| \to 0 \;\; \textit{as } h \to 0 \textit{ and } k \to 0$$

In the final note, we arrive at the following theorem.

Theorem 14.16 : (**Lax-Equivalence Theorem**) *Given a well-posed, linear initial value problem and a* ***consistent*** *linear finite difference equation approximating it. Then,* ***stability is the necessary & sufficient condition for convergence****. That is, for a consistent finite difference scheme,*

$$\textit{Consistency} + \textit{Stability} \implies \textit{Convergence}$$