ALGORITHM

Numerical Methods in Chemical Engineering (Practical)

(BTCH-18406)

Semester-IV

Department of Chemical Engineering

S.B.S STATE UNIVERSITY GURDASPUR

Bisection Method Algorithm

1. Start
2. Define function f(x)
3. Choose initial guesses x0 and x1 such that f(x0)f(x1) < 0
4. Choose pre-specified tolerable error e.
5. Calculate new approximated root as x2 = (x0 + x1) / 2
6. Calculate f(x0)f(x2)
    a. if f(x0)f(x2) < 0 then x0 = x0 and x1 = x2
    b. if f(x0)f(x2) > 0 then x0 = x2 and x1 = x1
    c. if f(x0)f(x2) = 0 then goto (8)
7. if |f(x2)| > e then goto (5) otherwise goto (8)
8. Display x2 as root.
9. Stop

Sardar Beant Singh State University, Gurdaspur
Department of Chemical Engineering

Algorithm for False Position Method

1. start

2. Define function f(x)

3. Choose initial guesses x0 and x1 such that f(x0)f(x1) < 0

4. Choose pre-specified tolerable error e.

5. Calculate new approximated root as:

   x2 = x0 - ((x0-x1) * f(x0)) / (f(x0) - f(x1))

6. Calculate f(x0)f(x2)
   - a. if f(x0)f(x2) < 0 then x0 = x0 and x1 = x2
   - b. if f(x0)f(x2) > 0 then x0 = x2 and x1 = x1
   - c. if f(x0)f(x2) = 0 then goto (8)

7. if |f(x2)|>e then goto (5) otherwise goto (8)

8. Display x2 as root.

9. Stop

Algorithm: Secant Method

1. Start

2. Define function as f(x)

3. Input initial guesses (x0 and x1),
   tolerable error (e) and maximum iteration (N)

4. Initialize iteration counter i = 1

5. If f(x0) = f(x1) then print "Mathematical Error"
   and goto (11) otherwise goto (6)

6. Calculate x2 = x1 - (x1-x0) * f(x1) / (f(x1) - f(x0))

7. Increment iteration counter i = i + 1

8. If i >= N then print "Not Convergent"
   and goto (11) otherwise goto (9)

9. If | f(x2) | > e then set x0 = x1, x1 = x2
   and goto (5) otherwise goto (10)

10. Print root as x2

11. Stop

Algorithm for Newton Raphson Method

1. Start

2. Define function as f(x)

3. Define first derivative of f(x) as g(x)

4. Input initial guess (x0), tolerable error (e)
   and maximum iteration (N)

5. Initialize iteration counter i = 1

6. If g(x0) = 0 then print "Mathematical Error"
   and goto (12) otherwise goto (7)

7. Calculate x1 = x0 - f(x0) / g(x0)

8. Increment iteration counter i = i + 1

9. If i >= N then print "Not Convergent"
   and goto (12) otherwise goto (10)

10. If |f(x1)| > e then set x0 = x1
    and goto (6) otherwise goto (11)

11. Print root as x1

12. Stop

Algorithm for Gauss Elimination Method

1. Start

2. Read Number of Unknowns: n

3. Read Augmented Matrix (A) of n by n+1 Size

4. Transform Augmented Matrix (A)
   to Upper Trainagular Matrix by Row Operations.

5. Obtain Solution by Back Substitution.

6. Display Result.

7. Stop

Algorithm for Gauss Jordan Method

1. Start

2. Read Number of Unknowns: n

3. Read Augmented Matrix (A) of n by n+1 Size

4. Transform Augmented Matrix (A)
   to Diagonal Matrix by Row Operations.

5. Obtain Solution by Making All Diagonal Elements to 1.

6. Display Result.

7. Stop

Algorithm for Power Method

1. Start

2. Read Order of Matrix (n) and Tolerable Error (e)

3. Read Matrix A of Size n x n

4. Read Initial Guess Vector X of Size n x 1

5. Initialize: Lambda_Old = 1

6. Multiply: X_NEW = A * X

7. Replace X by X_NEW

8. Find Largest Element (Lamda_New) by Magnitude from X_NEW

9. Normalize or Divide X by Lamda_New

10. Display Lamda_New and X

11. If | Lambda_Old - Lamda_New | > e then
    set Lambda_Old = Lamda_New and goto
    step (6) otherwise goto step (12)

12. Stop

Gauss Seidel Iterative Method Algorithm

1. Start

2. Arrange given system of linear equations in diagonally dominant form

3. Read tolerable error (e)

4. Convert the first equation in terms of first variable, second equation in terms of second variable and so on.

5. Set initial guesses for x0, y0, z0 and so on

6. Substitute value of y0, z0 ... from step 5 in first equation obtained from step 4 to calculate new value of x1. Use x1, z0, u0 .... in second equation obtained from step 4 to calculate new value of y1. Similarly, use x1, y1, u0... to find new z1 and so on.

7. If | x0 - x1 | > e and | y0 - y1 | > e and | z0 - z1 | > e and so on then goto step 9

8. Set x0=x1, y0=y1, z0=z1 and so on and goto step 6

9. Print value of x1, y1, z1 and so on

10. Stop

Algorithm: Lagrange Interpolation Method

1. Start

2. Read number of data (n)

3. Read data X_i and Y_i for i=1 ton n

4. Read value of independent variables say xp
   whose corresponding value of dependent say yp is to be determined.

5. Initialize: yp = 0

6. For i = 1 to n
     Set p = 1
     For j =1 to n
       If i ≠ j then
         Calculate p = p * (xp - X_j) / (X_i - X_j)
       End If
     Next j
     Calculate yp = yp + p * Y_i
   Next i

6. Display value of yp as interpolated value.

7. Stop

Linear Regression Algorithm (Fitting y = a + bx)

1. Start

2. Read Number of Data (n)

3. For i=1 to n:
    Read X_i and Y_i
   Next i

4. Initialize:
    sumX = 0
    sumX2 = 0
    sumY = 0
    sumXY = 0

5. Calculate Required Sum
   For i=1 to n:
    sumX = sumX + X_i
    sumX2 = sumX2 + X_i * X_i
    sumY = sumY + Y_i
    sumXY = sumXY + X_i * Y_i
   Next i

6. Calculate Required Constant a and b of y = a + bx:
   b = (n * sumXY - sumX * sumY) / (n*sumX2 - sumX * sumX)
   a = (sumY - b*sumX) / n

7. Display value of a and b

8. Stop

Trapezoidal Method Algorithm

1. Start
2. Define function f(x)
3. Read lower limit of integration, upper limit of integration and number of sub interval
4. Calculate: step size = (upper limit - lower limit) / number of sub interval
5. Set: integration value = f(lower limit) + f(upper limit)
6. Set: i = 1
7. If i > number of sub interval then goto
8. Calculate: k = lower limit + i * h
9. Calculate: Integration value = Integration Value + 2* f(k)
10. Increment i by 1 i.e. i = i+1 and go to step 7
11. Calculate: Integration value = Integration value * step size / 2
12. Display Integration value as required answer
13. Stop

Simpson's 1/3 Rule Algorithm

1. Start

2. Define function f(x)

3. Read lower limit of integration, upper limit of
   integration and number of sub interval

4. Calcultae: step size = (upper limit - lower limit) / number of sub interval

5. Set: integration value = f(lower limit) + f(upper limit)

6. Set: i = 1

7. If i > number of sub interval then goto

8. Calculate: k = lower limit + i * h

9. If i mod 2 =0 then
     Integration value = Integration Value + 2* f(k)
   Otherwise
     Integration Value = Integration Value + 4 * f(k)
   End If

10. Increment i by 1 i.e. i = i+1 and go to step 7

11. Calculate: Integration value = Integration value * step size / 3

12. Display Integration value as required answer

13. Stop

Euler's Method Algorithm (Ordinary Differential Equation)

1. Start

2. Define function f(x,y)

3. Read values of initial condition(x0 and y0),
   number of steps (n) and calculation point (xn)

4. Calculate step size (h) = (xn - x0) / b

5. Set i=0

6. Loop

    yn = y0 + h * f(x0 + i*h, y0)

    y0 = yn

    i = i + 1

   While i < n

7. Display yn as result

8. Stop

Ordinary Differential Equation Using Fourth Order Runge Kutta (RK) Method Algorithm

```
1. Start

2. Define function f(x,y)

3. Read values of initial condition(x0 and y0),
   number of steps (n) and calculation point (xn)

4. Calculate step size (h) = (xn - x0) / n

5. Set i=0

6. Loop
        k1 = h * f(x0, y0)
    k2 = h * f(x0+h/2, y0+k1/2)
    k3 = h * f(x0+h/2, y0+k2/2)
    k4 = h * f(x0+h, y0+k3)

    k = (k1+2*k2+2*k3+k4)/6

    yn = y0 + k

    i = i + 1
    x0 = x0 + h
    y0 = yn

  While i < n

7. Display yn as result

8. Stop
```