

Assignment: Homework Five Name: Cody Strange

Disclaimer: This is my work, not that of others

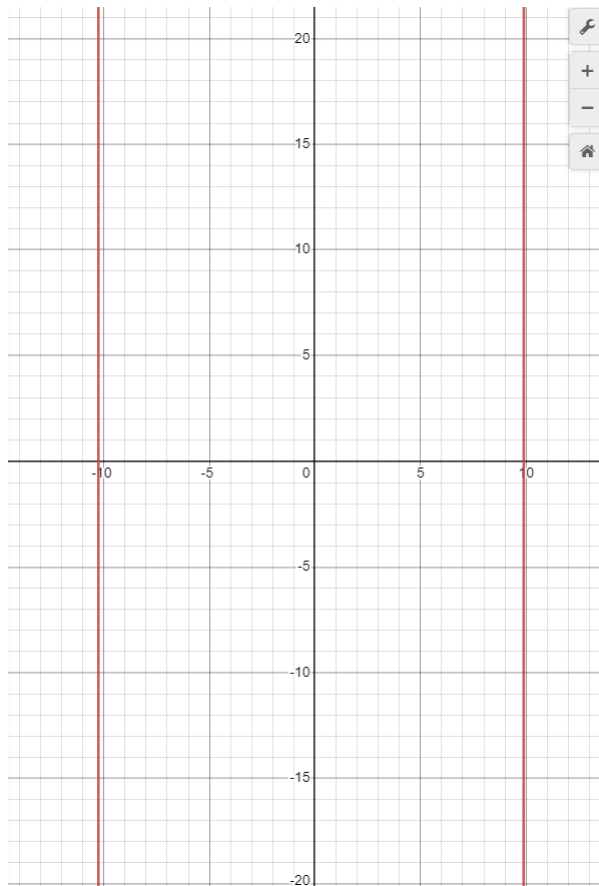
Total Score: 30 (in points, not percentage)

Problem 1 score: 10

Problem 2 score: 10

Problem 3 score: 10

1. Using the bisection method. The first step is to graph the function and see roughly where the roots are located.



The roots are located close to -10 and 10. Because there are two roots I have to create the program to be able to check for each one individually. To make this easier to implement I'll adjust the given code to take an extra parameter that checks if I am looking for the smaller root or the larger root. To find the -10 root I manually make  $x_r = x_m$  for the first iteration, after that I can treat the problem as if there was only one root and find it. The root for the one that was close to -10 is actually -10.260964380932979. To find the root closer to 10 you do the opposite, you set the  $x_l = x_m$  on the first iteration and then treat the problem as if that was the only root. The root for the one closest to 10 is actually 9.886002700947888.

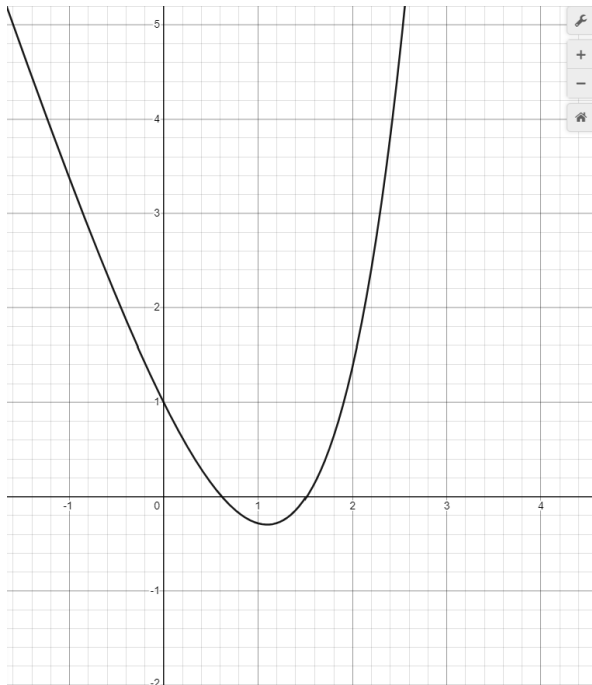
bisection.py > ...

```
1  import numpy as np
2
3  def myfunc(x):
4      return (x**4) - (3*(x**2)) + (75*x) - 10000
5
6  def bisect1(func,xl,xu,rootType,maxit=20):
7      """
8      Uses the bisection method to estimate a root of func(x).
9      The method is iterated maxit (default = 20) times.
10     Input:
11         func = name of the function
12         xl = lower guess
13         xu = upper guess
14     Output:
15         xm = root estimate
16         or
17         error message if initial guesses do not bracket solution
18     """
19     xm = (xl+xu)/2
20     if rootType == "lbr":
21         xr = xm
22     else:
23         xl = xm
24
25     for i in range(maxit-1):
26         xm = (xl+xu)/2
27         if func(xm)*func(xl)>0:
28             xl = xm
29         else:
30             xu = xm
31     return xm
32
33
34 def main():
35     print(bisect1(myfunc, -20, 20, "ubr", 1000))
36 if __name__ == "__main__":
37     main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
● PS D:\School\CS3320\HW\HW-5> & C:/Users/cody1/AppData/Local/Programs/Python/Python39/
9.886002700947888
○ PS D:\School\CS3320\HW\HW-5>
```

2. Using the bisectional method. The first step is to graph the function to get a rough estimate of where the roots are located.



```

Welcome  bisection.py x  financing.py  false_position.py
bisection.py > main
1  import numpy as np
2  import math
3  def myfunc(x):
4      return math.e**x-3*x
5
6  def bisect1(func,xl,xu,rootType,maxit=20):
7      """
8      Uses the bisection method to estimate a root of func(x).
9      The method is iterated maxit (default = 20) times.
10     Input:
11         func = name of the function
12         xl = lower guess
13         xu = upper guess
14     Output:
15         xm = root estimate
16         or
17         error message if initial guesses do not bracket solution
18     """
19     xm = (xl+xu)/2
20     if rootType == "ubr":
21         xu = xm
22     else:
23         xl = xm
24
25     for i in range(maxit-1):
26         xm = (xl+xu)/2
27         if func(xm)*func(xl)>0:
28             xl = xm
29         else:
30             xu = xm
31     return xm
32
33
34 def main():
35     print(bisect1(myfunc, 0, 1, "lbr", 1000))
36 if __name__ == "__main__":
37     main()

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```


1.5121345516578422
PS D:\School\CS3320\VM\VM-S> & C:\Users\codyl\AppData\Local\Programs\Python\Pyth
0.6190612867359451
PS D:\School\CS3320\VM\VM-S>

```

Roots are 1.5121345516578422 and 0.6190612867359451

3. Used the false position method

```

1  # financing.py >  false
2  import numpy as np
3
4  def myfunc(x):
5      return 71991 * ((x*(1+x)**84)/(((1+x)**84)-1)) - 1000
6
7  def false(func,xl,xu,maxit=20):
8      """
9      Uses the bisection method to estimate a root of func(x)
10     The method is iterated maxit (default = 20) times.
11     Input:
12         func = name of the function
13         xl = lower guess
14         xu = upper guess
15     Output:
16         xm = root estimate
17         or
18         error message if initial guesses do not bracket s
19     """
20
21     for i in range(maxit):
22         xm = xl - func(xl)*((xu-xl)/(func(xu)-func(xl)))
23         if func(xm)*func(xl)<0:
24             xl = xm
25         else:
26             xu = xm
27     return xm*1200
28
29 def main():
30     print(false(myfunc, .03, .09, 1000))
31 if __name__ == "__main__":
32     main()

```

APR = 4.47%