AMS 326 Report

William Zhang

February 2019

1 Problem Description

The following section is about finding a way to integrate the following function. Where the function is $x^2 - (y^2 - \sqrt{|x|})^2 = 2$. The function produces a heart shape on a 2D cartesian plane and therefore we want to find the area under this curve. My program uses the simpson method to approximate the integral

1.1 Algorithm Description/PseudoCode

This function was challenging to integrate. This function was non continous as well and was a finitely closed curve. Thus the function needed to be broken in 3 parts or rather 3 stages in order to compute . The reader should realize the function is symmetric along the y axis. That is why we dont need to integrate the function across the entire domain. We only to find the function where it is defined on the positive X axis. Thus the absolute function of X automatically turns positive. Then the function is turn into

$$x^2 - (y^2 - \sqrt{x})^2 = 2$$

When we integrate this function we need to multiply by 2 times to find the total area of the function. But then we needed to break this function even further is composed of two seperate functions in terms of x . When we solve the function for Y in terms of X we get two functions. Thus we will have 2 seperate methods for each integral.

$$y = \sqrt{2 - x^2} + \sqrt{x}$$
$$y = -\sqrt{2 - x^2} + \sqrt{x}$$

The top function represents the upper half the heart that is defined on the positive X axes. The bottom function represents the bottom half of the heart. So in order to compute the integral of the function that is defined on the positive X we needed to perform simpson method on both functions to approximate the area under the positive x. The following is the formula for simpson 1/3 rule

$$\int_a^b f(x)\,dx \approx \frac{\Delta x}{3}\left(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \cdots + 4f(x_{n-1}) + f(x_n)\right)$$

$$\Delta x = \frac{b-a}{n}$$
$$x_i = a + i\Delta x\text{”}$$

Now to the psuedocode for both the simpsonmethod. The function method is seperate and just represents the function itself. And it takes a float input and transforms and returns a function corresponding to its x

Algorithm 1: Simpson Method for $y = \sqrt{2-x^2} + sqrt(x)$

```
1 $n = 4000$ Which is the maximum number of iterations
2 $[a,b] = [0,\sqrt{2}]$ This is the domain of the function
3 $summation = 0.0$ This is to track the summation in the loop
4 $\Delta x = \frac{\sqrt{2}-0}{n}$ This is going to represent the height
5 for $K = 0$ to $n+1$ do
      if $x == 0$ then
          summation += function1(0);
      else if $x$ is odd then
          summation += 4 * function1(0.00 + x * height)
      else if $X$ is even then
          summation += 2 * function1(0.00 + x * height)
      else
          summation += 2 * function1(sqrt(2))
      return $\frac{\Delta x}{3}$ * summation
```

Algorithm 2: Simpson Method 2 for $y = -\sqrt{2-x^2} + \sqrt{x}$

```
1 $n = 4000$ Which is the maximum number of iterations
2 $[a,b] = [0,\sqrt{2}]$ This is the domain of the function
3 $summation = 0.0$ This is to track the summation in the loop
4 $\Delta x = \frac{\sqrt{2}-0}{n}$ This is going to represent the height
5 for $K = 0$ to $n+1$ do
      if $x == 0$ then
          summation += function1(0);
      else if $x$ is odd then
          summation += 4 * function1(0.00 + x * height)
      else if $X$ is even then
          summation += 2 * function1(0.00 + x * height)
      else
          summation += 2 * function1(sqrt(2))
      return $\frac{\Delta x}{3}$ * summation
```

Algorithm 3: Computing area of integral

```
1 return 2 * (SimpsonMethod1() + SimpsonMethod2())
```

All the reader has to do in order to run program is too press run on the function

1.2 Results

We computed the integral and approximately

$$4.484778722$$

This is pretty accurate as we computed the seperate integrals that is defined for both positive and negative X axis on a integral calculator and we added them up to be 4.485

1.3 Comments and reports

My algorithms runs prettty fast as I am only doing half as much work by only explicitly computing the integral only defined for the x axis.Thus it runs in

$$\mathcal{O}(log(n)$$

2 Description

The following problem is to try to find roots of the following polynomial

$$2.019^{-x^3} - x^5 * sin(x^4) - 1.984$$

. We use the bisection method to find all roots in the function

2.1 Algorithm Description/PsuedoCode

The following algorithm I used was called the bisection method.The algorithm requires intervals $[a, b]$ and perfrom a binary search on the interval $a, b]$ to find the root. To determine $[a, b]$ we first approximate the root on the graph first and then specify a error difference

$$[a - error, b + error]$$

. However, since we have multiple roots we must iterate through all the intervals that we think we might find the intervals. Same sign checks if two functions have the same same sign. That is we want to check if both numbers are or both of them are positive. The function is represented by the method function.

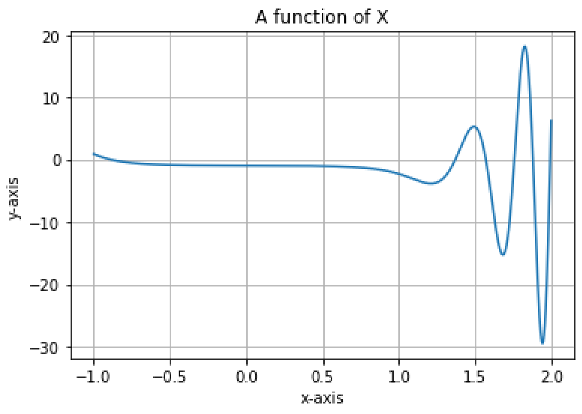
Algorithm 4: Bisection Method

```
1  $z = 1000$ the maximum number of iterations
2  interval = $[[a.b]....[a_1, b_1.], [a_n, b_n]]$ The intervals where roots might exist
3  for $x \in interval$ do
4      $a = [x][0]$
5      $b = [x][1]$
6      for $x$ to $z$ do
7          $c = \frac{a+b}{2}$
8          if function(c) = 0 then
               return $c$
9          else if samesign(function(a),function(b) then
               $a = c$
10         else
               $b = c$
       return $c$
```

2.2 Results and diagrams

The following are my results. Which includes graph,error,initial root, the approximated root and the boundaries. The reader that sees the code just has to run the code

```
In [4]: runfile('C:/Users/William Zhang/Desktop/PythonAdder/ComputationalFinance/RootFinder.py',
wdir='C:/Users/William Zhang/Desktop/PythonAdder/ComputationalFinance')
Boundaries: [-0.7, -0.9]        Rootx1: -0.8    Error: 0.1      Rootx0: -0.8925706635921733
Boundaries: [1.3, 1.4]  Rootx1: 1.35    Error: 0.05     Rootx0: 1.3704701300135875
Boundaries: [1.5, 1.6]  Rootx1: 1.55    Error: 0.05     Rootx0: 1.5703359881705325
Boundaries: [1.7, 1.8]  Rootx1: 1.75    Error: 0.05     Rootx0: 1.7575602933461454
Boundaries: [1.9, 2.0]  Rootx1: 1.95    Error: 0.05     Rootx0: 1.9928053306412115
```



```
In [5]:
```

2.3 Comments

The performance is extremely fast. The bisection method only requires us to do less work by only searching only half as much numbers. The only draw back since this has multiple roots we cannot use its whole domain from the interval $[a, b]$.