Pablo Leon

CS 301

**Project #2 Report**

**Part A: f(x) = 2x3 – 11.7x2 + 17.7x – 5**

Roots:

* .36510
* 1.9217
* 3.5632

When testing my programs I was quite happy with my results as all of my programs did arrive at the desired root output when given the correct starting points. But when I shifted those results over to graphing I realized I should have used the same starting point for testing each method for each root, but since I didn’t I feel my graphs and number of iterations are a tad misleading. Thus now that I have made clear this oversight I now present my data from my programs for part A.

**1st Root .36510**

//Bisection Method

Enter a value for a:0

Enter a value for b:1

Iterations Root Error

0 .5 1.0

1 .25 1.0

2 .375 .3333

3 .3125 .2

4 .3438 .0909

5 .3594 .0435

6 .3672 .0213

7 .3633 .0108

8 .3652 .0053

Approximate solution = 0.3642578125

//NewtonRaphson

Enter a value for x:0

Iter X Xn+1 F(x) Error

0 .0 .2825 -5.0 2.54

1 .2825 .3593 -.8886 .2138

2 .3593 .3651 -.0581 .0158

3 .3651 .3651 -.0003 .0001

Approximate solution = 0.36509824238985855

//Secant

Enter a value for x0:0

Enter a value for x1:1

Iter Xn-1 X Xn+1 F(xn-1) F(x) F(xn+1) Error

0 1.0 .0 .625 3.0 -5.0 1.9805 1.0

1 .625 .0 .4477 1.9805 -5.0 .7585 .3961

2 .4477 .0 .3887 .7585 -5.0 .2298 .1517

3 .3887 .0 .3716 .2298 -5.0 .0646 .046

4 .3716 .0 .3669 .0646 -5.0 .0178 .0129

5 .3669 .0 .3656 .0178 -5.0 .0049 .0036

Approximate solution = .3656

//Modified Secant

Enter a value for x0:1

Iter X Xn+1 F(x) Error

0 1.0 -11.3355 3.0 1.0882

1 -11.3355 -6.9873 -4622.1182 .6223

2 -6.9873 -4.0951 -1382.1475 .7063

3 -4.0951 -2.1889 -411.0295 .8708

4 -2.1889 -.9591 -120.7745 1.2821

5 -.9591 -.2063 -34.5047 3.65

6 -.2063 .1956 -9.1663 2.0545

7 .1956 .3434 -1.9704 .4304

8 .3434 .3647 -.2204 .0585

9 .3647 .3651 -.0037 .001

Approximate solution = .3651

//False Position

Enter a value for x0:0

Enter a value for x1:1

Iter Xn-1 X Xn+1 F(xn-1) F(x) F(xn+1) Error

0 .0 1.0 .625 -5.0 3.0 1.9805 1.0

1 .0 .625 .4477 -5.0 1.9805 .7585 .3961

2 .0 .4477 .3887 -5.0 .7585 .2298 .1517

3 .0 .3887 .3716 -5.0 .2298 .0646 .046

4 .0 .3716 .3669 -5.0 .0646 .0178 .0129

5 .0 .3669 .3656 -5.0 .0178 .0049 .0036

Approximate solution = .3656

As can be seen above all of my methods did arrive at a close approximation of the actual root But the graph shows some weird results especially with the Modified Secant Method. If I had the time to go back and fix this oversight I would have. But once I overlook that issue my graph seems to show that secant was the quickest method to reach the root while Modified secant ran onto some issue.

**2nd Root 1.9217**

//Bisection Method

Enter a value for a:1

Enter a value for b:3

Iterations Root Error

0 2.0 .5

1 1.5 .3333

2 1.75 .1429

3 1.875 .0667

4 1.9375 .0323

5 1.9062 .0164

6 1.9219 .0081

Approximate solution = 1.9140625

//NewtonRaphson

Enter a value for x:2

Iter X Xn+1 F(x) Error

0 2.0 1.9216 -.4 .4796

1 1.9216 1.9217 .0009 .0001

Approximate solution = 1.921740932764785

//Secant

Enter a value for x0:1

Enter a value for x1:3

Iter Xn-1 X Xn+1 F(xn-1) F(x) F(xn+1) Error

0 1.0 3.0 1.9677 3.0 -3.2 -.2352 1.0

1 1.9677 3.0 1.8858 -.2352 -3.2 .1832 .0434

2 1.8858 3.0 1.9462 .1832 -3.2 -.1248 .031

3 1.9462 3.0 1.9034 -.1248 -3.2 .0938 .0225

4 1.9034 3.0 1.9346 .0938 -3.2 -.0657 .0161

5 1.9346 3.0 1.9123 -.0657 -3.2 .0485 .0117

6 1.9123 3.0 1.9285 .0485 -3.2 -.0345 .0084

Approximate solution = 1.9285

//Modified Secant

Enter a value for x0:2.5

Iter X Xn+1 F(x) Error

0 2.5 1.6838 -2.625 .4847

1 1.6838 1.9339 1.1792 .1293

2 1.9339 1.9217 -.062 .0063

Approximate solution = 1.9217

//False Position

Enter a value for x0:3

Enter a value for x1:1

Iter Xn-1 X Xn+1 F(xn-1) F(x) F(xn+1) Error

0 3.0 1.0 1.9677 -3.2 3.0 -.2352 1.0

1 1.9677 1.0 1.8974 -.2352 3.0 .1244 .0371

2 1.9677 1.8974 1.9217 -.2352 .1244 .0001 .0127

3 1.9677 1.9217 1.9217 -.2352 .0001 .0 .0

Approximate solution = 1.9217

For the second root I believe I unintentionally picked much more comparable starting points as this point was located between two other roots. Therefore this restriction led me to pick the starting point in the range between 1 and 3. Thus the results of this graph line up closer to my expectations that Bisection method is the slowest one at reducing the error percentage. The Newton Raphson Method was the fastest to get to the root with only two iterations which is what I would expect from a method where the derivative is taken. Although I am surprised that the Secant method took as many iterations as the Bisection method to get under the .01 approximate error limit.

**3rd Root 3.5632**

//Bisection Method

Enter a value for a:3

Enter a value for b:5

Iterations Root Error

0 4.0 .75

1 3.5 .1429

2 3.75 .0667

3 3.625 .0345

4 3.5625 .0175

5 3.5938 .0087

Approximate solution = 3.578125

//NewtonRaphson

Enter a value for x:6

Iter X Xn+1 F(x) Error

0 6.0 4.7996 112.0 .7916

1 4.7996 4.0759 31.5573 .1776

2 4.0759 3.7033 8.1966 .1006

3 3.7033 3.5782 1.667 .035

4 3.5782 3.5634 .1606 .0042

Approximate solution = 3.5633662760929385

//Secant

Enter a value for x0:3

Enter a value for x1:7

Iter Xn-1 X Xn+1 F(xn-1) F(x) F(xn+1) Error

0 3.0 7.0 3.0545 -3.2 231.6 -3.0992 1.0

1 3.0545 7.0 3.1066 -3.0992 231.6 -2.966 .0168

2 3.1066 7.0 3.1558 -2.966 231.6 -2.8056 .0156

3 3.1558 7.0 3.2019 -2.8056 231.6 -2.6241 .0144

4 3.2019 7.0 3.2444 -2.6241 231.6 -2.4279 .0131

5 3.2444 7.0 3.2834 -2.4279 231.6 -2.2236 .0119

6 3.2834 7.0 3.3187 -2.2236 231.6 -2.0172 .0106

7 3.3187 7.0 3.3505 -2.0172 231.6 -1.8142 .0095

Approximate solution = 3.3505

//Modified Secant

Enter a value for x0:7

Iter X Xn+1 F(x) Error

0 7.0 5.4563 231.6 .2829

1 5.4563 4.4802 68.1358 .2179

2 4.4802 3.912 19.3104 .1453

3 3.912 3.645 4.9242 .0732

4 3.645 3.571 .9248 .0207

5 3.571 3.5635 .0825 .0021

Approximate solution = 3.5635

//False Position

Enter a value for x0:3

Enter a value for x1:5

Iter Xn-1 X Xn+1 F(xn-1) F(x) F(xn+1) Error

0 3.0 5.0 3.1448 -3.2 41.0 -2.8446 1.0

1 3.1448 5.0 3.2652 -2.8446 41.0 -2.322 .0369

2 3.2652 5.0 3.3581 -2.322 41.0 -1.7628 .0277

3 3.3581 5.0 3.4258 -1.7628 41.0 -1.2644 .0198

4 3.4258 5.0 3.4729 -1.2644 41.0 -.87 .0136

5 3.4729 5.0 3.5047 -.87 41.0 -.5815 .0091

Approximate solution = 3.5047

For the final root of this function I feel I once again got a very random graph do to my oversight of the differing starting points. But this graph is where I realized I had an error in my code where the error was coming out as a negative which made almost go back and redo my test data again. Sadly I was short on time as this point and couldn’t do so but I did fix this issue for the final test of my programs in Part B.

**Part B: f(x) = x + 10 – xcosh(50/x)**

Root: 126.632

//Bisection Method

Enter a value for a:80

Enter a value for b:150

The true root is 126.632

Iterations Root Error

0 115.0 .1011

1 132.5 .0443

2 123.75 .0233

3 128.125 .0117

4 125.9375 .0055

Approximate solution = 127.03125

//NewtonRaphson

Enter a value for x:80

The true root is 126.632

Iter X Xn+1 F(x) Error

0 80.0 108.5856 -6.1403 .1662

1 108.5856 123.9525 -1.7165 .0216

2 123.9525 126.5735 -.222 .0005

Approximate solution = 126.57352973750578

//Secant

Enter a value for x0:80

Enter a value for x1:150

The true root is 126.632

Iter Xn-1 X Xn+1 F(xn-1) F(x) F(xn+1) Error

0 150.0 80.0 135.6077 1.5892 -6.1403 .6773 .0662

1 135.6077 80.0 130.0831 .6773 -6.1403 .2719 .0265

2 130.0831 80.0 127.9596 .2719 -6.1403 .1064 .0104

3 127.9596 80.0 127.143 .1064 -6.1403 .0412 .004

Approximate solution = 127.143

//Modified Secant

Enter a value for x0:80

The true root is 126.632

Iter X Xn+1 F(x) Error

0 80.0 108.8979 -6.1403 .1629

1 108.8979 124.2039 -1.6817 .0195

2 124.2039 126.6088 -.2007 .0002

Approximate solution = 126.6088

//False Position

Enter a value for x0:80

Enter a value for x1:150

The true root is 126.632

Iter Xn-1 X Xn+1 F(xn-1) F(x) F(xn+1) Error

0 80.0 150.0 135.6077 -6.1403 1.5892 .6773 .0662

1 80.0 135.6077 130.0831 -6.1403 .6773 .2719 .0265

2 80.0 130.0831 127.9596 -6.1403 .2719 .1064 .0104

3 80.0 127.9596 127.143 -6.1403 .1064 .0412 .004

Approximate solution = 127.143

For Part B of this project I had realized my error in not picking similar starting values for my methods and decided to choose the lower end starting point to be 80 while if a second point was required I choose 150. The results shown in the graph did reflect my thoughts that the Bisection method would be the one that needs the most iterations. But the oddest thing about my results for part B was that the Secant method and the False-Position method gave me the exact same true error values and the Newton Raphson method and the Modified Secant method where almost exactly the same as well. So the graph doesn’t show as much detail as I would like, making me think I should of picked point much farther away from the root as testing data.