Table Of Contents For Algorithm

1. Calculate missing value

Algorithm:

1.start.

2.Find the number of values goven for f(x)

3.We know that $\Delta \equiv (E-1)$.Here we have to putten the value of Δ from step-1.

4.If there is one value missing then we have to put one number or if there is two values missing then we have to put two number in y.We put value into x for generating new euation.

5.Now solve the equation for getting new values.

6.End.

2. Bisection Method

Algorithm:

1.Start

2.Read a1, b1, TOL

*Here a1 & b1 are initial guesses

TOL is the absolute error or tolerance i.e. the desired degree of accuracy*

3. Compute: f1 = f(a1) & f3 = f(b1)

4. If (f1*f3) > 0, then display initial guesses are wrong and go to step 11 otherwise continue.

5. root = (a1 + b1)/2

6. If [(a1- b1)/ root] < TOL, then display root and go to step 11

* Here [ ] refers to the modulus sign.*

7. Else, f2 = f(root)

8. If (f1*f2) < 0, then b1= root

9. Else if (f2*f3) < 0 then a1 = root

10. else go to step 5

Now the loop continues with new values

11. Stop

3.Regular Falsi Method:

Algorithm:

1.Start

2. Read values of x0, x1 and

   *Here x0 & x1 are the two initial guesses

3. Computer function values f(x0) and f(x1)

4. Check whether the product of f(x0) and f(x1) is negative or not.

   If it is positive take another initial guesses

   If it is negative then go to step 5

5. Determine:

   X= [x0*f(x1) – x1*f(x1) – f(x0)] / (f(x1) - f(x0))

6.check whether the product of f(x1) and f(x) is negative or not.

   If it is negative, then assign x0 =x;

   If it is positive, assign x1 = x;

7. Check whether the value of f(x) is greater than 0.00001 or not

   If yes, go to step 5.

   If no, go to step 8.

8. Display the root as x

9. Stop

4. Newton-Rapshon Method:

Algorithm:

1. Start
2. Read x, e, n, d

   *x is the initial guess

   e is the absolute error i.e the desired degree of accuracy

   n is for operating loop

   d is for checking slope*
3. Do for i =1 to n in step of 2
4. f = f(x)
5. f1 = f'(x)
6. If ([f1] < d), then display too small slope and go to 11.

   [ ] is used as modulus sign
7. x1 = x – f/f1
8. If ([(x1 – x)/x1] < e), the display the root as x1 and go to 11.

   [ ] is used as modulus sign
9. x = x1 and end loop
10. Display method does not converge due to oscillation.
11. Stop

5. Newton Forward Interpolation:

Algorithm:

1. Start
2. Declare x[20], y[20], f, s, d, h, p as float data type and I, j, n as integer data type
3. Read the record n and read the elements of x & y using for loop
4. Calculate h = x[2] – x[1]
5. Read the point which is going to be searched
6. Calculate s = (f – x[1]/h)

   p = 1

   d = y[1]
7. Using for loop calculate p & d
   a. y[j] = y [j+ 1] – y [j-1]
   b. p = p *(s* I + 1) / i
   c. d = d + p * y[1]
8. print f & d
9. Stop

6. Newton Backward Interpolation:

Algorithm:

1. Start

2. Declare x[20], y[20], f, s, d, h, p as float data type and I, j, n as integer data type

3. Read the record n and read the elements of x & y using for loop

4. Calculate h = x[2] – x[1]

5. Read the point which is going to be searched

6. Calculate s = (f – x[n]/h),

   d = y[n],

   p = 1

7. Using for loop calculate f & d

   a. y[j] = y [j] – y [j-1]

   b. p = p *(s* k - 1) / k

   c. d = d + p * y[n]

8. print f & d

9. Stop

7. Lagrange's Formula:

Algorithm:

1. Start
2. Input number of terms n
3. Input the array ax
4. Input the array ay
5. For i=0; i<n; i++
6. nr=1
7. dr=1
8. for j=0; j<n; j++
9. if j !=ia. nr=nr*(x-ax[j])
10. b.dr*(ax[i]-ax[j])
11. End Loop j
12. y+=(nr/dr)∗ay[i]
13. End Loop i
14. Print Output x, y
15. Stop

9. Trapezoidal Rule

Algorithm:

1.start.

2. input a, b, number of interval n

3. h = (b- a) / f(b)

4. sum = f(a) + f(b)

5. If n = 1, 2 , 3,………..i

    Then, sum = sum + 2*y(a+i*h)

6. Display output = sum *h/2

7.stop.

10. Simpson's 1/3 Rule:

Algorithm:

1.start.

2. input a, b, number of interval n

3. h = (b-a) /n

4. sum = f(a) + f(b) + 4*f(a+h)

5. sum = sum + 4*f(a+i*h) + 2*f(a + (i-1)*h)

6. Display output = sum *h/3

7.Stop

11. Simpson's 3/8 Rule:

Algorithm:

1.start

1. input a, b, number of interval n
2. h = (b- a) / n
3. sum = f(a) + f(b)
4. If n is odd

   Then, sum = sum + 2*y(a+i*h)
5. else, when n is even

   Then, sum = sum + 3*y(a+i*h)
6. Display output = sum *3*h/8

7.Stop

12. Weddle's Rule:

Algorithm:

1. input a, b, number of interval n
2. h = (b- a) / n
3. If(n%6==0 Then,

   Sum = sum + ((3*h/10)*(y(a) + y(a +2*h) +5*y(a+h) + 6*y(a+3*h) + y(a+4*h) +5*y(a+5*h) +y(a +6*h)));

   a = a+6*h

   and Weddle's rule applicable then go step 6
4. else, Weddle's rule id not applicable
5. Display output

13. Picard's method:

Algorithm:

1.inputs

1.Given differential equation in the form of a function as (dy/dx) = f(x, y)

2.Initial value of function, that is, f(xo) = yo, where xo is initial value of x.Read xo and yo

3.End value of x, that is xn and number of iterationsn

4.Read xn and n

5.Read allowed error aerror

2. set x <= x0 and yi<= y0

3.calculate step size h <=(xn-x0)

4. print xo, " ", yo

5. for i = 1 to n do

    yn = yi+ h*f(xo,yo)

    print xn, " ", yn

    if abs(yn-yo) < aerror then stop

    yo <= yn

end of iloopprint " Maximum number of iterations reached"

6. stop

14. Euler's Method:

Algorithm:

Input:

Step 1-

(1) Given differential equation in the form of a function as (dy/dx) = f(x, y)

(2) Initial value of function, that is, f(xo) = yo, where xo is initial value of x.

(3) End value of x, that is xn and number of steps n

Step 2. set x <= x0 and y <= y0

Step 3. calculate step size h <= (xn-x0)/n

Step 4. print xo, " ", yo

Step 5. for i = 1 to n do

yn = yo + h*f(xo,yo)

xo <= xo+h

yo <= yn

print xo, " ", yo end of i loop

step 6. stop