Bisection Method

Bisection method is a numerical technique used to find the roots of a continuous function. It is based on the intermediate value theorem, which states that if a function f(x) changes sign over an interval [a, b], then there exists at least one root c within [a, b] such that f(c)=0. The method iteratively halves the interval by evaluating f(x) at the midpoint, narrowing down the subinterval where the root lies, until a desired level of accuracy is achieved. It is simple, reliable, and works well if the function is continuous and the initial interval is chosen The correctly.

$$C = \frac{a+b}{2}$$

Algorithm:

1.Calculate the midpoint:

$$C = \frac{a+b}{2}$$

2. Evaluate f(c):

- If f(c) =0, c is the root, and the process stops.
- Otherwise, proceed to the next step.

3. Determine the new interval:

- If $f(c)\cdot f(a)<0$, the root lies in [a, c], so set b=c.
- If $f(c)\cdot f(b)<0$, the root lies in [c, b], so set a=c.

4.Repeat steps 1–3 until $|b-a|<\epsilon$.

For equation : $f(x)=x^3-x-2$

Sample input: a=1, b=2 , tolerance =.001

Sample output: Iteration 1: c = 1.5, f(c) = -0.125

Iteration 2: c = 1.75, f(c) = 1.60938

Iteration 3: c = 1.625, f(c) = 0.666016

The approximate root is: 1.52148

False Position Method

The **false position method** (or **regula falsi method**) is a root-finding technique that improves upon the bisection method by interpolating the root more efficiently. It is used to find a root of a continuous function f(x) over an interval [a, b] where f(a)·f(b)<0. Unlike the bisection method, which uses the midpoint of the interval, the false position method uses a weighted interpolation formula to estimate the root.

$$C = b - \frac{f(b).(b-a)}{f(b)-f(a)}$$

Algoritms:

1. Begin with an interval [a, b] such that f(a)·f(b)<0.
2. Compute c using the formula above.
3. Evaluate f(c):

   If f(c)=0, c is the root.

4. Otherwise, update the interval:

   - If f(a)·f(c)<0, set b=c.
   - If f(b)·f(c)<0, set a=c.

5. Repeat the process until the desired tolerance is achieved.

Sample input : a=1, b=2 , tolerance =.001

Sample output: Iteration 1: c = 1.33333, f(c) = -0.962963

Iteration 2: c = 1.46269, f(c) = -0.333338

Iteration 3: c = 1.50402, f(c) = -0.046892

Iteration 4: c = 1.50804, f(c) = -0.000511

The approximate root is: 1.50804

Cramer’s Rule

Cramer’s Rule is a method for solving a system of linear equations using determinants. It is applicable when the system has n equations and n unknowns, and the coefficient matrix has a non-zero determinant.

Formula:

Given a system of linear equations:

a11x1+a12x2+a13x3=b1

a21x1+a22x2+a23x3=b2

a31x1+a3x2+a33x3=b3

Represent the system in matrix form:

Ax=b

Cramer’s Rule states :

$X_i = \frac{\det(Ai)}{\det(A)}$ where i = 1,2,3

det(A): Determinant of the coefficient matrix A.

Ai: Matrix obtained by replacing the i-th column of A with the constants column b.

Algorithm

- Coefficient matrix A (size n×n).
- Constants column b.

- Compute det(A).
- For each variable x_i :
  - Form A_i by replacing the i-th column of A with b.
  - Compute det(A_i) .
- If det(A)=0, the system has no unique solution (either inconsistent or infinitely many solutions).:
- For each i: $X_i = \frac{\det(Ai)}{\det(A)}$

Sample input :

2	1	-1	8
-3	-1	2	-11
-2	1	2	-3

Sample output: x1=1

x2=-2

x3 =-3

Secant Method

The Secant Method is an iterative numerical technique used to find the root of a real-valued function. Unlike the Newton-Raphson method, it does not require the derivative of the function but uses two initial guesses to approximate the root. The iterative formula is:

$$X_{n+1}=X_n+ f(\mathrm{Xn})\,\frac{X\mathrm{n}-\mathrm{Xn}-1}{f'(\mathrm{Xn})-f(Xn-1)}$$

This method is especially useful when the derivative of the function is difficult or expensive to compute.

Algorithm :

1.Input :

- Function is f(x).
- Derivative is f’(x).
- Initial guess x_0=0.
- Tolarence is e.
- Iteration number n.

2.Iteration:

- While n<N_{max}
- Calculate X_{n+1}
- If absolute value of (X_{n+1}-X_n) < e
- Then stop and return X_{n+1} as root.
- Else update X_n=X_{n+1}, X_{n-1}=X_n , n by 1.

3.Output :

- If convergence criteria are met, return x_{n+1}.
- Otherwise, after N_{max} iterations, report failure to converge.

Sample Input :

- Initial guess X_0=1,X_1=2.
- Tolerance (e): 0.0001
- Maximum iterations (Nmax): 100

Sample Output :

Root: 1.4142
Number of iterations: 6

Gauss-Seidel Method

The Gauss-Seidel method is an iterative technique for solving a system of linear equations of the form:

$$Ax=b$$

Here, A is a coefficient matrix, x is the vector of unknowns, and b is the constant vector. It refines the solution by solving each equation for one variable at a time and updating the solution immediately.

Algorithms:

1.Input:

- Coefficient matrix A.
- Constant vector .
- Initial guess vector $x_0 = 0$.
- Tolarence is e.
- Maximum iteration number N.

- 2.Iteration:
  - Set n= 0.
  - While $n < N_{max}$
  - Calculate

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right)$$

$$\text{Error} = \max_i |x_i^{(k+1)} - x_i^{(k)}|$$

  - If value of Error < e
  - Then stop and return X^{k+1} as solution.
  - Else update $X^k=X^{k+1}$, n by 1.

3.Output :

- If convergence criteria are met, return x^{K+1}.
- Otherwise, after N_{max} iterations, report failure to converge.

Sample Input :

System of Equations

2x – 2y + 3z = 20
x + 3y – z = 12
3x – y + 4z = 22

e = .001

N = 100

Sample Output :

x1=21
x2=-7
x3=12 Number of iterations: 7

Least Squares Regression

The Least Squares Regression method is a statistical technique used to find the best-fitting line (or curve) through a set of data points by minimizing the sum of the squared differences (errors) between the observed values and the values predicted by the line.

For a linear relationship y=mx+c, the least squares method determines the slope mmm and intercept ccc of the line.

Equations

1. **Slope (m):**

$$m = \frac{n \sum (xy) - \sum x \sum y}{n \sum (x^2) - (\sum x)^2}$$

2. **Intercept (c):**

$$c = \frac{\sum y - m \sum x}{n}$$

Where n is the number of data points .

x,y are the data points.

Algorithm :

1.Input:

- Enter the range n of data points.
- Input x and y values for n points.

2.Calculate Sums:

- Compute $\sum x$, $\sum y$, $\sum xy$, $\sum (x)$^2 using a loop.

3.Find Coefficients:

- Compute **b** = $\frac{\boldsymbol{n} * \sum xy - \sum x * \sum y}{\boldsymbol{n} * \boldsymbol{s} \sum (x)\hat{}2 - (\sum x)^{\mathbf{2}}}$
- Compute **A** = $\frac{\sum y - \boldsymbol{b} * \sum x}{\boldsymbol{n}}$

a = $\boldsymbol{a} + \boldsymbol{bx}$

4.Output Results:

- Print the values of a and b.

Sample Input:

Enter the Range of x and y:
5
Enter the input of X And Y:
1 0.5
2 2
3 4.5
4 8
5 12.5

Sample Output:

The value of a :0.5

The value of b :2

Fitting a Transcendental Function

Fitting transcendental equations involves approximating data using equations with functions such as **exponential**, **logarithmic**, or **trigonometric forms**. The method often transforms the equation into a linear form, applies the **least squares method** to determine the unknown constants, and then converts the results back to the original form of the equation.

$$y=ae^{bx}$$

$$\ln(y) = \ln(a) + bx$$

$$y = a + bx$$

$$\mathbf{b} = \frac{\boldsymbol{n}*\sum lnxy-\sum lnx*\sum lny}{\boldsymbol{n}*\boldsymbol{s}\sum(lnx)\text{^}2-(\sum lnx)^{\mathbf{2}}}$$

$$\mathbf{A} = \frac{\sum lny-\boldsymbol{b}*\sum lnx}{\boldsymbol{n}}$$

Algorithm :

1.Input:

- Enter the range n of data points.
- Input x and y values for n points.

2.Calculate Sums:

- Compute $\sum lnx$, $\sum lny$, $\sum lnxy$, $\sum(lnx)$^2 using a loop.

3.Find Coefficients:

- Compute **b** = $\frac{\boldsymbol{n}*\sum lnxy-\sum lnx*\sum lny}{\boldsymbol{n}*\boldsymbol{s}\sum(lnx)^\wedge 2-(\sum lnx)^{\boldsymbol{2}}}$
- Compute **A** = $\frac{\sum lny-\boldsymbol{b}*\sum lnx}{\boldsymbol{n}}$

a = $\boldsymbol{e^{A}}$

4.Output Results:

- Print the values of a and b.

Sample Input:

Enter the Range of x and y:
5
Enter the input of X And Y:
1 0.5
2 2
3 4.5
4 8
5 12.5

Sample Output:

The value of a :0.5

The value of b :1.99

Euler's Method

Euler's method is a straightforward numerical technique used to approximate the solution of an ordinary differential equation (ODE) of the form:

$$dx/dy = f(x,y)$$

The method starts with an initial value $y(x_0)= y_0$ and iteratively calculates the values of y at subsequent points. It uses a simple formula to find the next value of y based on the current value and the slope of the function.

The iterative formula is:

$$y_{n+1}=y_n+h\cdot f(x_n,y_n)$$

Where:

- h is the step size (the distance between the current and the next point).
- Y_n is the current value of y.
- $f(x_n,y_n)$is the value of the function at (x_n,y_n).
- y_{n+1} is the next value of y.

Algorithm :

1.Input:

Read x_0, y_0, step size h, and x_{end}.

2.Initialize:

Set x = x_0, y = y_0, and compute n = $\frac{x_{end}- x_0}{h}$.

3.Iterate:

For i=0 to n−1:

Calculate $y_{n+1}=y_n+h\cdot f(x_n,y_n)$
Update $y_n = y_{n+1}$
Update $x_{n+1} = x_n + h$.
Print x, y.

4.End:

Show the value of x,y.

Sample Input:

Enter the initial value of x and y:
1   2
Enter the step size:
0.5
Enter the end value of x:
2

Sample Output:

X: 1   Y: 2

X: 1.5   Y: 4

X: 2   Y: 7.875

Simpson's 1/3 Rule

Simpson's 1/3 Rule is a numerical method used for approximating the definite integral of a function. It is particularly effective when the function can be approximated by a second-degree polynomial over small subintervals. The rule is derived from fitting a parabola through three consecutive points and integrating it.

The formula for the rule is:

$$\int_a^b f(x)dx \approx \frac{h}{3}\left[f(a) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + f(b)\right]$$

Where:

- a and b are the lower and upper limits of integration.
- h=(b−a)/n is the step size.
- n is the number of subintervals (must be even).
- x1,x2… are the intermediate points in the range [a, b].

Algorithm for Simpson's 1/3 Rule

1. Input:

- The function f(x).
- The limits of integration a and b.
- The number of subintervals n (must be even).

2. Calculate Step Size:

- Compute h=(b−a)/h.

*3. **Compute the Integral**:*

- Initialize the result: Result=f(a)+f(b)
- Add the contributions of the intermediate points:
  - For odd indices i: Result+=4·f(x_i)
  - For even indices ii: Result+=2·f(x_i).
- Multiply the entire result by h\3

4. Output:

- The approximate value of the definite integral.

Sample Input:

Enter the value of h:
0.5
Enter the range:(lower and higher value)
1   1.5

Sample Output:

Result of Simpsons 1/3:
1.0625

Simpson's 3/8 Rule

Simpson’s 3/8 Rule is another numerical method used to approximate the definite integral of a function. It is based on fitting a cubic polynomial to the function over three subintervals (four points). This rule is particularly useful when the number of subintervals nnn is a multiple of 3.

The formula for Simpson’s 3/8 Rule is:

$$\int_a^b f(x)dx \approx \frac{3h}{8}\left[f(a) + 3f(x_1) + 3f(x_2) + 2f(x_3) + \cdots + f(b)\right]$$

Where:

- a and b are the lower and upper limits of integration.
- h=(b−a)/n is the step size.
- n is the number of subintervals (must be multiple of 3).
- x1,x2… are the intermediate points in the range [a, b].

Algorithm for Simpson's 3/8 Rule

1. Input:

- The function f(x).
- The limits of integration a and b.
- The number of subintervals n (must be multiple of 3).

2. Calculate Step Size:

- Compute h=(b−a)/h.

*3. **Compute the Integral**:*

- Initialize the result: Result=f(a)+f(b)
- Add the contributions of the intermediate points:
  - For indices I divisible by 3 : Result+=2·f(x_i)
  - For all other indices ii: Result+=3·f(x_i).
- Multiply the entire result by 3h\8.

4. Output:

The approximate value of the definite integral.

Sample Input:

Enter the value of h:
0.5
Enter the range:(lower and higher value)
1   1.5

Sample Output:

Result of Simpsons 3/8 :
1.19531