Numerical Analysis

Created	@October 12, 2022 7:19 PM
Last Edited Time	@January 14, 2023 5:31 PM
By	Borhan
Email	

References:

- TheOrganicChemistryTutor
- Dr. Gajendra Purohit
- https://www.youtube.com/playlist?list=PLVKIC9j3jSYsm8GELqAMFJ_1ebbMpK-9U

Error

- Modelling Errors These errors arise during the modelling process when scientists ignore effecting factors in the model to simplify the problem. Also, these errors known as formulation errors.
- Data Uncertainty These errors are due to the uncertainty of the physical problem data and also known as data errors.
- Discretization Errors Computers represent a function of continuous variable by a number of discrete values. Also, scientists approximated replace complex continuous problems by discrete ones and this results in discretization errors.

1.3.3 Absolute and Relative Errors

Definition 3 (Absolute Error). *The absolute error $\hat{e}$ of the error e is defined as the absolute value of the error e*

$$\hat{e} = |x - x^*|.$$

Definition 4 (Relative Error). *The relative error $\tilde{e}$ of the error e is defined as the ratio between the absolute error $\hat{e}$ and the absolute value of the exact solution x*

$$\tilde{e} = \frac{\hat{e}}{|x|} = \frac{|x - x^*|}{|x|}, \; x \neq 0.$$

Absolute error is the difference between measured or inferred value and the actual value of a quantity.

The relative error is defined as the ratio of the absolute error of the measurement to the actual measurement.

Rounding error or Roundoff error's the difference between a rounded-off numerical value and the actual value

Computers represent numbers in finite number of digits and hence some quantities cannot be represented exactly. The error caused by replacing a number a by its **closest machine number is called the roundoff error** and the process is called correct rounding.

Example:

$x = 22/7$ (actual)

But, you got $x = 3.14$

A truncation error is the difference between an actual and a truncated, or cut-off, value

Truncation errors also sometimes called chopping errors are occurred when chopping an infinite number and replaced it by a finite number or by truncated a series after finite number of terms.

Example :

→ $x = \sqrt{2} = 1.14......$ (actual)

But yout got $x = 1.143$

Bisection Method

AKA **Bolzano method**.

Steps:

- Set x_1, x_2 based on $f(x_1) * f(x_2) < 0$
- Find intermediate point

$$x = \frac{x_1 + x_2}{2}$$

- Shift x_1, x_2 based on two condition
  - $f(x) * f(x_1) < 0$

    $x_2 = x$
  - $f(x) * f(x_1) > 0$

    $x_1 = x$

*NB: $f(x_1) * f(x_2) < 0$ means that, you have to choose x_1 and x_2 such a way that they would return the value of the $f(x_1), f(x_2)$ same or different but **must be opposite in sign**.*

Code for Bisection method:

```
#include <stdio.h>
#include <math.h>
#define f(x) (x * x * x - 2 * x - 5)

int main()
{
  double a = 0, b = 0;
```

```
  double x1 = 0, x2 = 0;

  // Step 1 : finding the value of x1 and x2
  while (1)
  {
    /*
      ekhane dui bhabe check kortechi,
      dhoren,
      1) (0 1), (1,2) , ...
      2) (0,-1), (-1,-2), ....
      ig, yk why.
    */
    if (f(a) * f((a - (double)1.0000)) < 0)
    {
      printf("%lf %lf", f(a), f((a - (double)1.0000)));
      x1 = a;
      x2 = a - 1;

      break;
    }
    if (f(b) * f((b + (double)1.0000)) < 0)
    {
      x1 = b;
      x2 = b + 1;
      break;
    }
    a--;
    b++;
  }
  // Step 2 & 3 : finding the root

  // we'll just jot down the steps which are written on the blog
  double x; // the intermediate poing
  double ans; // we're gonna store our answer here
  while (1)
  {
    x = (x1 + x2) / 2; // the second step
    // the third step
    if (f(x1) * f(x) < 0)
      x2 = x;
    if (f(x2) * f(x) < 0)
      x1 = x;
    // break the loop(idea from class)
    if (fabs((x1) - (x2)) <= 0.001)
    {
      ans = x;
      break;
    }
  }

  printf("%lf", ans);
  return 0;
}
```

Regula–Falsi Method

- 1) Set x_1, x_2 based on $f(x_1) * f(x_2) < 0$
- 2) Find intermediate point

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

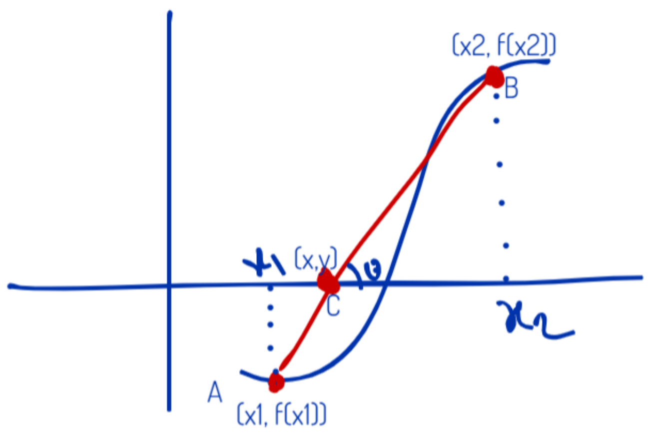
- 3) Shift x_1, x_2 based on two condition
  - $f(x) * f(x_1) < 0$

    $x_2 = x$
  - $f(x) * f(x_1) > 0$

    $x_1 = x$
- 4) Compare the previous and current answer, and repeat until you don't get the desired result.

NB: $f(x_1) * f(x_2) < 0$ means that, you have to choose x_1 and x_2 such a way that they would return the value of the $f(x_1), f(x_2)$ same or different but ***must be opposite in sign***. `(It is not going to be repeated again.)`

Derivation:



1) there are two points x1 and x2, and yk
the rule of choosing the x1 and x2, ig
2) draw a straight line or chord between the two points, then you will find the x
3) Slope of AB = Slope of CB

Slope of $AB =$ Slope of CB

$\Rightarrow \frac{f(x_1)-f(x_2)}{x_1-x_2} = \frac{0-f(x_2)}{x-x_2}$

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

Note :

(x, y) = (x, 0) $\Rightarrow$ y=0, check the graph again.

Slope $= \frac{y_1 - y_2}{x_1 - x_2}$

Code for implementing Regula–Falsi Method:

```
#include <stdio.h>
#include <math.h>
#define f(x) (x * x * x - 2 * x - 5)

int main()
{
  double a = 0, b = 0;
  double x1 = 0, x2 = 0;

  // Step 1 : finding the value of x1 and x2
  while (1)
  {
    /*
      ekhane dui bhabe check kortechi,
      dhoren,
      1) (0 1), (1,2) , ...
      2) (0,-1), (-1,-2), ....
      ig, yk why.
    */
    if (f(a) * f((a - (double)1.0000)) < 0)
    {
      printf("%lf %lf", f(a), f((a - (double)1.0000)));
      x1 = a;
      x2 = a - 1;

      break;
    }
    if (f(b) * f((b + (double)1.0000)) < 0)
    {
      x1 = b;
      x2 = b + 1;
      break;
    }
    a--;
    b++;
  }
  // Step 2 & 3 : finding the root

  // we'll just jot down the steps which are written on the blog
  double ans = a; // we're gonna store our answer here
  while (1)
  {
    double prevAns = ans;
    double x = ((x1 * f(x2) - x2 * f(x1)) /
          (f(x2) - f(x1)));
```

```
    ans = x;
    if (f(x) * f(x1) < 0)
    {
      x2 = x;
    }
    else
      x1 = x;
    // Calculating the the  literation of last two answer
    if (fabs((prevAns) - (ans)) <= 0.001)
    {
      break;
    }
  }

  // Printing the answer
  printf("%lf", ans);
  return 0;
}
```

Newton Raphson Method

> $f(x)$ **must be continuous and differentiable too.**

> It is the **fastest method.**

- Find x_1 and x_2
- Intermediate Point: (Use one of them)
  - Using middle point

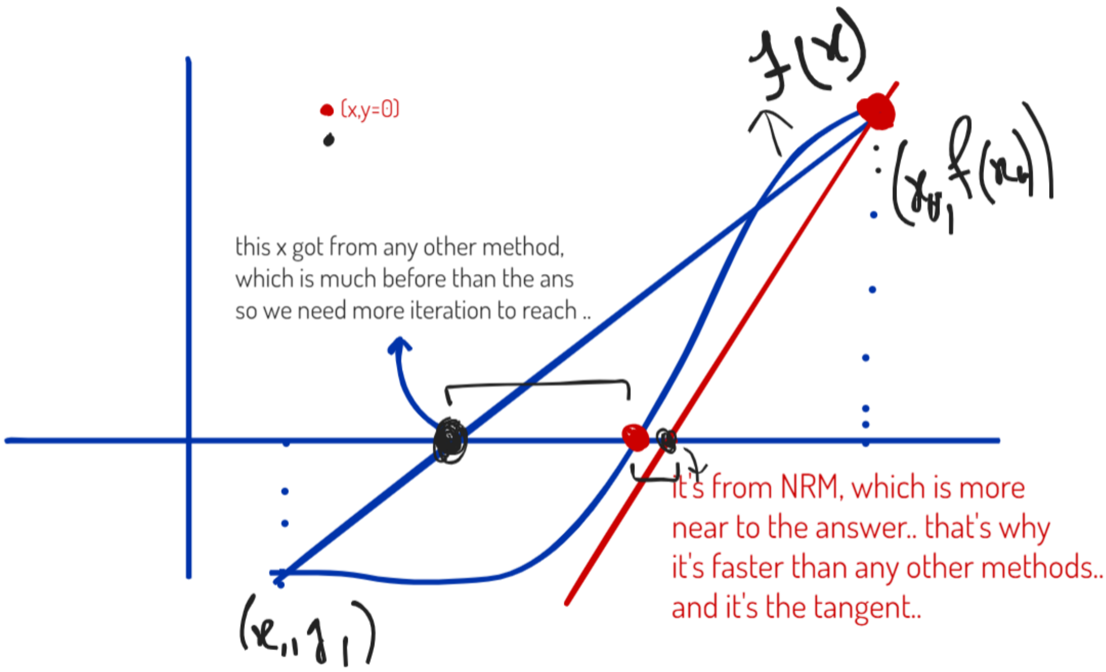
$$x = \frac{x_1 + x_2}{2}$$

  - Checking the function value(**recommended**)
    - $|f(x_1)| < |f(x_2)| \ \Rightarrow \ x = x_1;$
    - $|f(x_1)| > |f(x_2)| \ \Rightarrow \ x = x_2;$

  - Any point $\ x \ \epsilon \ (x_1, x_2)$

- Find roots : x1,….
  - $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$
- Repeat until the answer…..

Derivation:



Equation of tangent ⇒

$$y - y_0 = f'(x_0)(x - x_0)$$

$$\Rightarrow 0 - f(x_0) = f'(x_0)(x - x_0)$$

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

The tangent line to y = f(x) at the point (a, f(a)) has equation
y = f(a)+(x − a)f′(a).

Code for Newton Raphson Method:

```
#include <stdio.h>
#include <math.h>
#define f(x) (x * x * x - 2 * x - 5)
```

```
#define diff(x) (3 * x * x - 2)
int main()
{
    double a = 0, b = 0;
    double x1 = 0, x2 = 0;

    // Step 1 : finding the value of x1 and x2
    while (1)
    {
        /*
            ekhane dui bhabe check kortechi,
            dhoren,
            1) (0 1), (1,2) , ...
            2) (0,-1), (-1,-2), ....
            ig, yk why.
        */
        if (f(a) * f((a - (double)1.0000)) < 0)
        {
            printf("%lf %lf", f(a), f((a - (double)1.0000)));
            x1 = a;
            x2 = a - 1;

            break;
        }
        if (f(b) * f((b + (double)1.0000)) < 0)
        {
            x1 = b;
            x2 = b + 1;
            break;
        }
        a--;
        b++;
    }

    // MAIN CODE
    double x0;
    double ans;
    if (fabs(f(x1)) < fabs(f(x2)))
        x0 = x1;
    else
        x0 = x2;

    while (1)
    {
        double x = x0 - (f(x0) / diff(x0));
        if (fabs(x - x0) <= 0.001)
        {
            ans = x;
            break;
        }
        x0 = x;
    }
    printf("%lf", ans);
    return 0;
}
```

Secant Method

Same as Regula-Falsi method.

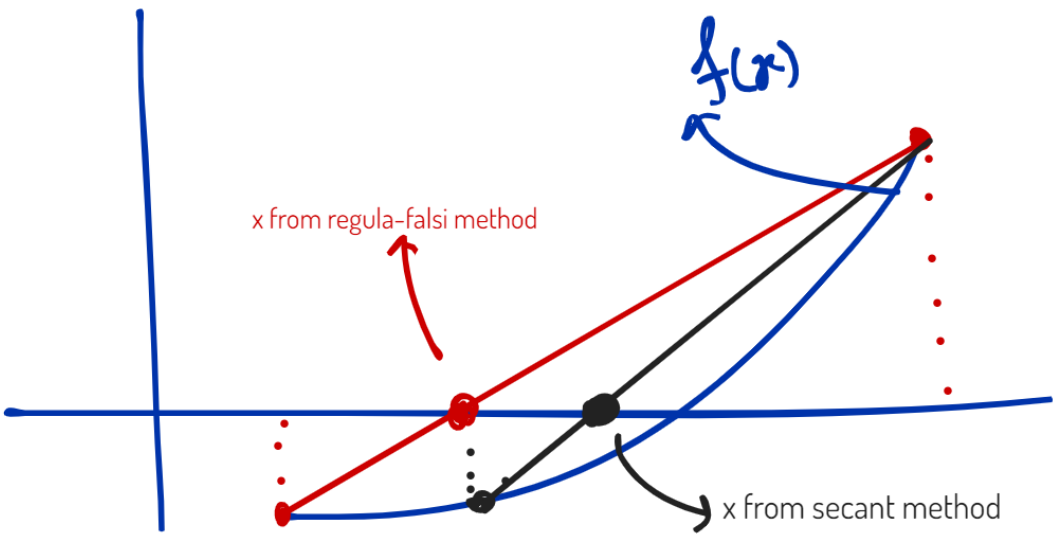
Steps:

- 1) Find x_1 and x_2
- 2) Find intermediate point

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

- 3) Shift x_1, x_2 such as below
  - $x_1 = x_2$

    $x_2 = x$
- 4) Compare the previous and current answer, and repeat if you don't get the desired result.



Code:

```
#include <stdio.h>
#include <math.h>
#define f(x) (x * x * x - 2 * x - 5)

int main()
{
  double a = 0, b = 0;
  double x1 = 0, x2 = 0;

  // Step 1 : finding the value of x1 and x2
  while (1)
  {
    /*
      ekhane dui bhabe check kortechi,
      dhoren,
      1) (0 1), (1,2) , ...
      2) (0,-1), (-1,-2), ....
      ig, yk why.
    */
    if (f(a) * f((a - (double)1.0000)) < 0)
    {
      printf("%lf %lf", f(a), f((a - (double)1.0000)));
      x1 = a;
      x2 = a - 1;

      break;
    }
    if (f(b) * f((b + (double)1.0000)) < 0)
    {
      x1 = b;
      x2 = b + 1;
      break;
    }
    a--;
    b++;
  }
  // Step 2 & 3 : finding the root

  // we'll just jot down the steps which are written on the blog
  double ans = 1e9; // Just assume a number which can never be the answer
  while (1)
  {
    double prevAns = ans;
    double x = ((x1 * f(x2) - x2 * f(x1)) / (f(x2) - f(x1)));
    ans = x;
    x1 = x2;
    x2 = x;
    // Calculating the the  literation of last two answer
    if (fabs((prevAns) - (ans)) <= 0.001)
    {
      break;
    }
```

```
  }

  // Printing the answer
  printf("%lf", ans);
  return 0;
}
```

Iteration Method

- Find $phi(x)$ and $phi'(x)$

$$y = x^3 + x^2 - 1$$

$$y = 0$$

$$\Rightarrow x^2(x+1) = 1$$

$$\Rightarrow x = \frac{1}{\sqrt{1+x}} = \phi(x)$$

$$\phi'(x) = \frac{d}{dx}\phi(x)$$

- Find x_1 and x_2
- Find x_0

$$x_0 = \frac{x_1 + x_2}{2}$$

- $phi'(x_0) < 1$:
  - $x_n = phi(x_n)$
- $phi'(x_0) >= 1$:NOT POSSIBLE
- Repeat

Code:

```
#include <stdio.h>
#include <math.h>

#define f(x) (x * x * x + x * x - 1)
#define phi(x) 1 / sqrt(1 + x)
#define diffPhi(x) (0.5 / sqrt(1 + x))

/*
y = x^3 - 2x - 5, is unsolvable by Iteration method.
idky.

#define f(x) (x * x * x - 2* x - 5)
#define phi(x) 5/(x*x - 2)
#define diffPhi(x)(10*x/((x*x - 2)*(x*x - 2)))

*/
int main()
{
  double a = 0, b = 0;
  double x1 = 0, x2 = 0;

  // Step 1 : finding the value of x1 and x2
  while (1)
  {
    /*
      ekhane dui bhabe check kortechi,
      dhoren,
      1) (0 1), (1,2) , ...
      2) (0,-1), (-1,-2), ....
      ig, yk why.
    */
    if (f(a) * f((a - (double)1.0000)) < 0)
    {
      printf("%lf %lf", f(a), f((a - (double)1.0000)));
      x1 = a;
      x2 = a - 1;

      break;
    }
    if (f(b) * f((b + (double)1.0000)) < 0)
```

```
    {
      x1 = b;
      x2 = b + 1;
      break;
    }
    a--;
    b++;
  }
  // Step 2 & 3 : finding the root
  // we'll just jot down the steps which are written on the blog
  double ans = 1e9; // Just assume a number which can never be the answer
  double x0 = (x1 + x2) / 2;

  if (abs(diffPhi(x0)) < 1)
  {
    while (1)
    {
      double xn = phi(x0);
      if (fabs(xn - x0) <= 0.001)
      {
        ans = xn;
        break;
      }
      x0 = xn;
    }
  }
  else
  {
    printf("NOT FOUND");
    return 0;
  }

  // Printing the answer
  printf("%lf", ans);
  return 0;
}
```

Interpolation

When interpolation?

- We know the value of x and y of a function but don’t know the function (like y=f(x)=x^2 + 5)

Equal Intervals:

x = 3   5   7   9   11

y= 2   6   8   12   13

The difference between x_i and x_{i+1} is equal. **2 for this example.**

if,

x= 3.5 (at starting) → we will use **Newton Forward Interpolation**

x= 5.1 - 7.9 → we will use **Central Difference Interpolation**

x=9.5 (at end) → we will use **Newton Backward Interpolation**

Newton Forward Interpolation (for Equal Intervals):

Procedure:

- Create a difference table
- Use the formula and calculate
    - h = difference b/w two contiguous x
    - f(a), f^2(a), …. → upper value (1st row)

let,

$$P_n = A_0 + A_1(x-a) + A_2(x-a)(x-a-h) + \dots$$

$x=a$, $f(a) = A_0$

$x=a+h$, $f(a+h) = A_0 + A_1 h$

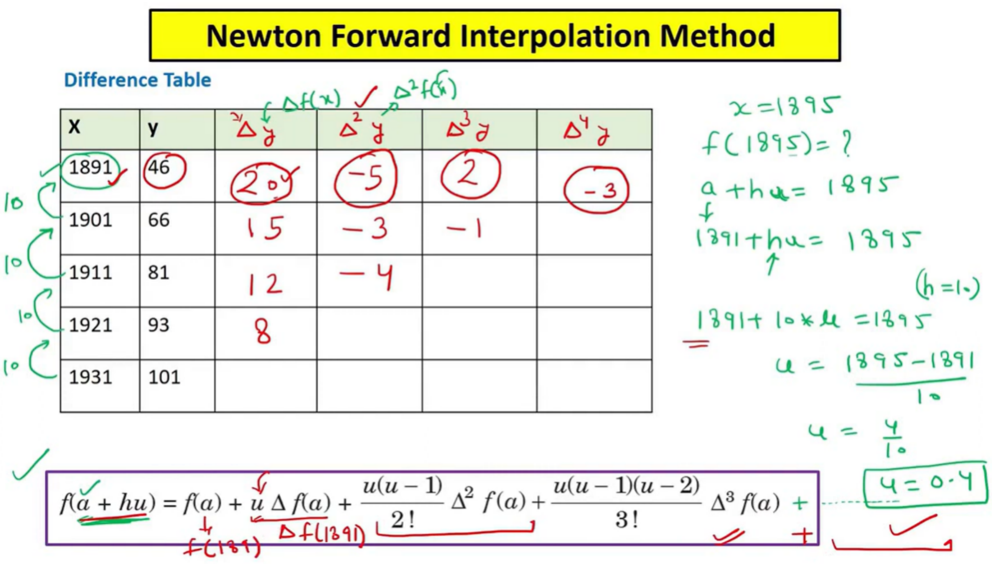
$$\Rightarrow A_1 = \frac{f(a+h)-f(a)}{1!\,h} = \frac{1}{h}\Delta f(a)$$

$$A_2 = \frac{1}{2!\,h^2}\Delta^2 f(a)$$

$$f(x) = f(a) + \frac{1}{h}\Delta f(a) + \frac{1}{2!\,h^2}\Delta^2 f(a)$$

$x = a+uh$

$$f(a+uh) = f(a) + u\Delta f(a) + u(u-1)\Delta^2 f(a) + \dots$$



Newton Backward Interpolation (For Equal Interval):

Procedure :

- Create a difference table
- Use the formula an calculate
  - f(a), f^2(a), … → the lower value (the last value of every column)

let, $P_n(x) = A_0 + A_1(x-a-nh) + A_2(x-a-nh)(x-a-nh-h) + \dots$

$x = a+nh$, $P_n(a+nh) = A_0 = f(a+nh)$

$x = a+nh+h$, $P_n(a+nh+h) = A_0 + A_1 h$

$$\Rightarrow A_1 = \frac{f(a+nh+h) - f(a+nh)}{h} = \frac{1}{h}\nabla f(a)$$

$$A_2 = \frac{1}{2!h^2}\nabla^2 f(a)$$

$x = a+nh+hu$

$$P_n(x) = P_n(a+nh+hu) = f(a+h) + u\nabla f(a+n-h) + \frac{u(u+1)}{2!}\nabla^2 f(a+n-h) + \dots$$

$$y_{1925} = y_{1931} + u\nabla y_{1931} + \frac{u(u+1)}{2!}\nabla^2 y_{1931} + \frac{u(u+1)(u+2)}{3!}\nabla^3 y_{1931} + \frac{u(u+1)(u+2)(u+3)}{4!}\nabla^4 y_{1931}$$

$$= 101 + (-.6)(8) + \frac{(-.6)(.4)}{2!}(-4) + \frac{(-.6)(.4)(1.4)}{3!}(-1) + \frac{(-.6)(.4)(1.4)(2.4)}{4!}(-3)$$

$= 96.8368$ thousands.

x	y	∇y	$\nabla^2 y$	$\nabla^3 y$	$\nabla^4 y$
1891	46	20	-5	2	-3
1901	66	15	-3	-1	
1911	81	12	-4		
1921	93	8			
1931	101				

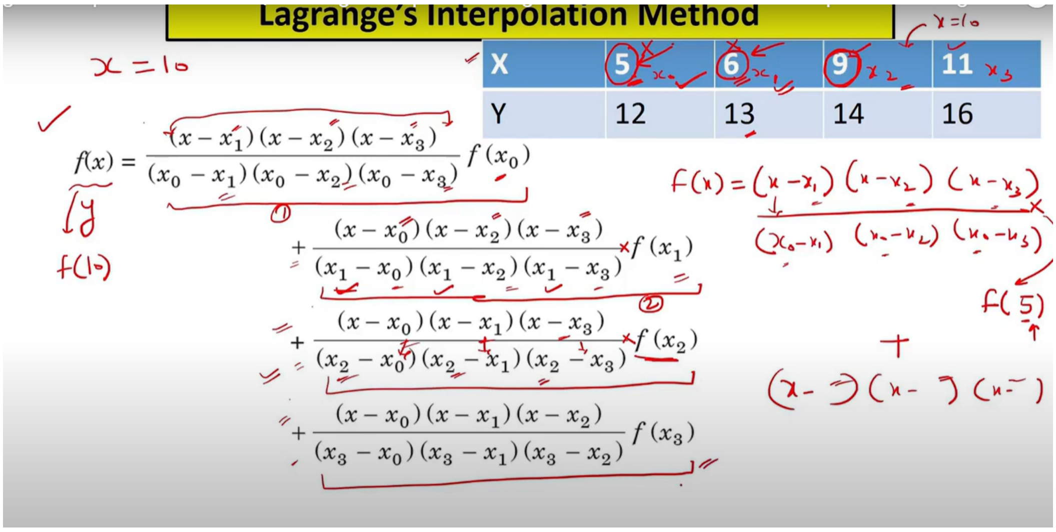
For Unequal Intervals

Difference between x(i) and x(i+1) is not equal for all.

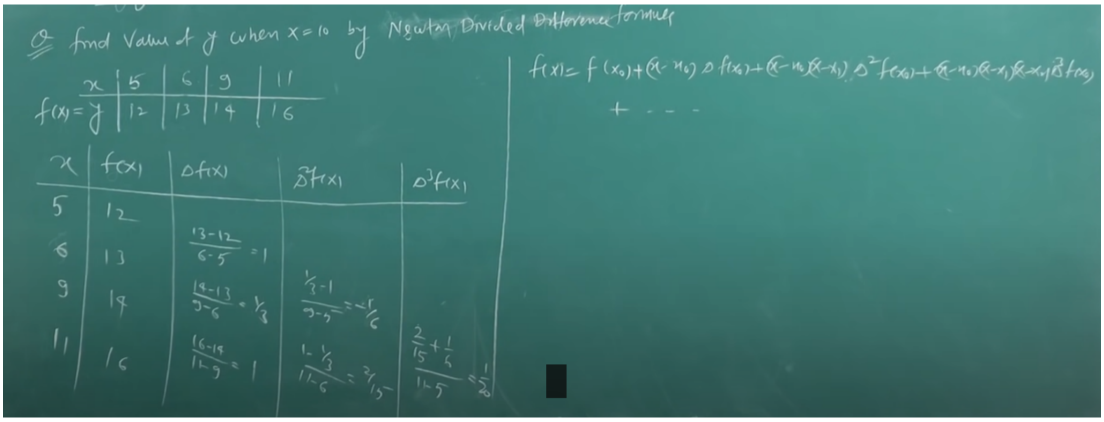
Lagrange’s Interpolation Method:

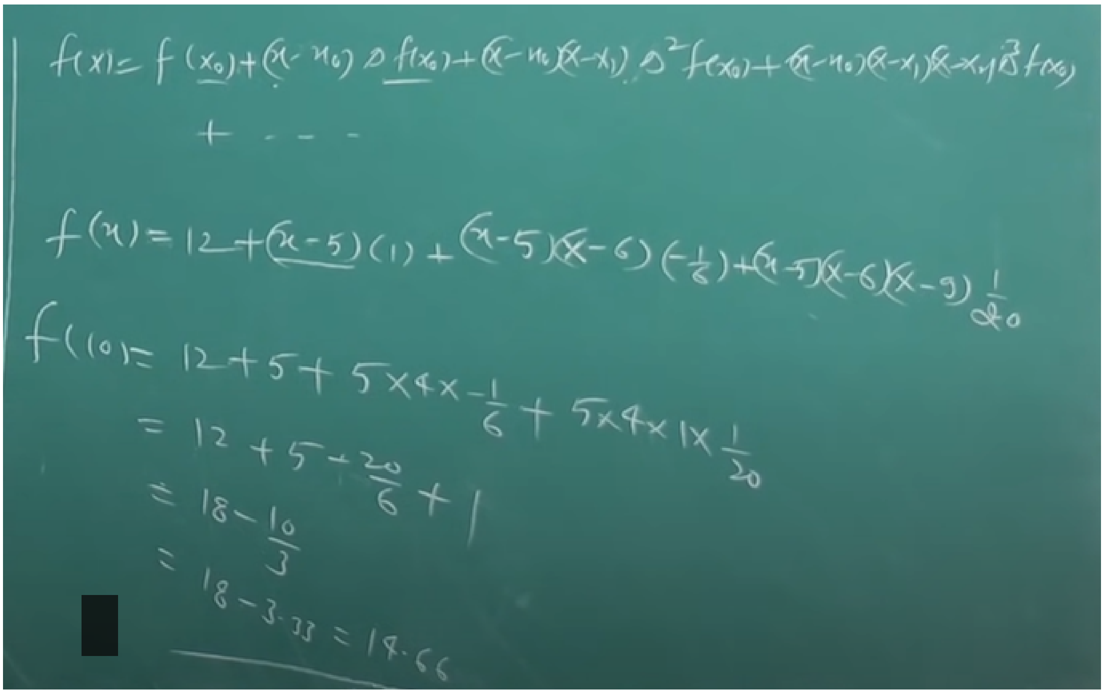
Procedure :

- Evaluate the formula and put the value of x0, x1, … and f(x)…
- x = the value given in question



Newton Divided Difference:





Shift Operator (E)

$Ef(x) = f(x+h)$

$E^n f(x) = f(x+nh)$

$E^{-1} f(x) = f(x-h)$

$E^{-n} f(x) = f(x-nh)$

$\triangle f(x) = f(x+h) - f(x)$ *[forward difference operator]*

$\triangledown f(x) = f(x) - f(x-h)$ *[backward difference operator]*

Relation between forward/backward difference operator to Shift Operator

$$Prove, E = 1 + \triangle$$
$$\triangle f(x) = f(x+h) - f(x)$$
$$or, \triangle f(x) = Ef(x) - f(x)$$
$$Hence, \triangle = E - 1$$

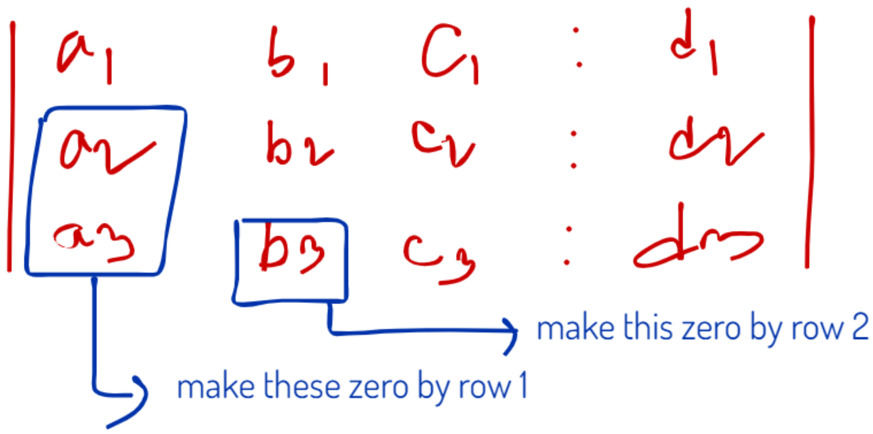
$$Prove, \nabla = 1 - E^{-1}$$
$$\nabla f(x) = f(x) - f(x-h)$$
$$or, \triangle f(x) = f(x) - E^{-1} f(x)$$
$$or, \nabla = 1 - E^{-1}$$

Solution of Linear Algebraic Method

- Directed Method
- Indirect Method

Gauss Elimination Method:

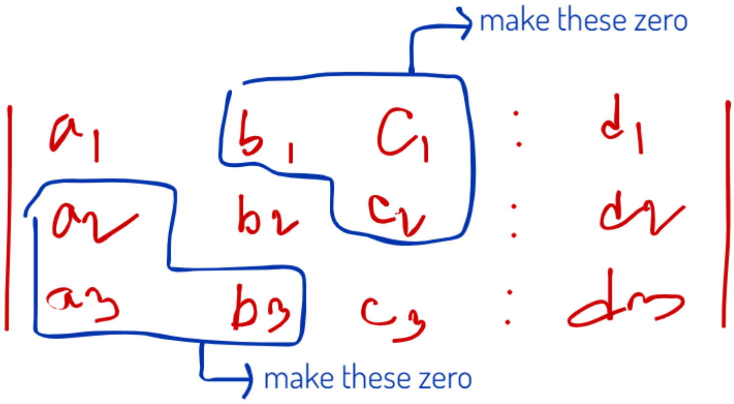
<u>Procedure:</u>



Gauss-Jordan Method:

<u>Procedure:</u>

- Convert the matrix to diagonal matrix



Numerical Integration

$h = \frac{b-a}{n}$

n = stripe, (generally, it's 6)

Trapezoidal Rule

$\int_a^b f(x)dx = h(\frac{y_0+y_n}{2} + y_1.... + y_n)$

Applicable for any no. interval.

Simpson one-third rule

$\int_a^b f(x)dx = \frac{h}{3}(y_0 + y_n + 4(y_1 + y_3 + ...) + 2(y_2 + y_4 + ...))$

Applicable for only **even intervals.**

Simpson three-by-eight rule

$\int_a^b f(x)dx = \frac{3h}{8}(y_0 + y_n + 3(y_2 + y_4...) + 2(y_1 + y_3 + y_6 + ...))$

Applicable for only **multiple of 3 intervals.**

LU Decomposition:

AKA **Factorization Method, Cholesky's Method**

1) $$A = LU = \begin{vmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{vmatrix} \begin{vmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{vmatrix}$$

2) $Ax = B$

$\Rightarrow LUx = B$ $\quad \boxed{Ux = y}$

$\Rightarrow Ly = B$

$\boxed{y_1, y_2, y_3}$

3) $Ux = y$

$$\Rightarrow \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Jacobi Method:

* $a_1x + b_1y + c_1z = D_1$

$a_2x + b_2y + c_2z = D_2$

$a_3x + b_3y + c_3z = D_3$

Let

$|a_1| > |b_1| + |c_1|$

$|b_2| > |a_2| + |c_2|$

$|c_3| > |b_3| + |a_3|$

Not mandatory but it reduces the iteration

2) $x = \frac{1}{a_1}(D_1 - b_1y - c_1z)$

$y = \frac{1}{b_2}(D_2 - a_2x - c_2z)$

$z = \frac{1}{c_3}(D_3 - a_3x - b_3y)$

3) $x_0 = 0, \quad y_0 = 0, \quad z_0 = 0$

$x_1 = \frac{D_1}{a_1}$ | $x_2 = \frac{1}{a_1}(D_1 - b_1y_1 - c_1z_1)$

$y_1 = \frac{D_2}{b_2}$ ⋮

$z_1 = \frac{D_3}{c_3}$