**Report Numerical**

**“Root Finder Program”**

**Pseudo-code for each method :**

**Biserction :**

Given a , b // two initial guesses

if f(a)\*f(b) >0

return error ;

else

xr = (a+b)/2

calculate error ;

while error > tolerance and i < maxIteration

then repeate calculate the approx. root , update the upper and lower points and calculate the error.

return root ;

**FalsePosition:**

Given a , b // two initial guesses

if f(a)\*f(b) >0

return error ;

else

xr = b – (f(b)(b-a)/(f(b)-f(a)));

calculate error ;

while error > tolerance and i < maxIteration

then repeate calculate the approx. root , update the upper and lower points and claculate the error.

return root ;

**FixedPoint:**

Given x0 , g(x) // one initial guess , iterated function

xi = g(x0);

calculate error ;

while error > tolerance and i < maxIteration

then repeate calculate the approx. root , update the initial point for each iteration and claculate the error.

return root ;

**NewtonRaphson:**

Given x0 // one initial guess

Find the derivative of f(x) as f `(x)

xi = x0-f(x)/f `(x);

calculate error ;

while error > tolerance and i < maxIteration

then repeate calculate the approx. root , update the initial point for each iteration and claculate the error.

return root ;

**Secant:**

Given xi , xi-1 // two initial guesses

xi+1 = xi – (f(xi)(xi-xi-1)/(f(xi)-f(xi-1)));

calculate error ;

while error > tolerance and i < maxIteration

then repeate calculate the approx. root

xi = xi+1;

xi-1 = xi;

and calculate the error again;

return root ;

**Data structure used:**

Matrices are used for different purposes such as collecting the approximated root points to plot them on the graph.

**Analysis for the behavior of different examples:**

First example:

F(x) = x^3+3\*x^2+12\*x+8 , and given interval : [-2 ,3] and tolerance = 0.0001

In case of Bisection , it will take 16 iterations to reach the root.

In case of FalsePosition , it will take 21 iterations to reach the root.

In case of fixedPoint , it will take 8 iterations to reach the root.

In case of NewtonRaphson , it will take 4 iterations to reach the root.

In case of Secant , it will take 6 iterations to reach the root.

So the NewtonRaphson will converge faster.

Second example:

F(x) = 3\*x^4+6.1\*x^3-2\*x^2+3\*x+2 , and given interval : [-1 ,0] and tolerance = 0.0001

In case of Bisection , it will take 8 iterations to reach the root.

In case of FalsePosition , it will take 6 iterations to reach the root.

In case of fixedPoint , it will diverge for this initial guess.

In case of NewtonRaphson , it will take 4 iterations to reach the root.

In case of Secant , it will take 5 iterations to reach the root.

So the NewtonRaphson will converge faster.

Third example:

F(x) = e^x+x^2-x-4 , and given interval : [-3 ,0] and tolerance = 0.0001

In case of Bisection , it will take 15 iterations to reach the root.

In case of FalsePosition , it will take 9 iterations to reach the root.

In case of fixedPoint , it will diverge for this initial guess.

In case of NewtonRaphson , it will take 5 iterations to reach the root.

In case of Secant , it will take 7 iterations to reach the root.

So the NewtonRaphson will converge faster.

**Problematic functions:**

First:

F(x) = x^3-0.165\*x^2+3.993\*10^-4 , this function may lead to problem if we start with initial guess 0.11 or 0 because this will lead to devision by 0 if we use newton raphson where at this points the tangent to the function will be zero and we can avoid this problem by choose another initial guess.

Second:

F(x) = sin(x) , this function may lead to problem if we start with for an example 2.4π as an initial guess with newton raphson because this will lead to convergence to the root that is far from the initial guess instead of convergence to the close root from initial guess and to avoid this problem we can choose another initial guess that is more close to the root .

Third:

F(x) =(x-1)^3+0.512 , this function may lead to problem because this function has an inflection point at x = 1 that is close to the root x = 0.2 , so by using newton raphson , at certain iteration which the approximated root is close to an inflection point , this method will diverge and after more iterations it will converge again .

**Sample runs:**













