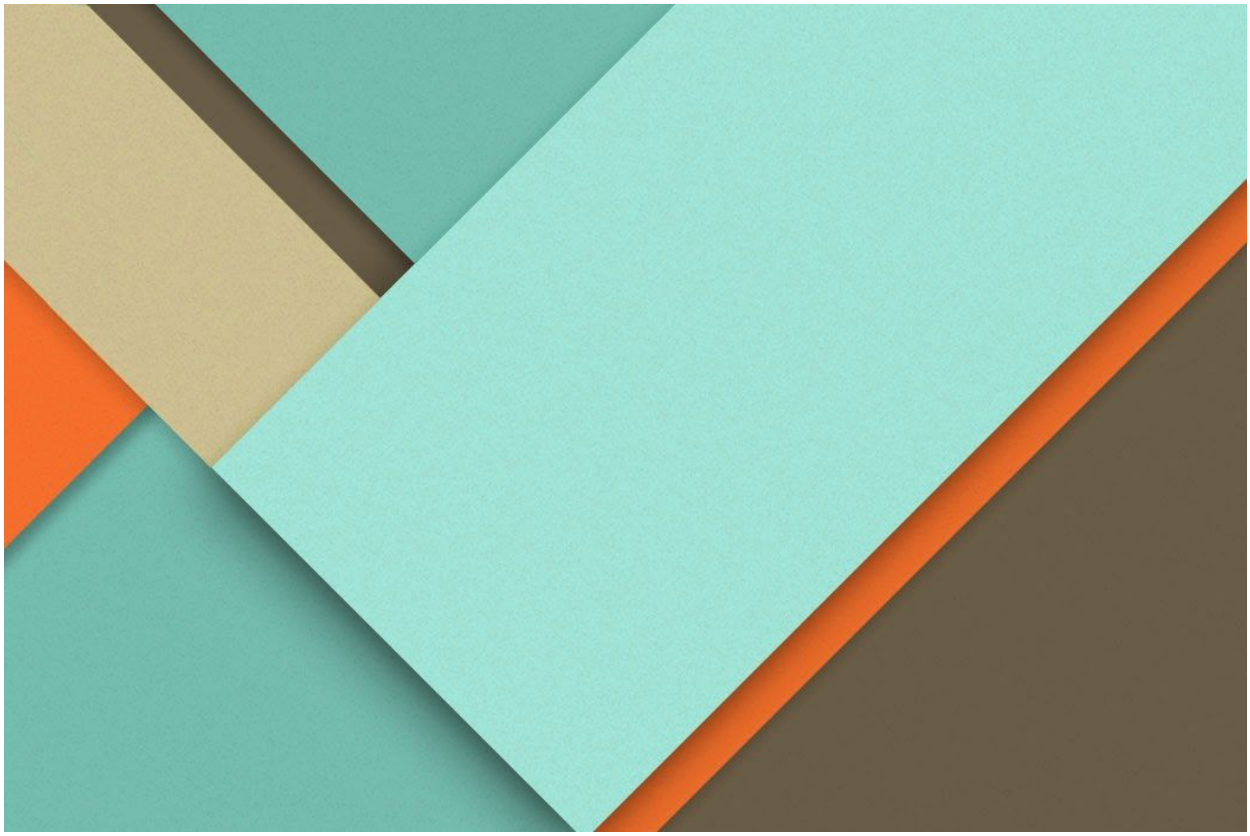


Project Report - Part I

Numerical Analysis

Abobakr Abdelaziz 2

Elsayed Akram Elsayed 16

Fares Medhat 47

Mahmoud Mohamed 60

Mohamed Salah 55

Overview

The aim of this part is to compare and analyze the behavior of the different numerical methods studied in class: Bisection, False-Position, Fixed point, Newton-Raphson, and Secant.

Algorithms Analysis

General Algorithm

- ➢ Birge_Vieta algorithm was chosen to be our General Algorithm Because it produces the next equation to get the rest of the roots.
- ➢ Problem with Birge_Vieta :
  - It only works with polynomial equations, so we check if the equation has any function, ex: exp(x), then we can use any other iterative algorithm such as Newton-Raphson.
- ➢ Inputs :
  - Equation (required).
  - Max Iterations, Error tolerance, initial guess for root (Optional).
- ➢ Outputs :
  - Roots: Vector of roots
- ➢ Pseudo Code:

```
If (the expression is not polynomial)
      Go to any iterative code
Else
      Assume root = 1
      For i from 1 to numOfRootS
            While root is not found && max Iterations not exceeded
                  Calculate a's, b's, c's where :
                        a = coeff( expression )
                        b□ = a□ + root * b□-1
                        c□ = b□ + root * c□-1
```

```
                        root = root - b0 / c1
                Endwhile
                Add root to roots Vector
        EndFor
        Return roots
```

Bisection

- ➢ Inputs :
  - Function, interval (upper and lower) (required).
  - Max Iterations, Error tolerance (Optional).
- ➢ Outputs :
  - root and vectors of (upper, lower, mid (guess), error in each iteration)
- ➢ Pseudo Code:

```
For i from 1 to maxIter
        mid = ((upper - lower) / 2) + lower;
        error = abs((new mid - old mid) / (new mid)) * 100
        If error < error tolerance
                Break
        If (func(mid) * func(lower) < 0)
                Upper = mid
        Else
                Lower = mid
endFor
Return all calculations done in the above for_loop
```

False-Position

- ➢ Inputs :
  - Function, interval (upper and lower) (required).
  - Max Iterations, Error tolerance (Optional).
- ➢ Outputs :
  - root and vectors of (upper, lower, mid (guess), error in each iteration)
- ➢ Pseudo Code:

```
For i from 1 to maxIter
      mid = ((lower * func(upper)) - (upper * func(lower))) /
(func(upper) - func(lower))
      error = abs((new mid - old mid) / (new mid)) * 100
      If error < error tolerance
            Break
      If (func(mid) * func(lower) < 0)
            Upper = mid
      Else
            Lower = mid
endFor
Return all calculations done in the above for_loop
```

Fixed Position

- ➢ Inputs :
  - Function, G(x), initial X(required).
  - Max Iterations, Error tolerance (Optional).
- ➢ Outputs :
  - root and vectors of (current Guess, next Guess(G(current Guess)), error in each iteration)
- ➢ Pseudo Code:

```
For i from 1 to maxIter
      nextGuess = G(CurrentGuess)
      error = abs((nextGuess - CurrentGuess) / abs(nextGuess)) *
100;
      If error < error tolerance
            Break
      currentGuess = nextGuess // update the root Guess
endFor
Return all calculations done in the above for_loop
```

Newton-Raphson

- Inputs:
  - Required: function and initial X (xi)
  - Optional: max iterations and error tolerance
- Outputs:
  - root and vectors of (xi, final X (xf), error in each iteration)
- Pseudo Code:

```
If the number of arguments == 2
      Max iterations = 50 (default)
      Error tolerance = 0.0001 (default)
Else if the number of arguments == 3
      Error tolerance = 0.0001 (default)
Getting the differentiation of input function (f ‘)
For i = 1 to max iterations
      xf = xi - f(xi) / f ‘(xi)
      Error = absolute( (xf - xi) / xf ) * 100
      if the error is < error tolerance
            break from loop
      xi = xf
endFor
Return all calculations done in the above for_loop
```

Secant

- Inputs:
  - Required: function, initial guess (x0) and secondary guess (x1)
  - Optional: max iterations and error tolerance
- Outputs:
  - root and vectors of (x0, x1, final X (x2), error in each iteration)
- Pseudo Code:

```
If the number of arguments == 3
      Max iterations = 50 (default)
      Error tolerance = 0.0001 (default)
```

```
Else if the number of arguments == 4
      Error tolerance = 0.0001 (default)
For i = 1 to max iterations
      x2 = x1 - f(x1) * ( (x1 - x0) / (f(x1) - f(x0)) )
      Error = absolute( (x2 - x1) / x2 ) * 100
      x0 = x1
      x1 = x2
      xr = x2
      if the error is < error tolerance
            break from loop
endFor
Return all calculations done in the above for_loop
```

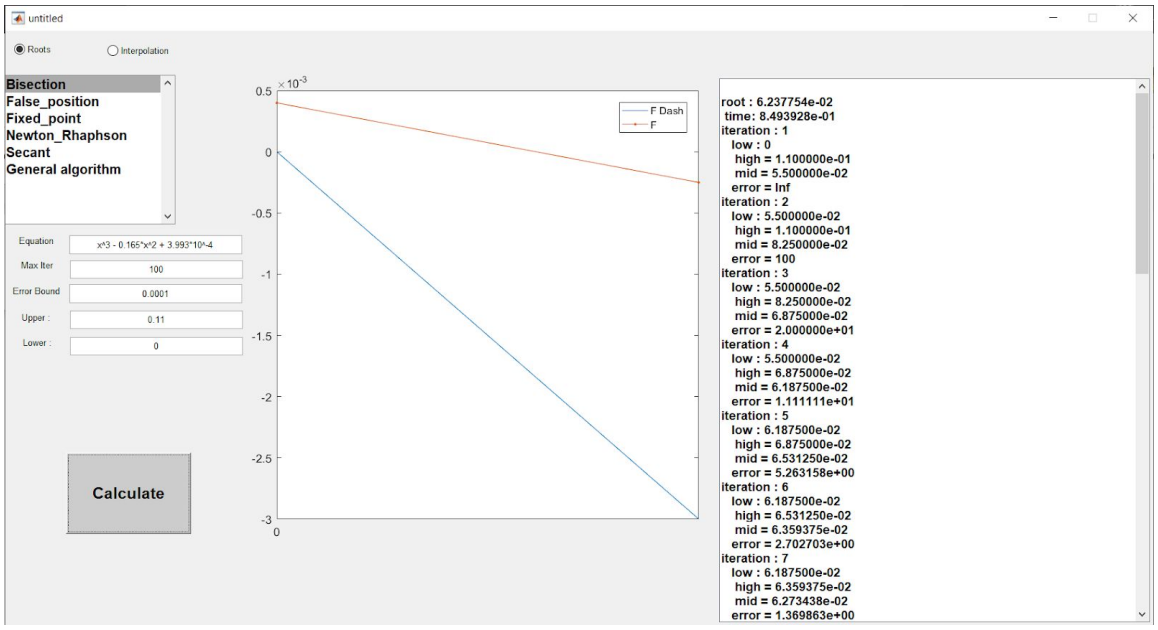
Data Structures Used & Why

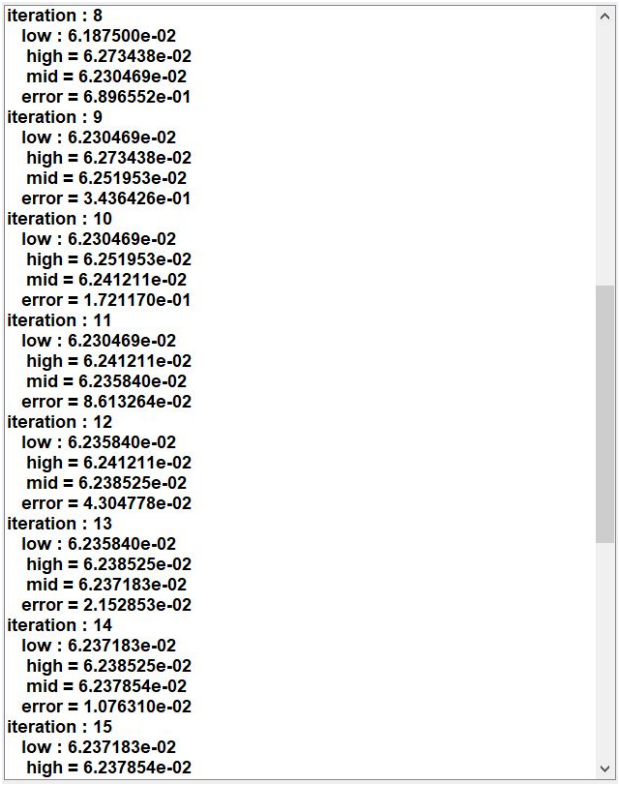
Vectors: because they are fast and reliable.

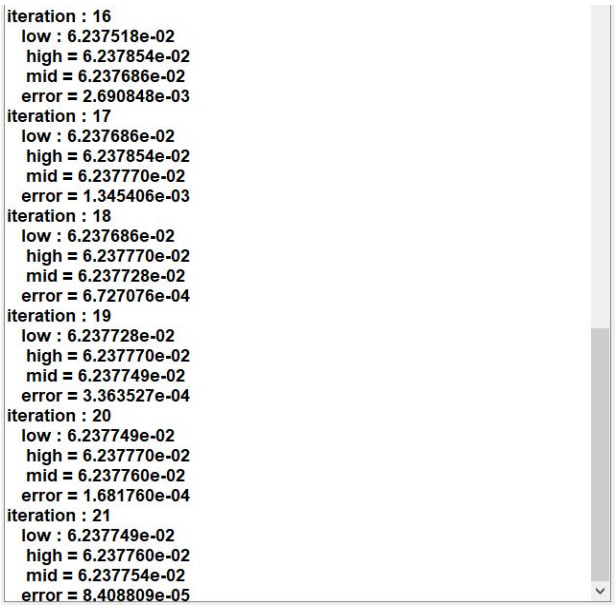
Different Examples & their Analysis Templates

I. Example 1

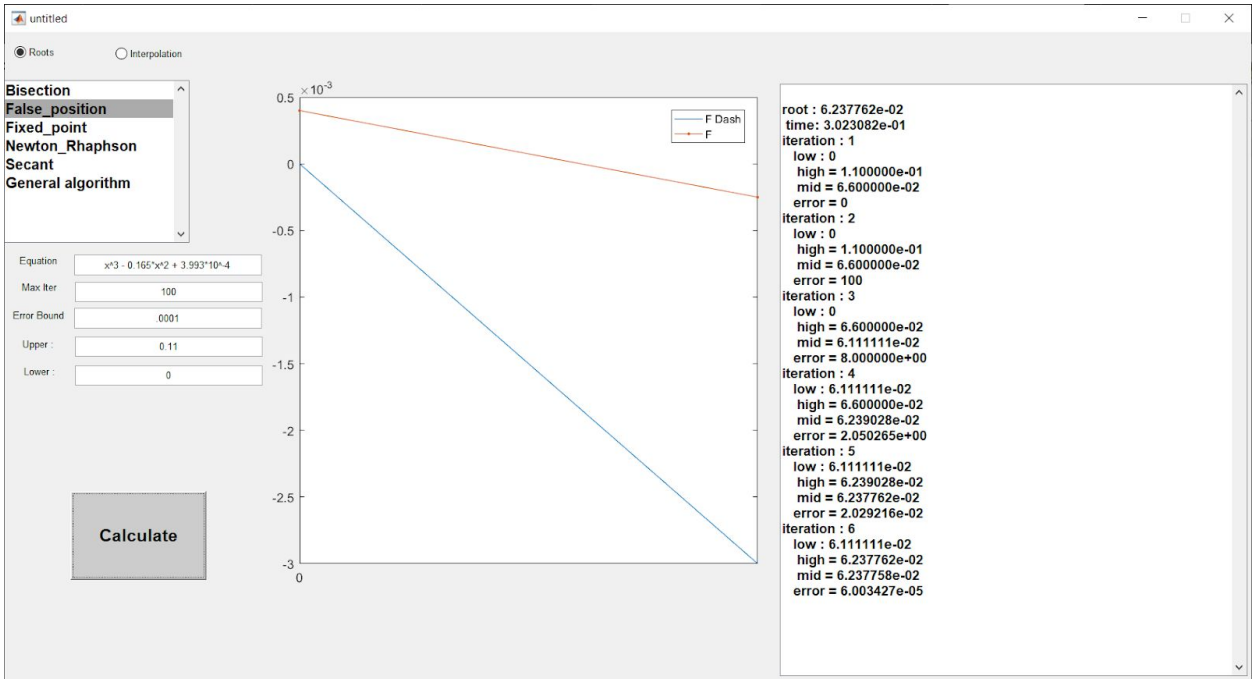
Using Bisection


untitled
Roots
Interpolation
Bisection
False_position
Fixed_point
Newton_Rhaphson
Secant
General algorithm
Equation
x^3 - 0.165*x^2 + 3.993*10^-4
Max Iter
100
Error Bound
0.0001
Upper :
0.11
Lower :
0
Calculate
F Dash
F
root : 6.237754e-02
time: 8.493928e-01
iteration : 1
low : 0
high = 1.100000e-01
mid = 5.500000e-02
error = Inf
iteration : 2
low : 5.500000e-02
high = 1.100000e-01
mid = 8.250000e-02
error = 100
iteration : 3
low : 5.500000e-02
high = 8.250000e-02
mid = 6.875000e-02
error = 2.000000e+01
iteration : 4
low : 5.500000e-02
high = 6.875000e-02
mid = 6.187500e-02
error = 1.111111e+01
iteration : 5
low : 6.187500e-02
high = 6.875000e-02
mid = 6.531250e-02
error = 5.263158e+00
iteration : 6
low : 6.187500e-02
high = 6.531250e-02
mid = 6.359375e-02
error = 2.702703e+00
iteration : 7
low : 6.187500e-02
high = 6.359375e-02
mid = 6.273438e-02
error = 1.369863e+00

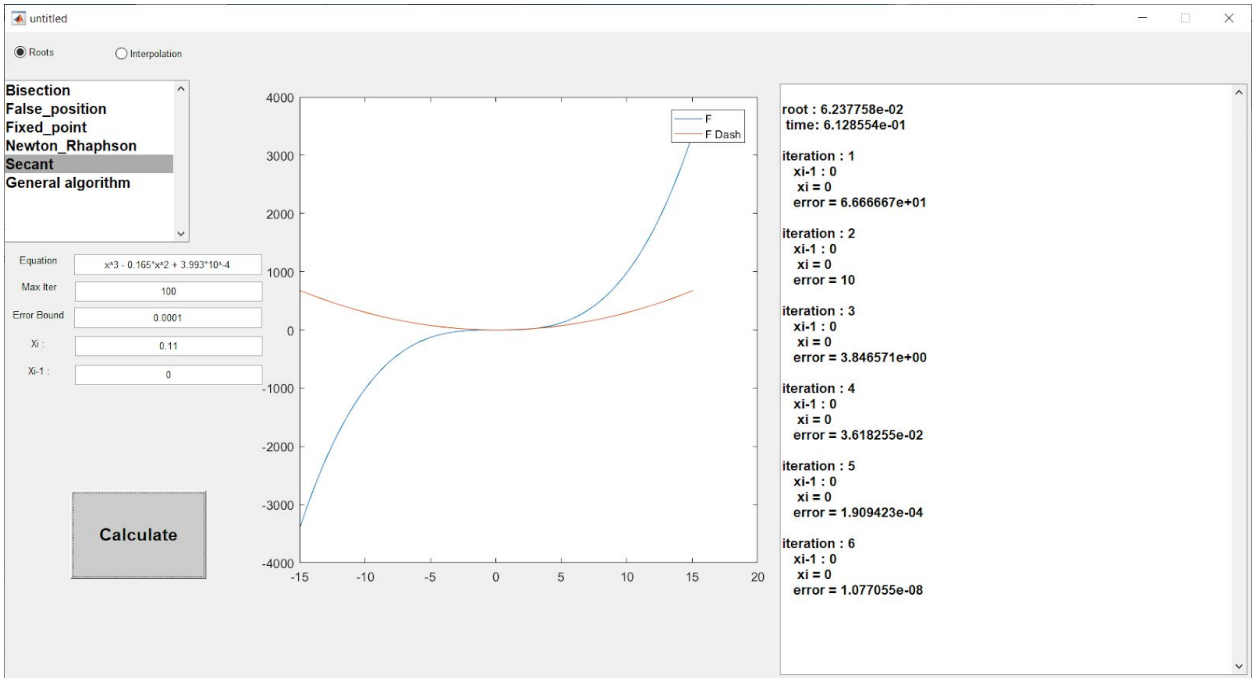

iteration : 8
low : 6.187500e-02
high = 6.273438e-02
mid = 6.230469e-02
error = 6.896552e-01
iteration : 9
low : 6.230469e-02
high = 6.273438e-02
mid = 6.251953e-02
error = 3.436426e-01
iteration : 10
low : 6.230469e-02
high = 6.251953e-02
mid = 6.241211e-02
error = 1.721170e-01
iteration : 11
low : 6.230469e-02
high = 6.241211e-02
mid = 6.235840e-02
error = 8.613264e-02
iteration : 12
low : 6.235840e-02
high = 6.241211e-02
mid = 6.238525e-02
error = 4.304778e-02
iteration : 13
low : 6.235840e-02
high = 6.238525e-02
mid = 6.237183e-02
error = 2.152853e-02
iteration : 14
low : 6.237183e-02
high = 6.238525e-02
mid = 6.237854e-02
error = 1.076310e-02
iteration : 15
low : 6.237183e-02
high = 6.237854e-02


iteration : 16
low : 6.237518e-02
high = 6.237854e-02
mid = 6.237686e-02
error = 2.690848e-03
iteration : 17
low : 6.237686e-02
high = 6.237854e-02
mid = 6.237770e-02
error = 1.345406e-03
iteration : 18
low : 6.237686e-02
high = 6.237770e-02
mid = 6.237728e-02
error = 6.727076e-04
iteration : 19
low : 6.237728e-02
high = 6.237770e-02
mid = 6.237749e-02
error = 3.363527e-04
iteration : 20
low : 6.237749e-02
high = 6.237770e-02
mid = 6.237760e-02
error = 1.681760e-04
iteration : 21
low : 6.237749e-02
high = 6.237760e-02
mid = 6.237754e-02
error = 8.408809e-05

Using False-Position

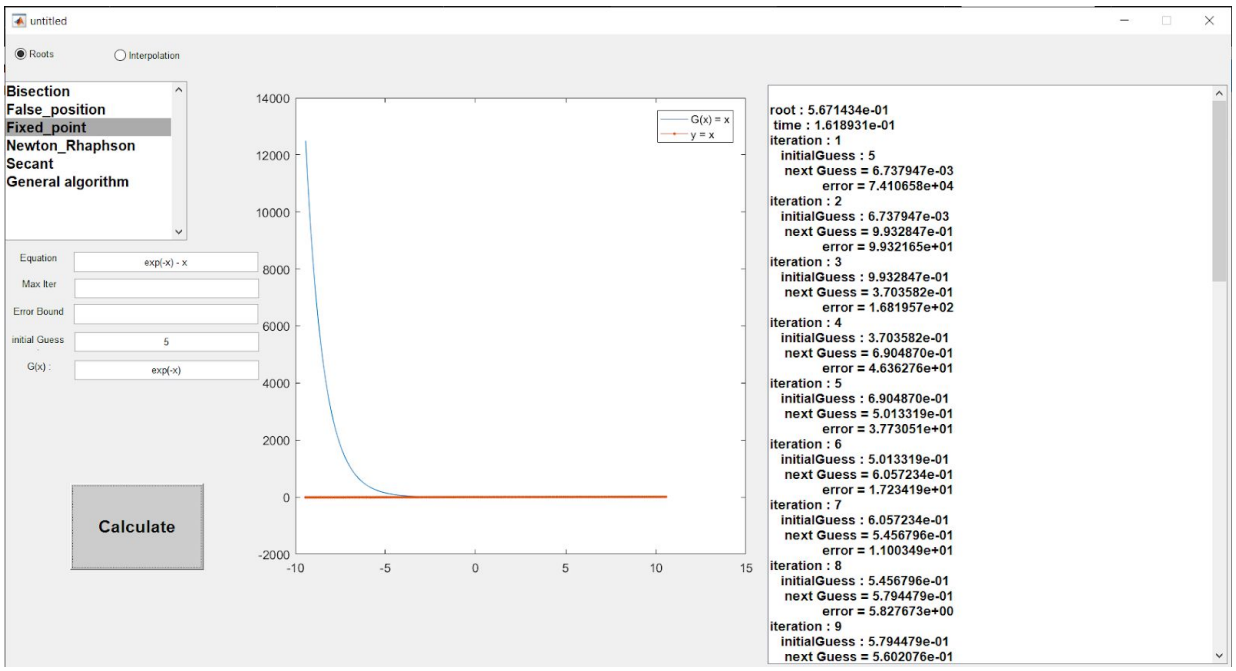


Using Secant



II. Example 2

Using Fixed Point

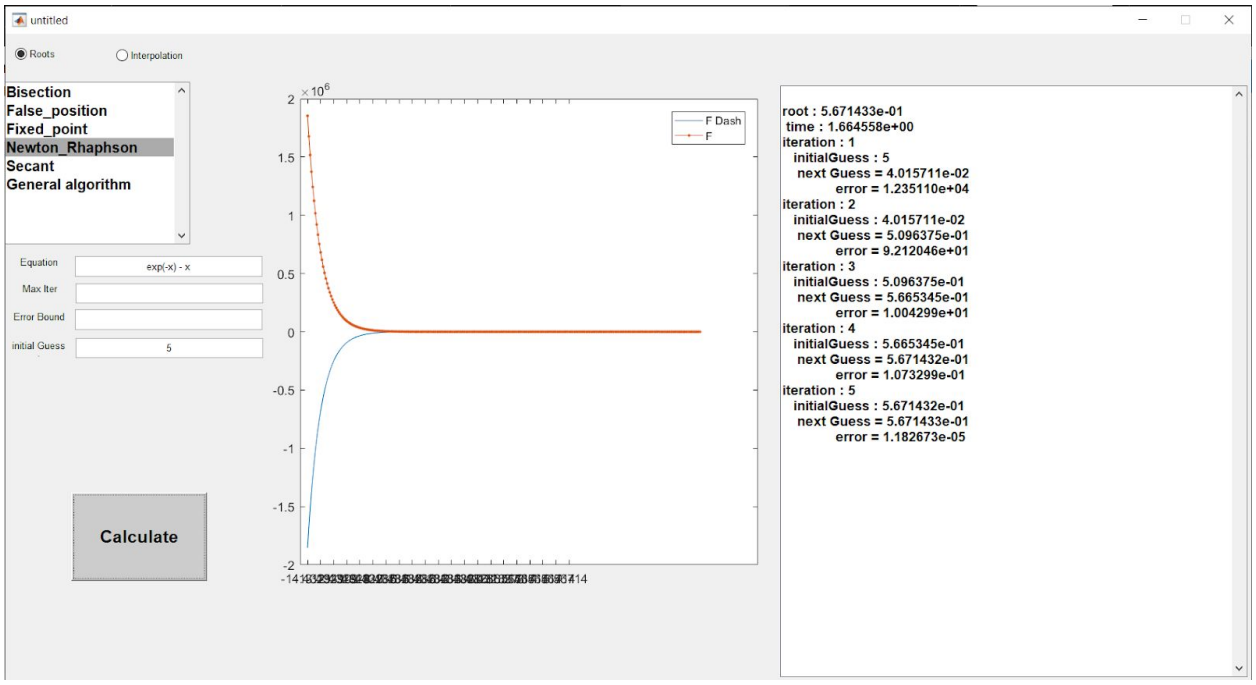



```
iteration : 9
  initialGuess : 5.794479e-01
   next Guess = 5.602076e-01
         error = 3.434495e+00
iteration : 10
  initialGuess : 5.602076e-01
   next Guess = 5.710905e-01
         error = 1.905639e+00
iteration : 11
  initialGuess : 5.710905e-01
   next Guess = 5.649091e-01
         error = 1.094236e+00
iteration : 12
  initialGuess : 5.649091e-01
   next Guess = 5.684118e-01
         error = 6.162371e-01
iteration : 13
  initialGuess : 5.684118e-01
   next Guess = 5.664243e-01
         error = 3.508907e-01
iteration : 14
  initialGuess : 5.664243e-01
   next Guess = 5.675512e-01
         error = 1.985556e-01
iteration : 15
  initialGuess : 5.675512e-01
   next Guess = 5.669120e-01
         error = 1.127540e-01
iteration : 16
  initialGuess : 5.669120e-01
   next Guess = 5.672745e-01
         error = 6.390117e-02
iteration : 17
  initialGuess : 5.672745e-01
   next Guess = 5.670689e-01
         error = 3.625607e-02
iteration : 18
  initialGuess : 5.670689e-01
```

```
iteration : 18
  initialGuess : 5.670689e-01
   next Guess = 5.671855e-01
         error = 2.055758e-02
iteration : 19
  initialGuess : 5.671855e-01
   next Guess = 5.671194e-01
         error = 1.166064e-02
iteration : 20
  initialGuess : 5.671194e-01
   next Guess = 5.671569e-01
         error = 6.612756e-03
iteration : 21
  initialGuess : 5.671569e-01
   next Guess = 5.671356e-01
         error = 3.750540e-03
iteration : 22
  initialGuess : 5.671356e-01
   next Guess = 5.671477e-01
         error = 2.127042e-03
iteration : 23
  initialGuess : 5.671477e-01
   next Guess = 5.671408e-01
         error = 1.206354e-03
iteration : 24
  initialGuess : 5.671408e-01
   next Guess = 5.671447e-01
         error = 6.841704e-04
iteration : 25
  initialGuess : 5.671447e-01
   next Guess = 5.671425e-01
         error = 3.880244e-04
iteration : 26
  initialGuess : 5.671425e-01
   next Guess = 5.671437e-01
         error = 2.200649e-04
iteration : 27
  initialGuess : 5.671437e-01
```

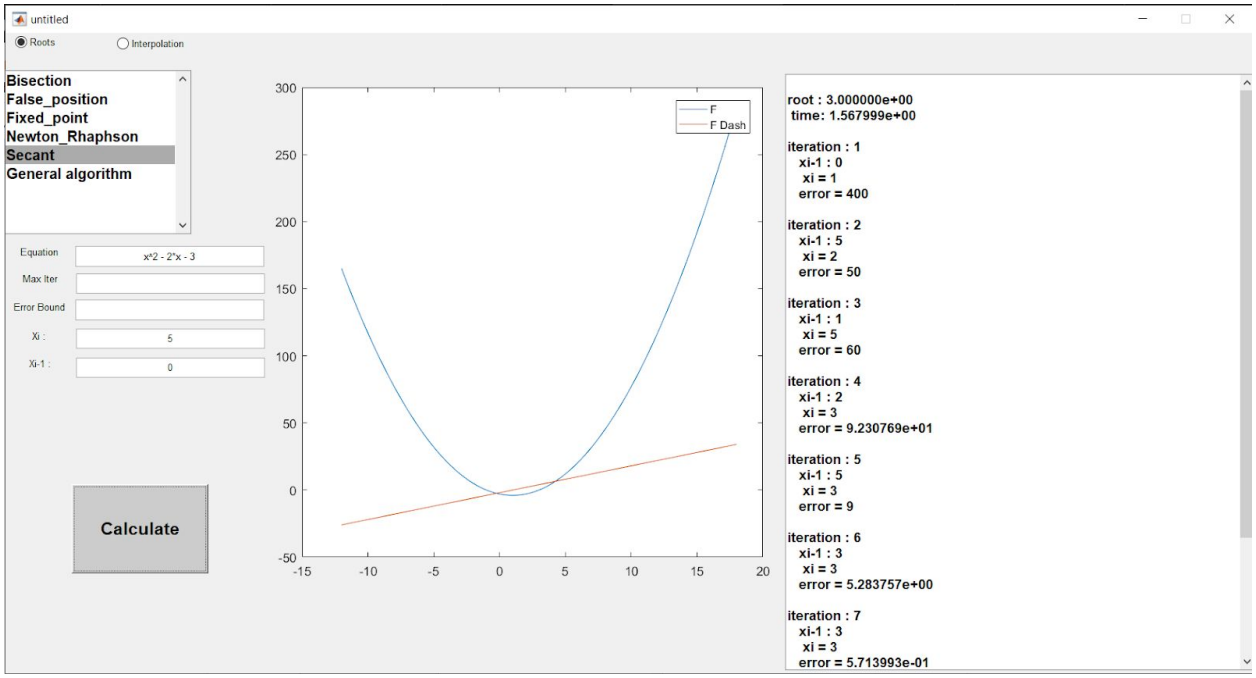
```
iteration : 27
  initialGuess : 5.671437e-01
   next Guess = 5.671430e-01
         error = 1.248085e-04
iteration : 28
  initialGuess : 5.671430e-01
   next Guess = 5.671434e-01
         error = 7.078424e-05
```

Using Newton-Raphson

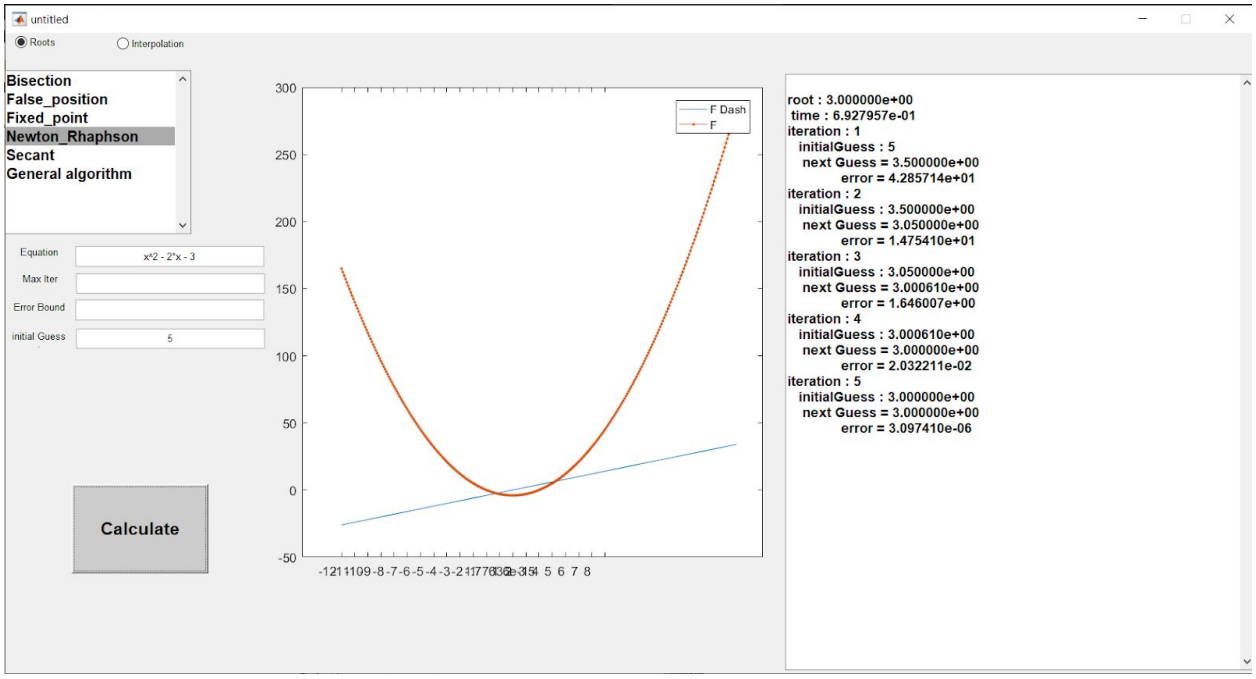


III. Example 3

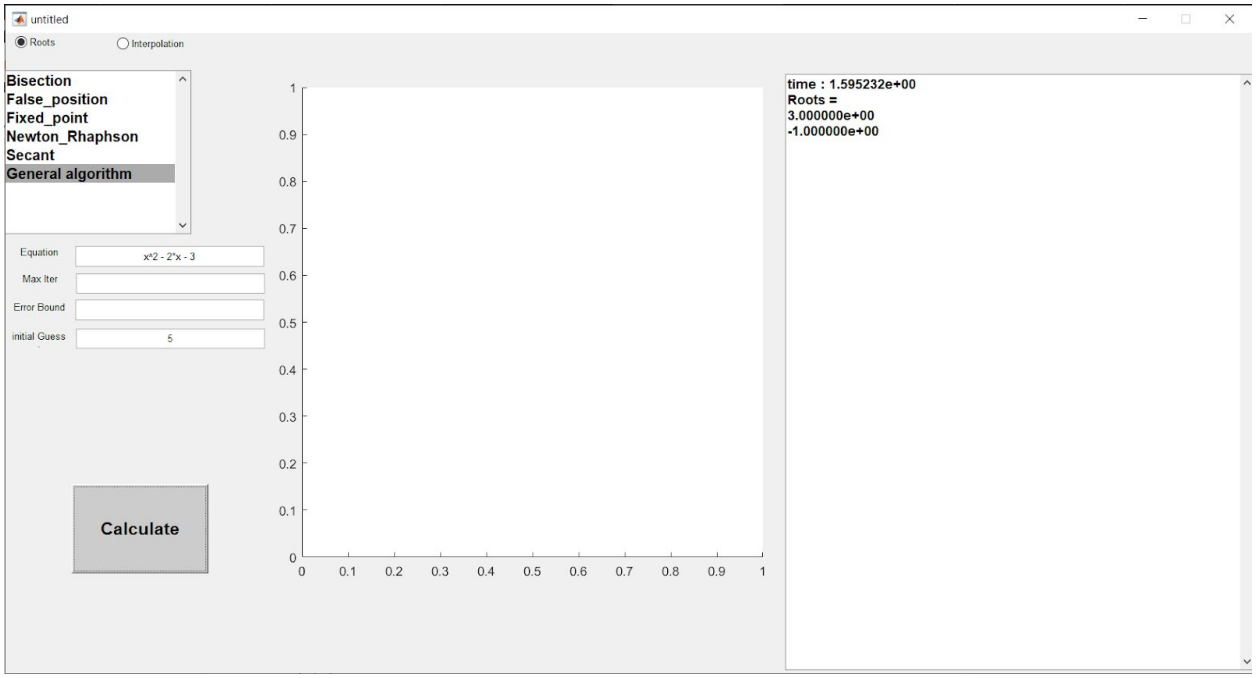
Using Secant



Using Newton-Raphson



Using The General Algorithm



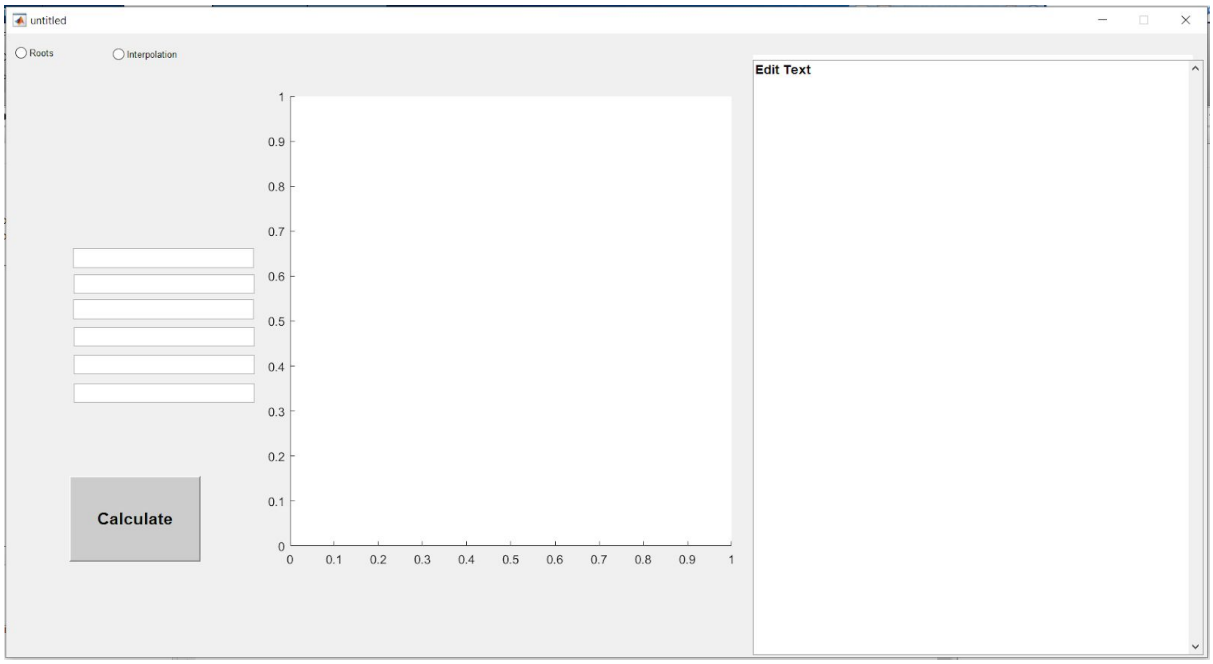
Problematic Functions

Fixed Point Algorithm:

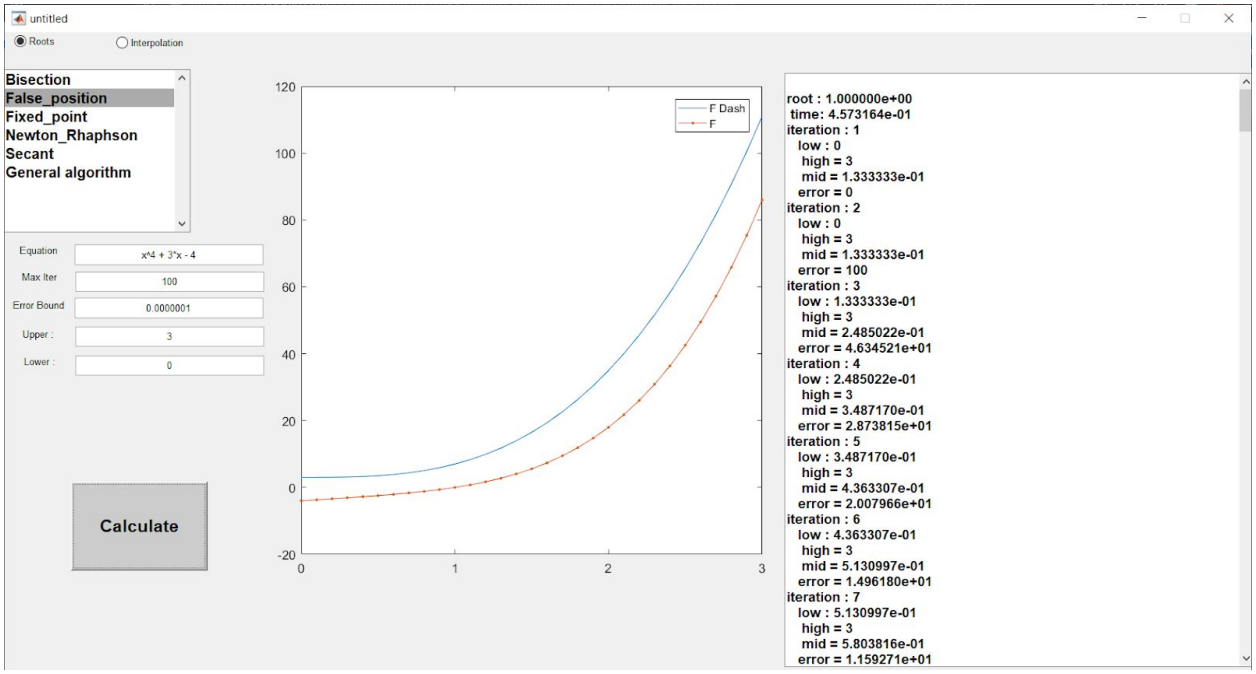
1. There is no check for the chosen G(x) if it's correct.
2. There is no guarantee that the algorithm won't diverge.

Sample Runs

Initial



After checking roots then testing false position



After that testing the general algorithm