Algorithm of Bisection Method

1. Define function $f(x)$ and specified Error (ϵ)
2. Input x_0 & x_1, such that:
   $f(x_0) * f(x_1) < 0$
   else
   Print message - input error
   and go to step 2.
3. Compute new approximated root $x = \frac{x_0 + x_1}{2}$
4. Compute error of computation
   error = $|x_1 - x_0|$
5. If $(f(x_0) * f(x) < 0)$, then replace x_1 by x
   i.e. $x_1 = x$
   else
   Replace x_0 by x
   i.e. $x_0 = x$
6. If error $> \epsilon$, then go to step 3
   else
   Print x as a root.

Algorithm of Regular Falsi Method

1. Define function $f(x)$ and specified error (ε)
2. Input the two initial guesses such that the interval $[x_0, x_1]$ consists real root, i.e.

$$f(x_0) * f(x_1) < 0$$

else

print message-input error
and go to step 2.

3. Compute the approximiated root,

$$x_2 = x_0 - \frac{f(x_0)\,(x_1 - x_0)}{f(x_1) - f(x_0)}$$

and error of computation

$$\text{error} = \text{fabs}\,(f(x_2))$$

4. if $f(x_2) * f(x_0) < 0$, then replace x_1 by x_2
   i.e. $x_1 = x_2$
   else
   $x_0 = x_2$

5. if (error $> \varepsilon$), then go to step 3
   else
   print x_2 as a root.

6. End

Algorithm of Newton Raphson Method

1. Define function $f(x)$, $f'(x)$ and specified error (ε)
2. Input x_0 as initial guess such that $f'(x_0) \neq 0$
   else
   Print message - input error
   and goto step 2.
3. Compute the next approximiated root
   $$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$
   and also the error of computation
   $$\text{error} = \left| \frac{x_1 - x_0}{x_1} \right|$$
4. Replace x_0 by x_1
5. If $f'(x_0) = 0$, then print message
   there is no convergence of root and exit from the program
   else
   continue
6. If error > specified error, then goto step 3.
   else,
   print x_1 as root
7. End

$$x_3 = x_2 - \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)}$$

$$\therefore x_3 = x_2 - \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)}$$

This is called as approximiate formula for secant method.

Algorithm: Secant Method

1. Define function $f(x)$ and specified error (ε)
2. Take two initial guesses x_0 and x_1.
3. Compute next approximiated root and error

$$x_3 = x_2 - \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)}$$

$$\text{error} = \left| \frac{x_3 - x_2}{x_3} \right|$$

4. Replacement $x_1 = x_2$

$x_2 = x_3$

5. If error $> \varepsilon$, then go to step 3
   else print x_3 as a root.
6. End

Where,

x_0 be the initial guesses, then approximated root can be evaluated as

$$x_1 = g(x_0)$$

Similarly,

$$x_2 = g(x_1)$$
$$x_3 = g(x_2)$$
$$\vdots$$
$$x_{i+1} = g(x_i)$$

$\boxed{x_{i+1} = g(x_i)}$ This is the next approximated root expression for fixed point iteration method

Algorithm: Fixed Point

1. Define function $g(x)$ and specified error ε.
2. Take x_0 as initial guesses
3. Compute next approximated root and error as

$$x_1 = g(x_0)$$
$$\text{error} = \left| \frac{x_1 - x_0}{x_1} \right|$$

4. Replace x_0 by x_1
5. If (error $> \varepsilon$), then go to step 3
   else print x_1 as result.
6. End

Find the value of $f(x) = x^3 + x^2 - x + 1$ at $x = 1$ using Horner's formula.

Here,

$x_0 = 1$, $a_2 = 0$, $a_0 = 1$, $a_3 = 1$, $a_1 = -1$

we have,

$b_3 = a_3 = 1$

$b_2 = a_2 + b_3 x_0$

$= 0 + 1 * 1$

$= 1$

$b_1 = a_1 + b_2 x_0$

$= -1 + 1 * 1$

$= 0$

~~$b_0 = a_0 + b_1 x_0$~~

$b_0 = a_0 + b_1 x_0$

$= 1 + 0 * 1$

$= 1$

Algorithm: Horners

1. Input the order of polynomial say n and point of evaluation say x_0
2. Read the point of coefficients of polynomials.
   for i = 0 to n
   Read a[i]
3. Set b[n] = a_n
4. Do following operation for n >= 0
   b[n-1] = a[n-1] + $b_n * x_0$
5. Print b_0 as a functional value
6. End

$$= \frac{(5-2)(5-3)(5-4)}{(6-2)(6-3)(6-4)}$$

$$= \frac{3 \times 2 \times 1}{4 \times 3 \times 2}$$

$$= \frac{6}{24} = \frac{1}{4}$$

Then from relation (1),

$$f(5) = \frac{1}{4} \times 40 - 1 \times 55 + \frac{3}{2} \times 68 + \frac{1}{4} \times 78$$

$$= 10 - 55 + 102 + 19.5$$

$$= 76.5$$

Algorithm Lagrange

1. Input the no. of set of date say 'n'
2. Input the data set

```
for i:=0 upto n
    Read x[i]
    Read y[i]
```

3. Read the point of valuation say (p)
4. Compute lagrange coefficients as below:

```
for i = 0 upto n {
        term = 1
        for j = 0 upto n {
            if (j ≠ i), then
                term = term * (p - x[j]) / (x[i] - x[j])
        }
        l[i] = term;
}
```

5. Compute interpolation value

```
sum = 0
for i = 0 upto n {
        sum = sum + l[i] * y[i]
}
```

6. Print sum as required solution

7. End

Linear Regression

Algorithm

1. Input the no. of set of data say(n)
2. Read the data as

```
for i=0 upto n
    Read x[i]
    Read y[i]
end for
```

3. Set sumx = 0, sumk = 0, sumxy = 0, sumx2 = 0
4. Compute the values as:

```
for i=0 upto n
    sumx = sumx + x[i]
    sumy = sumy + y[i]
    sumxy = sumxy + x[i] * y[i]
    sumx2 = sumx2 + x[i] * x[i]
```

5. Compute the regression coefficients as a & b

$$b = (n * sumxy - sumx * sumy) / (n * sumx2 - sumx * sumx)$$

$$a = ((sumy) - (b * sumx)) / n;$$

6. Print y = a + bx as fit line.
7. End #

Banday Value Problem

Numerical Differentiation using Three Point ~~Forward~~ Formula

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

Algorithm:

1. Define $f(x)$, $h = 0.001$
2. Input the point of evaluation, p
3. Compute $f1 = |f(p+h) - f(p-h)| / h$

   $$f2 = \frac{f(p+h) - 2f(p) + f(p-h)}{h*h}$$

4. Print $f1, f2$ as first derivative & second derivative
5. End

Trapezoidal Rule:

Algorithm:

1. Define the function $f(x)$
2. Read the lower limit say (a), upper limit say (b) and no. of segments (n).
3. Compute $h = \left(\frac{b-a}{n}\right)$
4. Set $IV = f(a) + f(b)$, $x[0] = a$;
5. Compute the function values
   for i = 1 upto n
   $x[i] = x[i-1] + h$
   $y[i] = f(x[i])$
   end for
6. Compute integral value as:

1/3

```
for i = 1 upto n
    IV = IV + 2 * y[i]
end for
IV = IV * h/2
```

```
for i = 1 upto n
    if (i % 2 == 0)
        IV = IV + 2 * y[i]
    else
        IV = IV + 4 * y[i]
end for
IV = IV * h/3
```

3/8

```
for i = 1 upto n
    if (i % 3 == 0)
        IV = IV + 2 * y[i]
    else
        IV = IV + 3 * y[i]
end for
IV = 3/8 * IV
```

7. Print IV as Result
8. End

Solution of ODE

Algorithm:

1. Define function $f(x, y)$
2. Read initial conditions $x_0 = a$ & $y_0 = b$
3. Read the step size (say h) and point of evaluation say (p).
4. Compute the solution:

Euler's

while ($x_0 < p$) {
Compute
$m = f(x_0, y_0)$
$x_0 = x_0 + h$
$y_0 = y_0 + m * h$
}

R.K

while ($x_0 < p$) {
$m_1 = f(x_0, y_0)$
$m_2 = f(x_0 + h/2, y_0 + m_1 * h/2)$
$m_3 = f(x_0 + h/2, y_0 + m_2 * h/2)$
$m_4 = f(x_0 + h, y_0 + m_3 * h)$
$m = \left(\frac{m_1 + 2m_2 + 2m_3 + m_4}{6}\right)$
$x_0 = x_0 + h$
$y_0 = y_0 + m * h$

5. Print y_0 as root
6. End

Algorithm:

Read the dimension of system of equations (say n), Coefficients of matrix row wise, elements of RHS vector, accuracy limit (error)

Set initial guesses as:

```
for i=1 upto n
    new_x[i]=0
end for
```

Compute the following

```
for i=1 upto n
  sum = b[i]
  for j=1 upto n
    if (i≠j)
      sum = sum - a[i][j] * new_x[j]
  end for
  old_x[i] = new_x[i]
  new_x[i] = sum/a[i][i]
  E[i] = | (new_x[i] - old_x[i]) / new_x[i] |
end for
```

Check for error level

```
for i=1 upto n
  if (E[i] > error)
    go to step 3
end for
```

Display new_x vector as result.

End