Numerical Computations Project Phase#2

Fall Semester 2021

Veronica Romany Hanna 19016156

Norah Ahmed Hassan 19016810

Mariam Tarek Mostafa 19016637

Mariam Mohamed Ahmed 19017274

Yara Hossam Abdelaziz 19016871

Problem Statement

- Implementation of a program which takes an input of a system of linear equations and solves it using the desired method.
- Compare and analyze the behavior of the different numerical methods used for calculating roots of equations:
    1. Bisection.
    2. False Position.
    3. Fixed Point.
    4. Newton Raphson.
    5. Secant Method.

Design Decisions

1. The program is written in Python.
2. The equations entered don't contain an equal sign. (e.g "exp(-x) -x" is valid)
3. Invalid input format generates an error.
4. Users can set the decimal rounding value. If not chosen, the default value is 4.
5. If not chosen by the user, default Epsilon = 0.00001 and default Max Iterations = 50
6. Input criteria:
    a. 5x as 5*x
    b. e^x as exp(x).
    c. sinx as sin(x)

d. Power x^3 as x^3

Pseudocode for each method

1. Bisection

Bisection:

Start

GET xl,xu,e , function,noofiterations,significatnnum

If (function(xu)*function(xl))

    Return “ Given values do not bracket the root.”

End if

Step→1

condition→true

while (condition or step< noofiterations)

    xr→(xu+xl)/2

    if(function(xl)*function(xr)<0.0)

        xu→xr

    else

        xl→xr

    step→step+1

    condition→abs(evaluate(xr, function)) > e

        Return xr

2. False Position

False Position

•Start

•GET xl, xu, function, tol, maxIterations .

```
Declare xold ← 0
If (function(xl)*function(xu) >0)
    Return "Given values do not bracket the root."
End if
For i = 0→ maxIterations
       xr ← (xl*function(xu) - xu*function(xl))/(function(xu) – function(xl))
      if (function(xr)==0)
            Return xr
      if function(xr) * function(xl )< 0:
            xu ← xr
       else:
           xl ← xr
       if i > 0:
             if(abs(xr - xold)< tol): break
       xold ← xr
  Return xr
```

3. Fixed Point

```
Input function ,initial value ,Es ,Max_iterations ,significant

Output Xr and runtime

Initially set Xr equals to initial value

Set iterations initially to zero

StartTime=time()

While True

        Try:

                 Let Xr_old=Xr

                 Xr=Substitute in f(x) with Xr_old +Xr_old

                 If Xr not equal zero

                         Ea =(Xr_old- Xr /Xr_old) *100

                End If

                 Increment iterations

                Endtime=time()

                Total_time=Endtime –StartTime

                If (Ea <Es or Max_iteartions < iterations)

                         Xr=round(Xr ,significant)

                         Then return Xr , Total_time
```

```
        End If
        Except:
                Then return error
End While
```

4. Newton Raphson

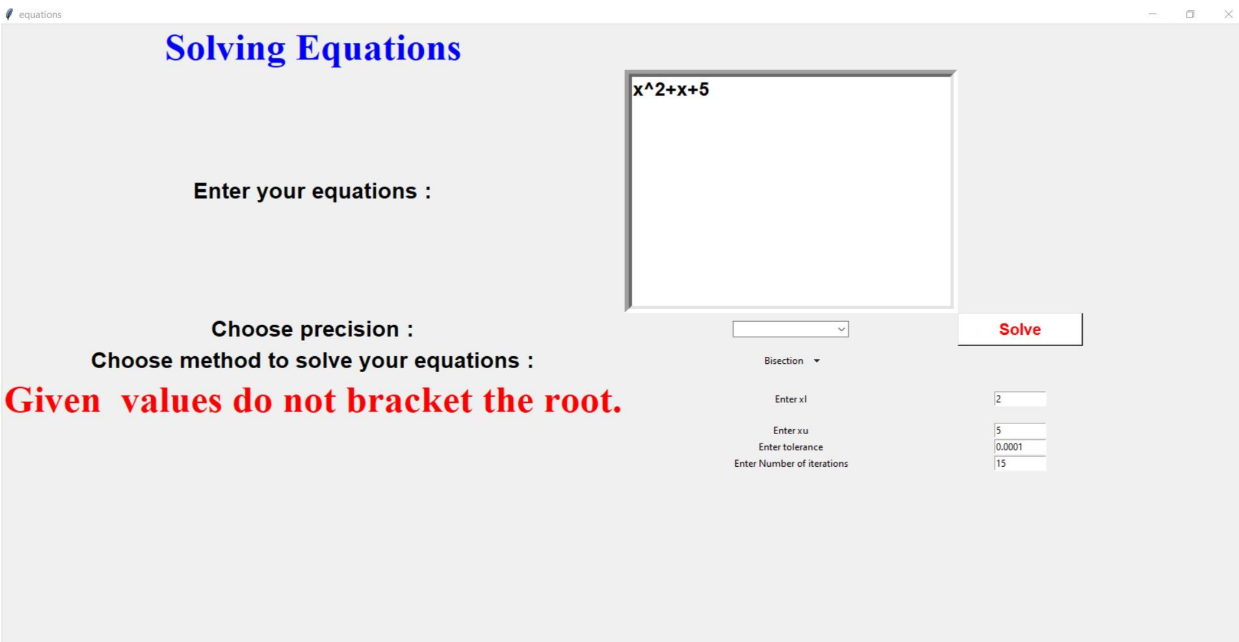
```
NewtonRaphson(function,intialGuess,MaxInterations,precision,Es):
Ea=0
i=1
Xi=intialGuess
Xi+1=0
While (Ea<Es and i<MaxIteration){
        Xi+1 = round(Xi – f(Xi)/f'(Xi),precision)
        If(i!=1){
                Ea=abs(Xi+1-Xi)/Xi+1 * 100
        }
        Xi=Xi+1
        i++
}
```

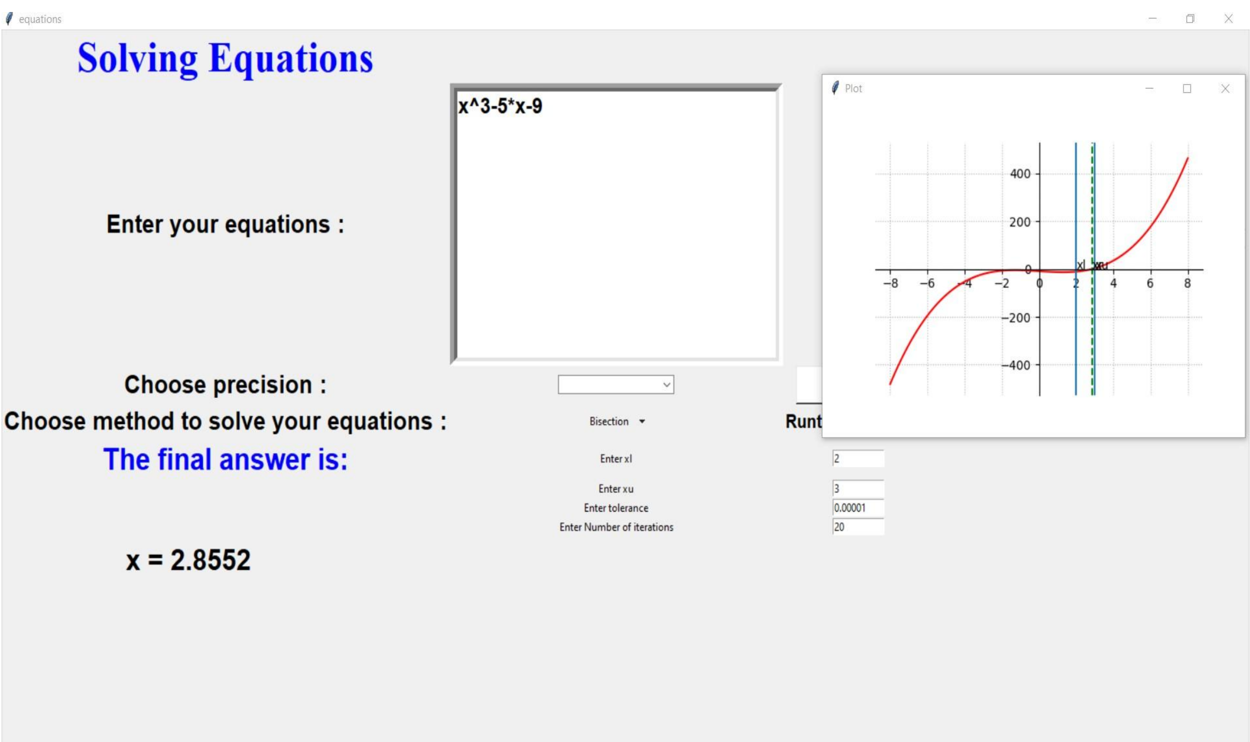
5. Secant Method

```
Function secant(equation, x1, x2 ,maxIterations, tolerance):
  For i = 1 to maxIterations:
        x = x2
        Fnx2 = evaluation(equation(x))
        x = x1
        Fnx1 = evaluation(equation(x))
        x3 = x2 - ((fnx2 * (x1-x2)) / (fnx1 - fnx2))
        If ((x3-x2) / x3) < tolerance:
                Return x3
                End If
        x1 = x2
        x2 = x3
End For
Return x3
```

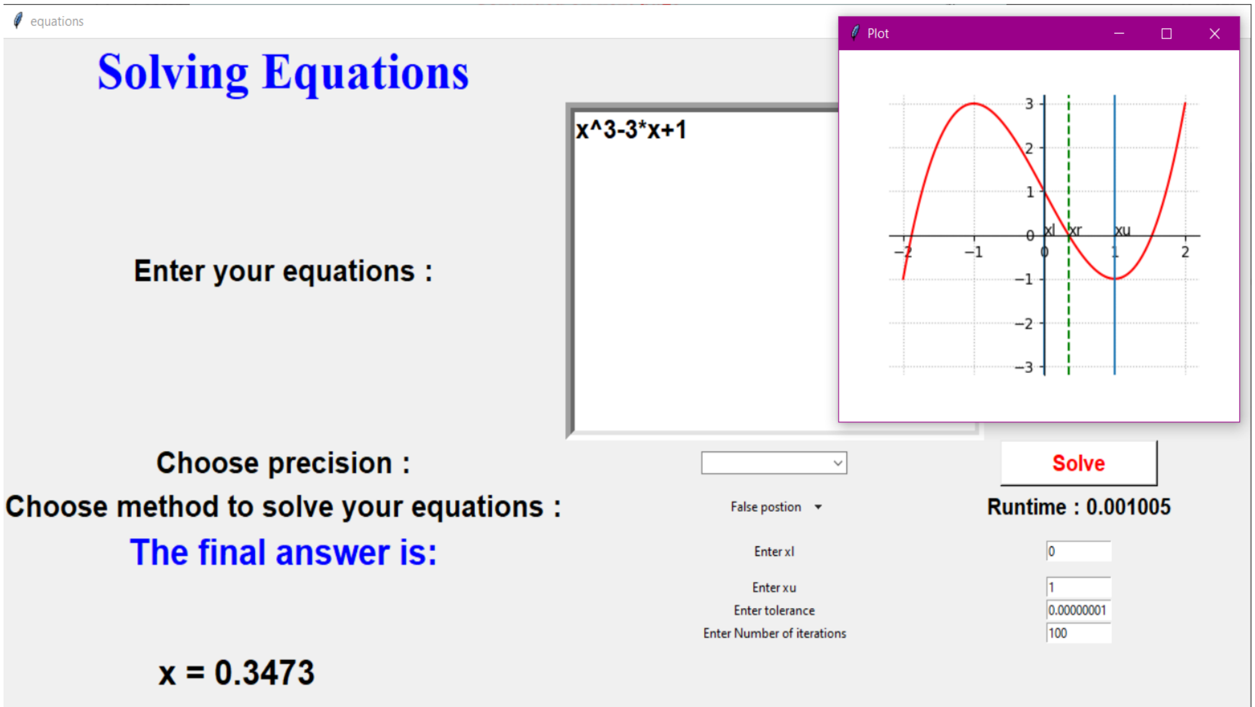
Sample runs

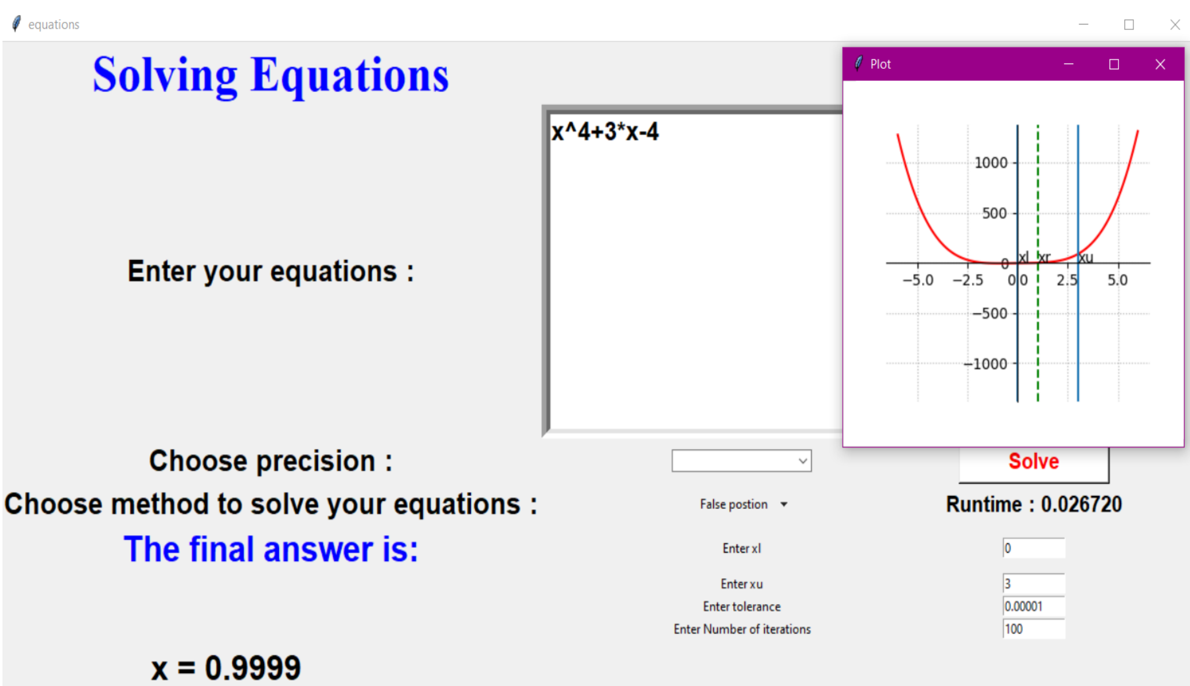
1. Bisection

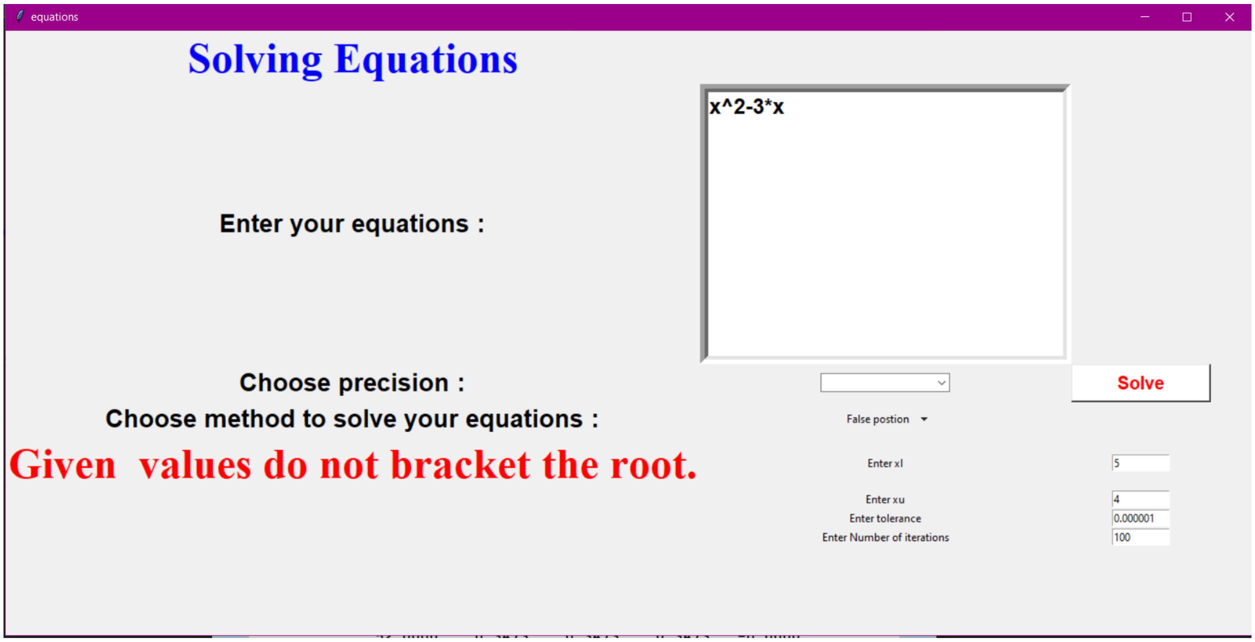




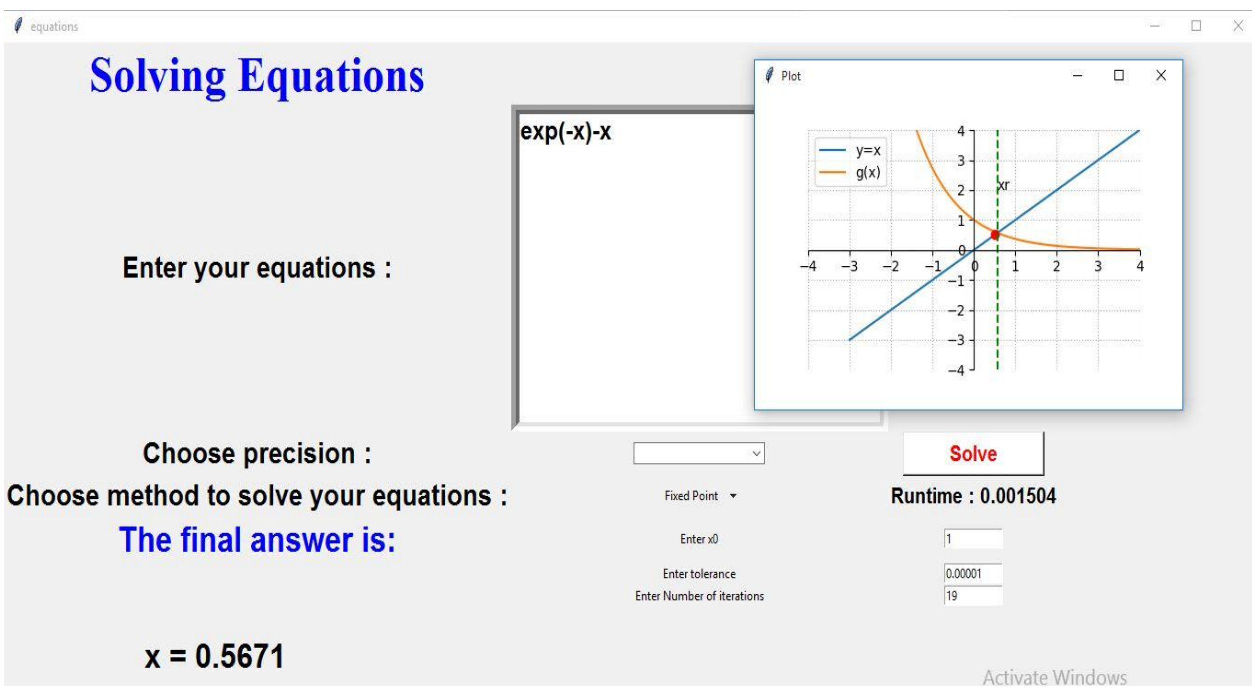
2. False Position

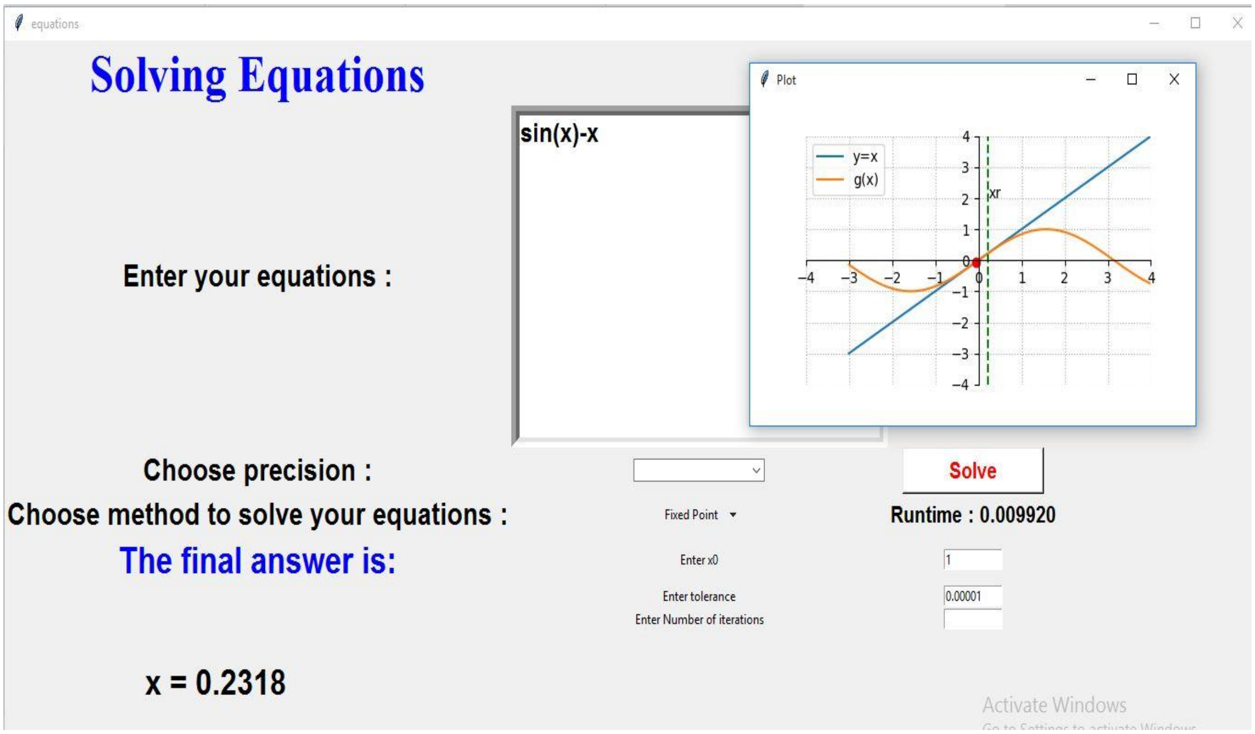


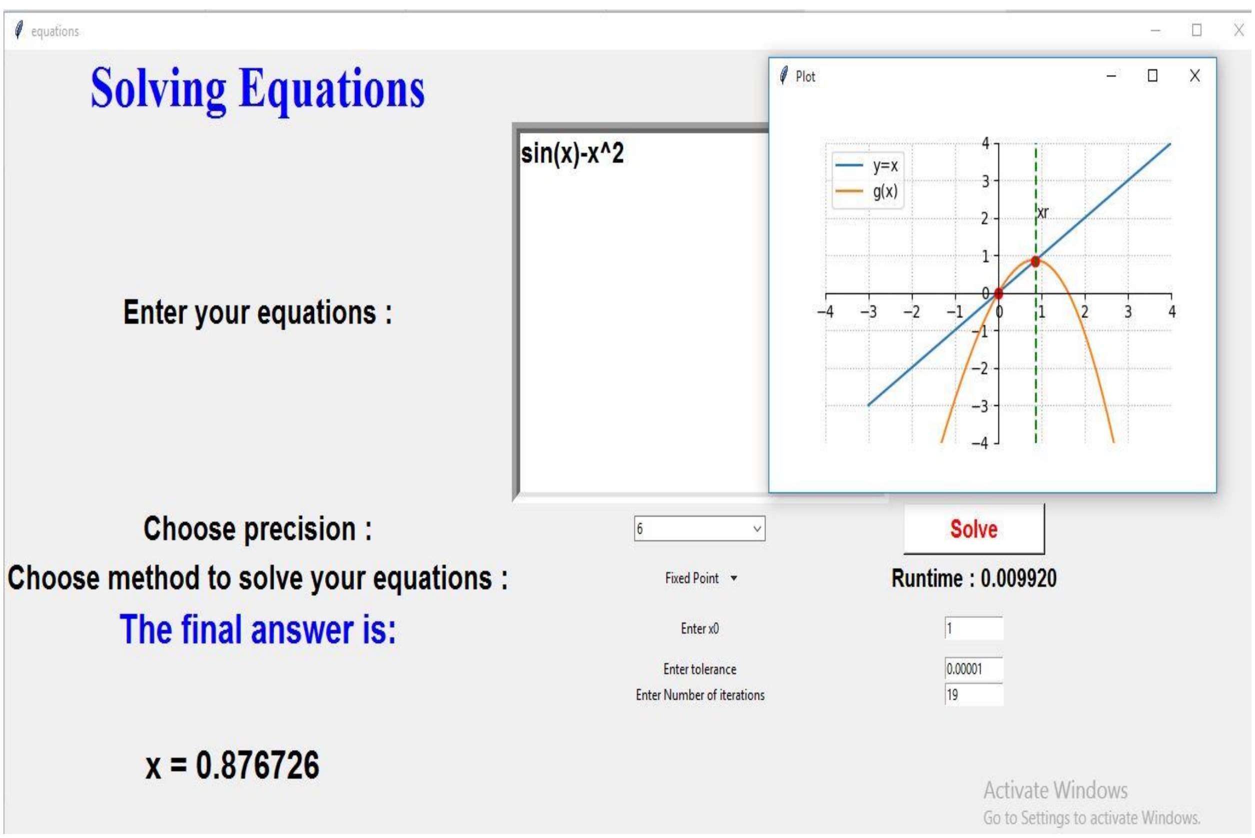


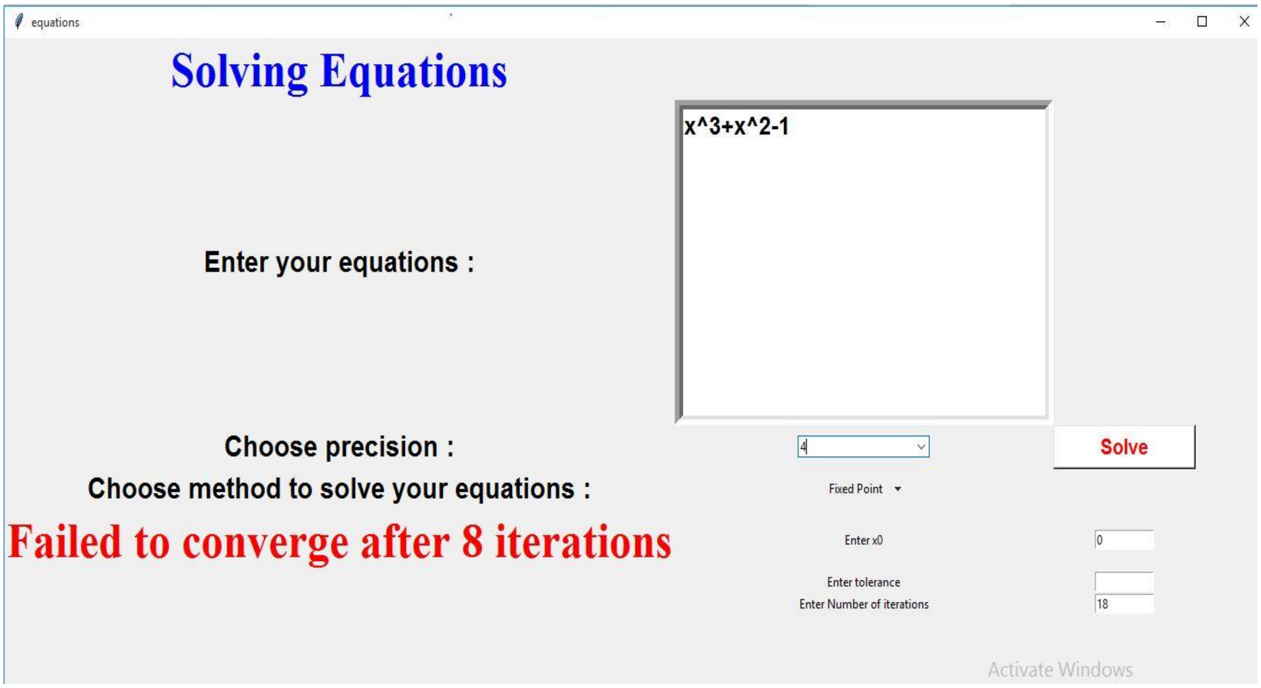


3. Fixed Point

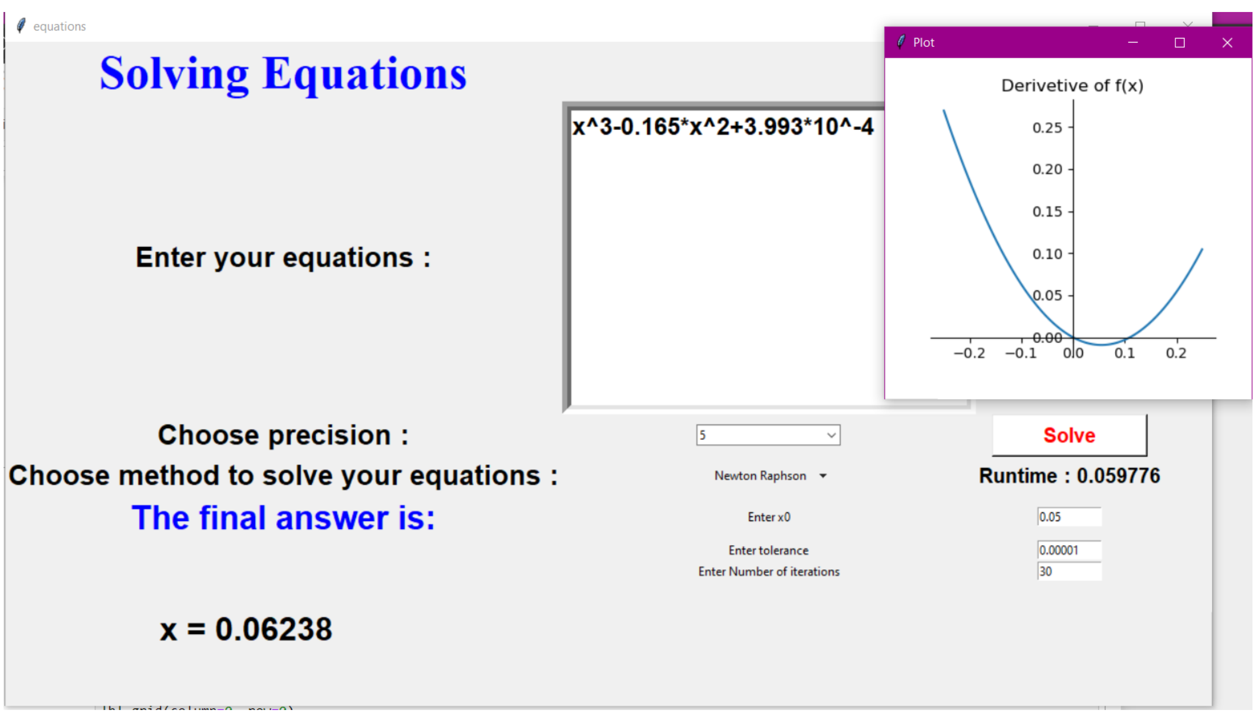


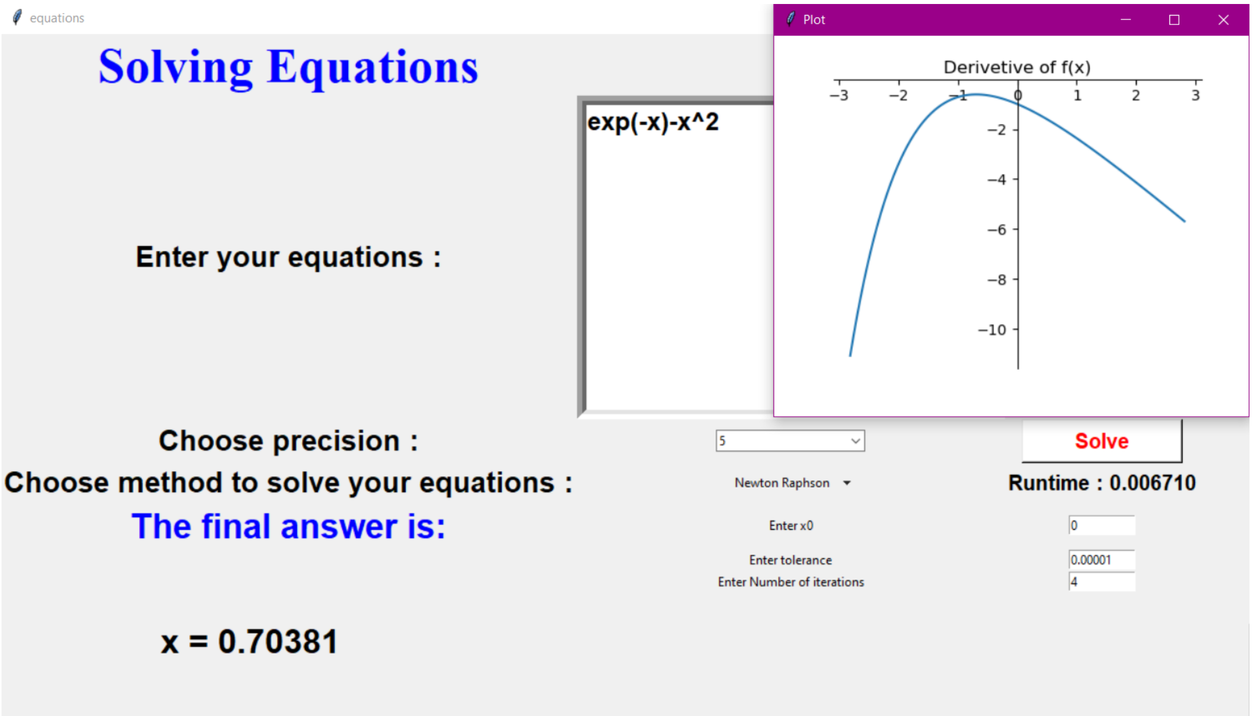

equations
Solving Equations
Enter your equations :
sin(x)-x
Plot
y=x
g(x)
xr
Choose precision :
Solve
Choose method to solve your equations :
Fixed Point
Runtime : 0.009920
The final answer is:
Enter x0
1
Enter tolerance
0.00001
Enter Number of iterations
x = 0.2318
Activate Windows


equations
Solving Equations
Enter your equations :
sin(x)-x^2
Plot
y=x
g(x)
xr
Choose precision :
6
Solve
Choose method to solve your equations :
Fixed Point
Runtime : 0.009920
The final answer is:
Enter x0
1
Enter tolerance
0.00001
Enter Number of iterations
19
x = 0.876726
Activate Windows
Go to Settings to activate Windows.

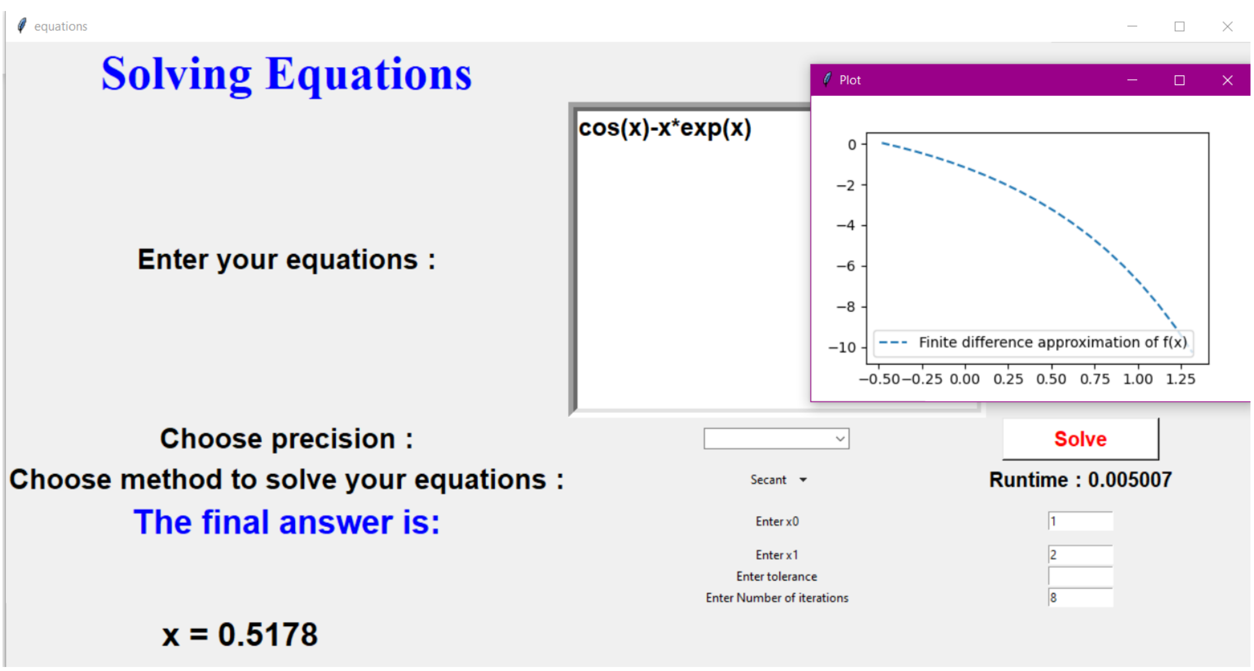


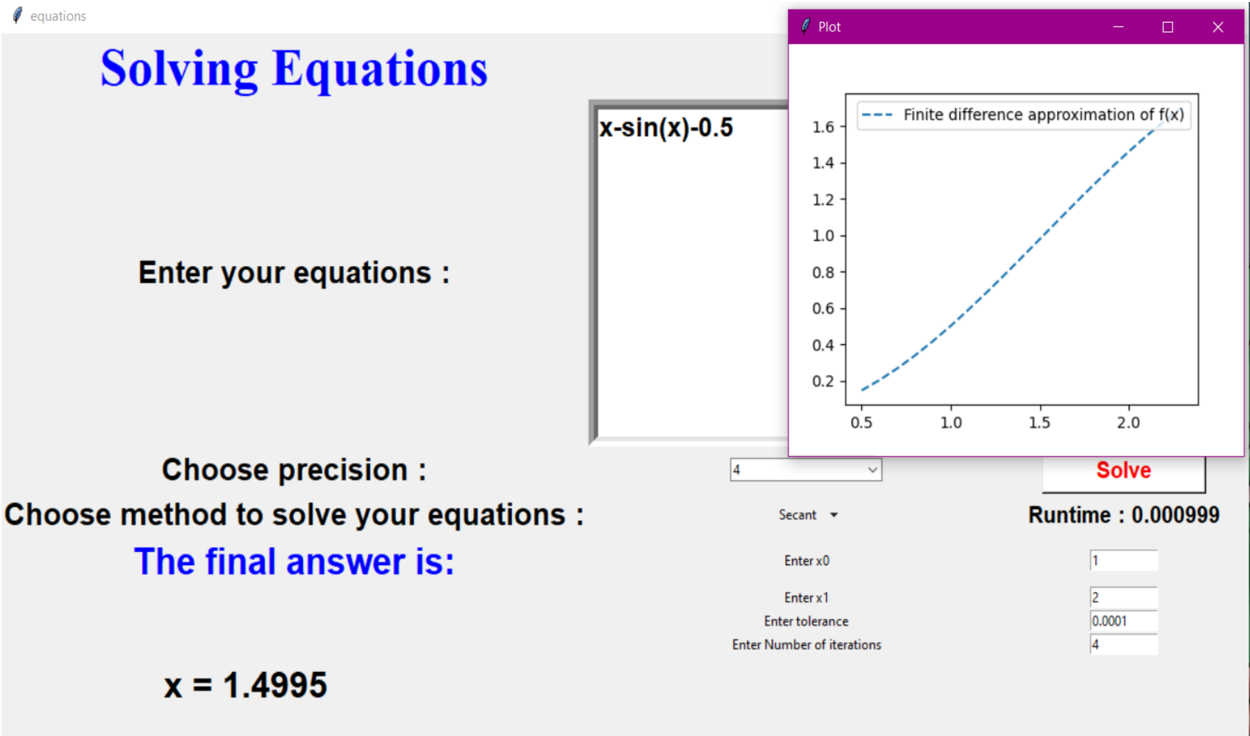
4. Newton Raphson



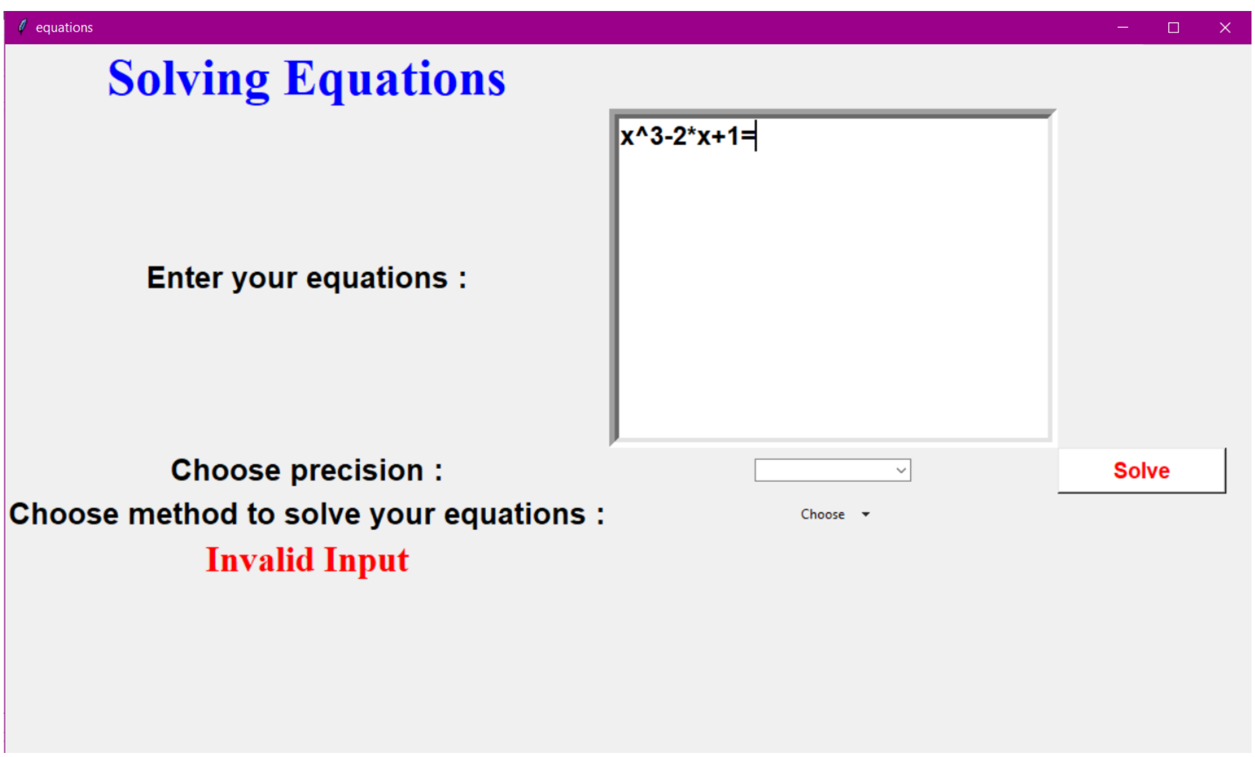

equations
Solving Equations
Enter your equations :
exp(-x)-x^2
Plot
Derivetive of f(x)
Choose precision :
5
Solve
Choose method to solve your equations :
Newton Raphson
Runtime : 0.006710
The final answer is:
Enter x0
0
Enter tolerance
0.00001
Enter Number of iterations
4
x = 0.70381

5. Secant


equations
Solving Equations
Enter your equations :
cos(x)-x*exp(x)
Plot
Finite difference approximation of f(x)
Choose precision :
Solve
Choose method to solve your equations :
Secant
Runtime : 0.005007
The final answer is:
Enter x0
1
Enter x1
2
Enter tolerance
Enter Number of iterations
8
x = 0.5178



6. Input Validation



Comparison between different methods

P.O.C	Time Complexity	Convergence	Best Case	Approximate error
Bisection	O (n)	-Guaranteed convergence but slow -Linear convergence rate	Root is the median of the upper and lower values.	The error bound is guaranteed to decrease by one-half with each iteration
False Position	O (n)	-Guaranteed convergence -Linear convergence rate	Root is the median of the upper and lower values.	The error bound is guaranteed to decrease by one-half with each iteration
Fixed Point	O (logn)	-May diverge -Linear convergence rate	When $\lvert g'(x)\rvert<1$ where the slope of the line $g(x)=x$	The error is roughly proportional to or less than the error of the previous step, therefore it is called linearly convergent when $\lvert g'(x)\rvert < 1$.

Newton Raphson	O (logn)	-May diverge -Quadratic convergence rate	A function fx with simple derivative and good choice of initial value xi.	-Error in xn+1 is nearly proportional to the square of the error in xn. -When the initial error is sufficiently small, this shows that the error in the succeeding iterates will decrease very rapidly
Secant	O (logn)	-May diverge. -Aureal number (super linear) convergence rate	The initial values x0, x1, are sufficiently close to the root.	-The error formula closely resembles the Newton Raphson error formula. -Newton's method should require fewer iterations to attain a given error tolerance. -However, secant method will require less time per iteration than the Newton method.

Data structures used

- Arrays and Lists: used for storing the points data in plotting graphs.