NUMERICAL ANALYSIS

Presentation by

Kum K Shalini
Lecturer in Mathematics
SKR & SKR College for women, kadapa.

Course Brief Description

This course is an introduction to basic and classical numerical algorithms. We will describe numerical algorithms for floating point computation, rootfinding, solving linear systems, interpolation and quadrature. We will also discuss the underlying mathematical principles and theories of these numerical methods and their implementations.

PENNSTATE

What is Numerical Analysis?

"It is the study of algorithms that use numerical approximation for the problem of mathematical analysis"

---- *Wikipedia*

- *study of algorithms:*
  Create, analyze, and implement algorithms
- *numerical approximation:*
  Solving problems numerically and approximately
- *mathematical analysis:*
  Problems of continuous mathematics. Such problems originate generally from real-world applications of algebra, geometry and calculus. and they involve variables which vary continuously.

PENNSTATE

What Is Numerical computing ?

- Numerical Computing is an approach for solving complex mathematical problems using only simple arithmetic operations.
- The approach involves formulation of model of physical situations that can be solved with arithmetic operations.
- Steps involved
  1. Formulation of mathematical model.
  2. Construction of an appropriate numerical method.
  3. Implementation of the method to obtain a solution.
  4. Validation of the solution.

Unlike Linear equations in which f(x) is directly dependent upon (x) ; in Non Linear equation f(x) is not in exact proportion to the changes of variable (x). Ex amples

$f(x) = x^2 + 2x + 3$

$f(x) = x^3 + 7$

Method of solution

There are a number of method to solve a Non linear equation.

1. Direct Analytical method
2. Graphical method
3. Trial and Error method
4. Iterative Method

Iterative Methods

Iterative method being there in the process of one or more guesses at the solution being sought. Iterative method based on number of guesses they use can be grouped into two categories:

1. Bracketing method

Bisection Method

Regula Falsi Method

2.Open end method

Newton Raphson Method

Secant Method

Fixed Point Method

1. Bisection Method

1. Declare the function f(x)=0
   //For finding initial guesses
2. For i=0 Compute f(i) & f(i+1)
   if f(i)*f(i+1) <0
   then
   break the loop
   else
   Continue by incrementing (i) by 1.
   //setting initial roots
3. x0= i ; x1= i+1
4. Enter the number of iteration , n

//calculation of next root

5. Compute x2 by taking Arithmetic Mean of x0 and x1.

   x2=(x0+x1)/2

6. Compute f(x2)

// checking the roots and interchanging the values

7. If f(x0)*f(x2) <0 then

x1=x2

else

x0=x2

8. n=n-1

9.If n >0 Goto step 5

10. The root of equation is x2

11. Exit

Additional information

- In this method we choose a mid point x2 in between x0 and x1. Depending upon sign of f(x0), f(x1), f(x2); x1 or x0 is replaced by x2. such that root lie between the new interval.
- In this the interval of root reduced by a factor of 2 every time
- So after n step the interval reduced to $(x1-x0)/2^n = \Delta x/2^n$

  After n iteration root must lie between $\pm \Delta x/2^n$ of our estimate. This means that the error bound at the nth iteration is

$$E_n = \left\| \Delta x/2^n \right\|$$

Similarly

$$E_{n+1} = \left\| \Delta x/2^{n+1} \right\|$$

Thus the bisection method is linearly convergent

2. Regula Falsi method

1. Declare the function f(x)=0

   //For finding initial guesses

2. For i=0 Compute f(i) & f(i+1)

   if f(i)*f(i+1) <0

   then

   break the loop

   else

   Continue by incrementing (i) by 1.

   //setting initial roots

3. x0= i ; x1= i+1

4. Compute f(x0) and f(x1) by putting x0 and x1 in the equation.

//calculation of next root

5.Compute x2 by using formula

$$x2 = \frac{x0 * f(x1) - x1 * f(x0)}{f(x1) - f(x0)}$$

6. Compute f(x2)

// checking the roots and interchanging the values

7. If f(x0)*f(x2) < 0 then

x1=x2

else

x0=x2

8. If |x1-x0| > 0.0005 Goto step 4

9. The root of equation is x2

11. Exit

3. Newton- Raphson Method

1. Declare the function f(x)=0

   // computing Differentiation

2. Compute the differentiation of the function

   Let it be df(x)=0

   //For finding initial guesses

3. if f(i)*f(i+1) <0 break the loop

   else Continue by incrementing (i) by 1.

   //setting initial roots

4. x0= i ; x1= i+1

5. Take a value either x0 or x1 to proceed further

6. Compute f(x0) and df(x0)

//calculation of next root

5.Compute x1by using formula

$$x1 = x0 - \frac{f(x0)}{df(x0)}$$

6. Compute f(x1)

// checking the roots and interchanging the values

7. If $|x1-x0| > 0.0005$ then x0 = x1 ;Goto step 4

8. The root of equation is x0

10. Exit

4. Secant Method

1. Declare the function f(x)=0

   //For finding initial guesses

2. For i=0 Compute f(i) & f(i+1)

   if f(i)*f(i+1) <0

   then

   break the loop

   else

   Continue by incrementing (i) by 1.

   //setting initial roots

3. x0= i ; x1= i+1

4. Compute f(x0) and f(x1) by putting x0 and x1 in the equation.

//calculation of next root

5.Compute x2 by using formula

$$x2 = x1 - f(x1) * \left\{ \frac{x1 - x0}{f(x1) - f(x0)} \right\}$$

6. Compute f(x2)

// checking the roots and interchanging the values

7. If f(x0)*f(x2) < 0 then

x1=x2

else

x0=x2

8. If |x1-x0| > 0.0005 Goto step 4

9. The root of equation is x2

11. Exit

5. Fixed Point Method

- In Numerical analysis, **Fixed point iteration** is a method of computing fixed points of iterated functions.
- More specifically, given a function **f(x)** defined on the real numbers with real values and given a point **x0** in the domain of **f(x),** the fixed point iteration is
- which gives rise to the sequence which is hoped to converge to a point **x**. If **f(x)** is continuous, then one can prove that the obtained **x** is a fixed point of **f(x)**, i.e.,

$$f(x) = x.$$

Example :-Babylonian method

Babylonian method (for finding square root)

1. Begin with an arbitrary positive starting value x_0 (the closer to the root, the better).
2. Let x_{n+1} be the average of x_n and S / x_n (using the Arithmetic Mean to approximate the Geometric Mean).

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{S}{x_n}\right),$$

3. Repeat step 2 until the desired accuracy is achieved.

$$\mathcal{X}_0 = \sqrt{S}$$

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{S}{x_n}\right),$$

$$\sqrt{S} = \lim_{n \to \infty} x_n.$$

Interpolation Techniques

- In the mathematical subfield of Numerical analysis, interpolation is a method of constructing new data points within the range of a **discrete set** of known data points.
- For example, suppose we have a table like this, which gives some values of an unknown function *f(x)*.

x	*f*(*x*)
0	0
1	0.8415
2	0.9093
3	0.1411
4	-0.7568
5	−0.9589

Interpolation provides a means of estimating the function at intermediate points, such as $x = 2.5$

1. Lagrange Interpolation

- Let y=f(x) be a function which takes values (x0,y0),(x1,y1),_ _ , (xn, yn). Since there are (n+1) pairs of values of (x) and (y).
- Lagrange's Formula state that

$$Y= \frac{(x-x1)(x-x2)\ldots\ldots.(x-xn)}{(x0-x1)(x0-x2)\ldots..(x0-xn)} * y0 + \frac{(x-x0)(x-x2)\ldots\ldots.(x-xn)}{(x1-x2)(x0-x2)\ldots..(x0-xn)} * y1$$

$$+ \ldots. \ldots\ldots + \frac{(x-x0)(x-x1)(x-x2)\ldots\ldots.(x-xn-1)}{(xn-x0)(xn-x1)\ldots..(xn-xn-1)} * yn$$

1. Read the Number of values to be entered be (n)
2. Read the value of X for which the f(x) is to be find.
3. Start a loop from 0 to n store value of (x) and f(x) is array of length n . x[n], y[n]
4. Declare a variable Sum=0 .
5. Start a loop from (i= 0 to n-1)
6. Product =1
7. Start another loop from (j=0 to n-1)
8. If (i ≠ j) then

   product= product * (x- x[j]) / (x[i] - x[j])

   end j loop
9. Sum = sum + y[i] * product

   end i loop
10. f(x)= sum
11. Exit

- We wish to interpolate $f(x) = x^3$ over the range $1 \le x \le 3$, given these 3 points:

  The interpolating polynomial is:

  x0 = 1 f(x0) = 1

  x1 = 2 f(x2) = 8

  x3 = 3 f(x3) =27

  x4= 4 f(x4) =64

  x(5)= ?

2.Newton Interpolation method

- There are two types of newtons Interpolation Methods

1 . **Newton's forward method**

It is applicable when the asked x value does not lie between last given x interval

2. **Newton's backward method**

It is applicable when the asked x value lie between last given x interval

Newton's forward method

1. Read the Number of values to be entered be (n)
2. Read the value of X for which the f(x) is to be find.
3. Start a loop from 0 to n store value of (x) and f(x) is array of length n . x[n], y[n]
4. Compute h= x[n+1] – x[n]
5. **Compute** $u = \frac{x - x[0]}{h}$ **// step differ in both method**
6. Declare a two Dimensional array.
7. Compute the Difference table

	Y 0	ΔY 1	ΔY 2	ΔY 3	....	ΔY n
0	Y[0,0]	Δ Y1(0,1)= Y[1,0]-Y[0,0]	Δ Y2(0,2)= Δ Y1(1,1) -Δ Y1(0,1)	Δ Y3(0,3)= Δ Y2(2,1) -Δ Y1(0,2)		n-1
1	Y[1,0]	Δ Y1(1,1)= Y[2,0]-Y[1,0]	Δ Y2(2,1)= Δ Y1(2,1) -Δ Y1(1,1)			
2	Y[2,0]	Δ Y1(2,1)= Y[3,0]-Y[2,0]				
3	Y[3,0]					

8. Start a loop from 0 to n-1 , save value of first row of previous array in that name it Ynew[].

9. Start a loop from (i= 0 to n-1)

declare product = 1

10. Start a loop from (j= 0 to i)

$$prod = prod * \frac{u - j}{j+1}$$

end j loop

11. Value = prod* Ynew[(i+1)]

sum=sum+value

end I loop

12. The value of function at given x ie f(x) is sum.

Newton's Backward method

1. Read the Number of values to be entered be (n)
2. Read the value of X for which the f(x) is to be find.
3. Start a loop from 0 to n store value of (x) and f(x) is array of length n . x[n], y[n]
4. Compute h= x[n+1] – x[n]
5. **Compute** $u = \frac{x - x[n]}{h}$ **// step differ in both method**
6. Declare a two Dimensional array.
7. Compute the Difference table

	Y 0	ΔY 1	ΔY 2	ΔY 3	....	ΔY n
0	Y[0,0]	Δ Y1(0,1)= Y[1,0]-Y[0,0]	Δ Y2(0,2)= Δ Y1(1,1) -Δ Y1(0,1)	Δ Y3(0,3)= Δ Y2(2,1) -Δ Y1(0,2)		n-1
1	Y[1,0]	Δ Y1(1,1)= Y[2,0]-Y[1,0]	Δ Y2(2,1)= Δ Y1(2,1) -Δ Y1(1,1)			
2	Y[2,0]	Δ Y1(2,1)= Y[3,0]-Y[2,0]				
3	Y[3,0]					

8. Start a loop from (i=0 to n-1) ,

Start a loop from (j=n to 0) ,

assign Ynew[i] = difference[n][i]

end loop j

End loop I

9. Start a loop from (i= 0 to n-1)

declare product = 1

10. Start a loop from (j= 0 to i)

$$prod = prod * \frac{u - j}{j+1}$$

end j loop

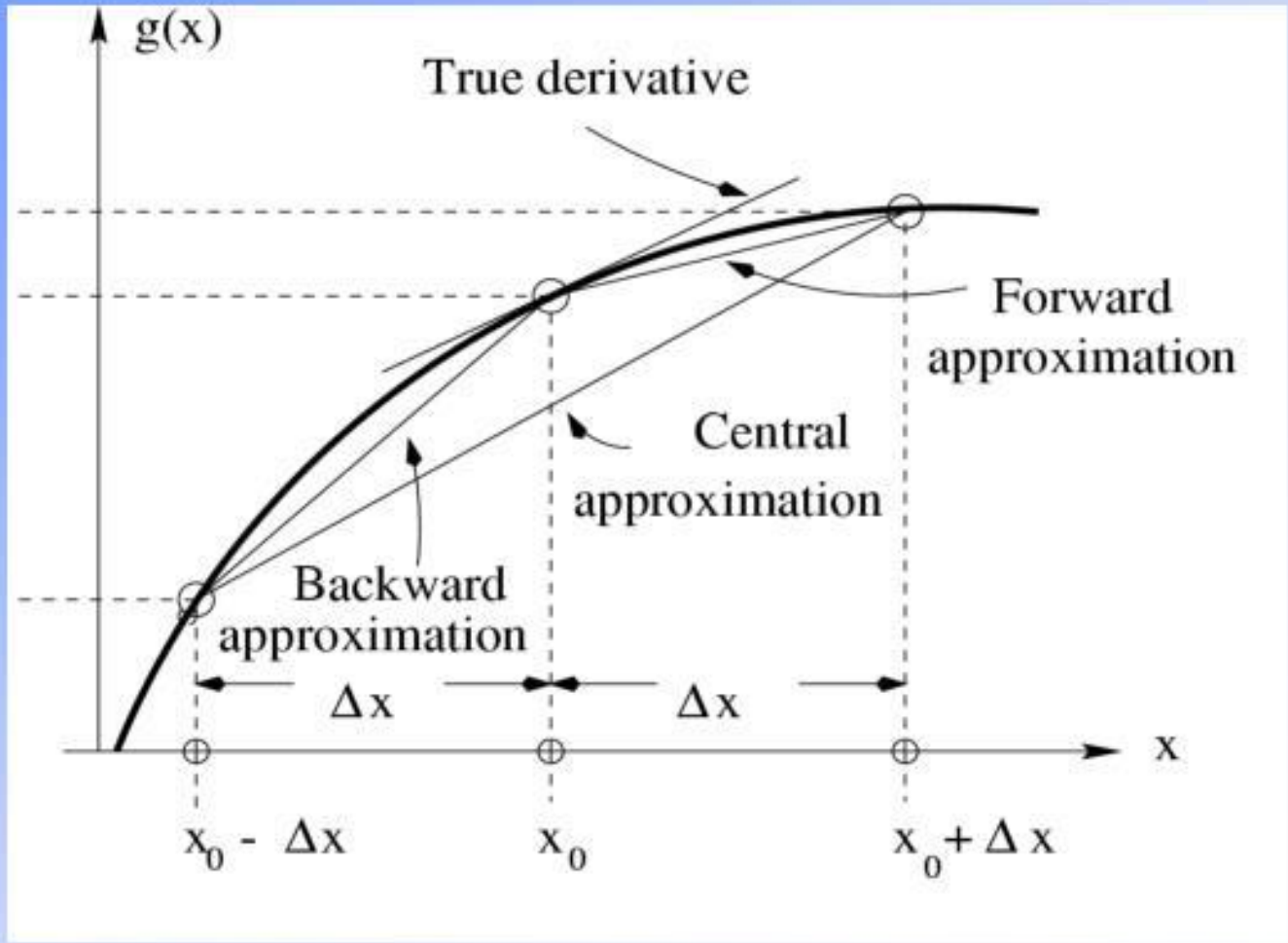
11. Value = prod* Ynew[(i+1)]

sum = sum+ value

end i loop

12. The value of function at given x ie f(x) is sum.

Finite difference approximation to derivatives



Consider a smooth function g(x). Taylor's theorem reads:

$$g(x_0+\Delta x)=g(x_0)+\sum_k \frac{\Delta x^k}{k!}g^{(k)}(x_0)$$

In particular:

$$g(x_0+\Delta x)=g(x_0)+\Delta x g^{(1)}(x_0)+O(\Delta x^2) \qquad (1)$$

$$g(x_0-\Delta x)=g(x_0)-\Delta x g^{(1)}(x_0)+O(\Delta x^2) \qquad (2)$$

$$Eq.(1)\rightarrow g^{(1)}(x_0)=\frac{g(x_0+\Delta x)-g(x_0)}{\Delta x}+O(\Delta x) \qquad (3)$$

$$Eq.(2)\rightarrow g^{(1)}(x_0)=\frac{g(x_0)-g(x_0-\Delta x)}{\Delta x}+O(\Delta x) \qquad (4)$$

$$Eqs.(3)-(4)\rightarrow g^{(1)}(x_0)=\frac{g(x_0+\Delta x)-g(x_0-\Delta x)}{2\Delta x}+O(\Delta x^2)$$

Now approximate partial derivatives.

Use finite differences:

$$q_t \approx \frac{q_i^{n+1} - q_i^n}{\Delta t} + O(\Delta t) : \text{forward in time (FT)}$$

$$q_x \approx \frac{q_{i+1}^n - q_{i-1}^n}{2\Delta x} + O(\Delta x^2) : \text{centred in space (CS)}$$

Then the PDE is replaced by a finite difference approximate operator:

$$L_a(q_i^n) = \frac{q_i^{n+1} - q_i^n}{\Delta t} + \lambda \frac{q_{i+1}^n - q_{i-1}^n}{2\Delta x} = 0$$

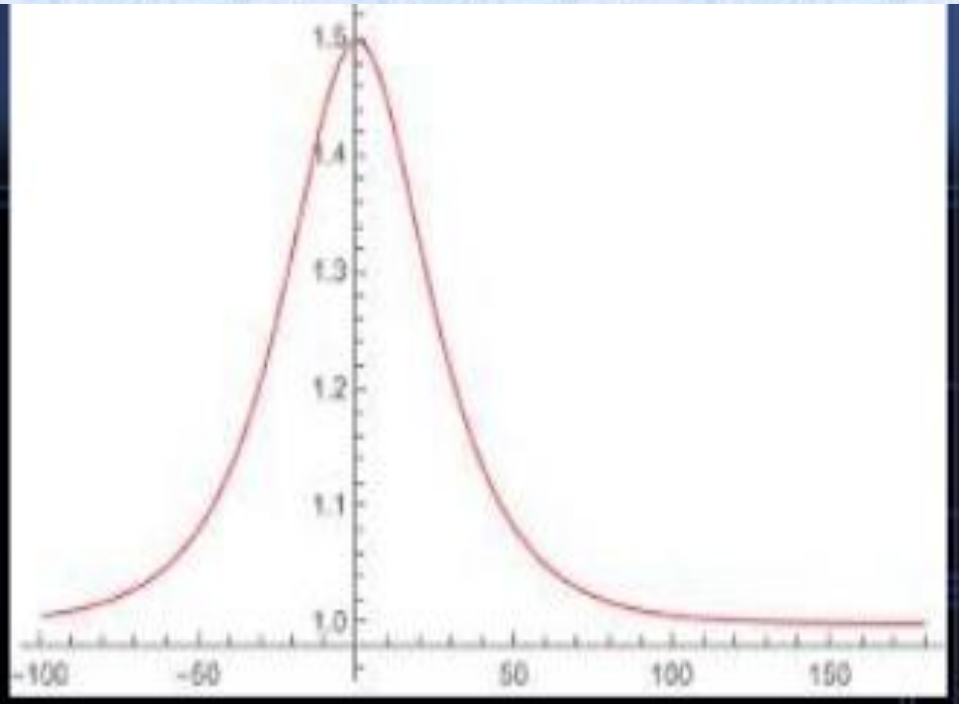
$$\xrightarrow{\text{numerical scheme}} q_i^{n+1} = q_i^n - \left(\frac{\lambda \Delta t}{\Delta x}\right)\left(\frac{q_{i+1}^n - q_{i-1}^n}{2\Delta x}\right)$$

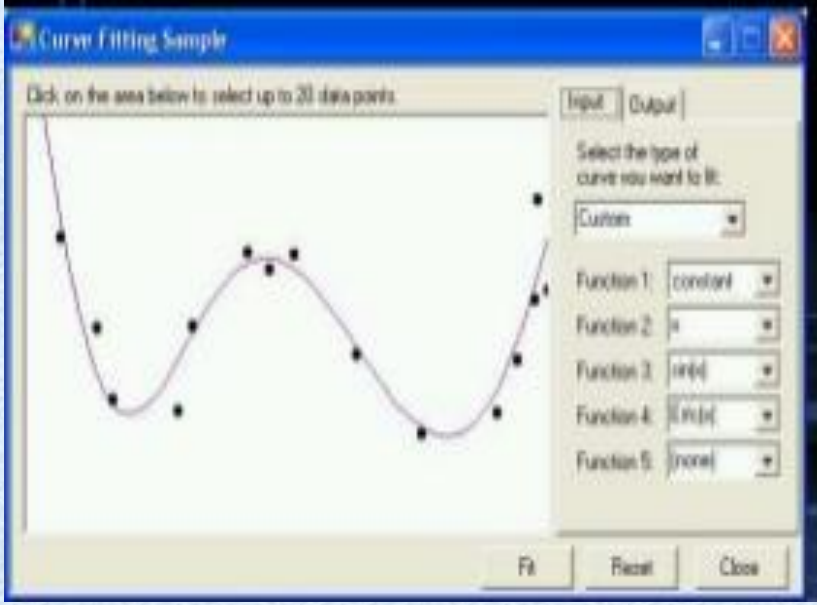
APPLICATION OF NUMERICAL METHODS IN DAY-TO-DAY LIFE

ESTIMATION OF OCEAN CURRENTS

MODELLING OF AIRFLOW OVER AIRPLANES

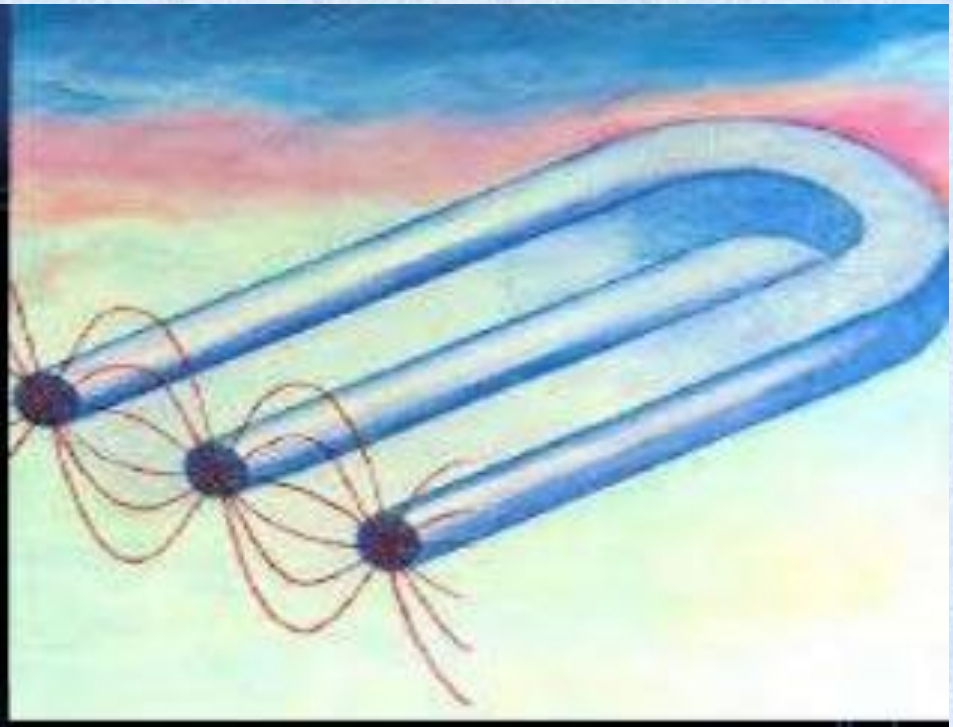
Shock wave





Curve fitting of tabular data

Electromagnetics



Shuttle tank separation


THANK YOU !!