theorem

HOMEWORK # 03

02/07/2025

---

1 Question 1

In this question, we are asked to consider the equation $2x - 1 = \sin x$. We are then asked to find an interval containing a root r and use the **IVT** to prove that r exists. In the remaining parts, we are asked to show that the r from Part(a) is the only root of the equation, and implement code to report the approximate root $\hat{r}$ to 8 decimal places.

1.1 Root Finding

In this part, we are asked to find an interval $[a, b]$ such that it contains a root of the aforementioned equation. We first begin by defining our function.

$$f(x) = 2x - 1 - \sin x$$

We know that a root of a function occurs where $f(x) = 0$ so we can set our function equal to zero.

$$f(x) = 0$$

$$2x - 1 - \sin x = 0$$

With this, we can evaluate some behavior by plugging in values for x as shown below.

$$f(0) = 2(0) - 1 - \sin 0 = -1$$

$$f(1) = 2(1) - 1 - \sin 1 = 1 - \sin 1$$

Since $1 < \pi$, we know that $f(1)$ is positive. Since the function at changes signs between $x = 0$ and $x = 1$, we know that a root r must exist on the interval $[0, 1]$. The Intermediate Value theorem, or IVT, states that if a function $f(x)$ is continuous over an interval $[a, b]$, and $f(a)f(b) < 0$, then a root must exist within the interval.

1.2 Uniqueness of r

In order to prove that r is the only root for the function $f(x)$, consider $f'(x)$ as shown below.

$$f'(x) = 2 - \cos x$$

We know that $\cos x$ oscillates between -1 and 1. Therefore, we know that $1 \leq f'(x) \leq 3$. Therefore, we know that $f(x)$ is strictly increasing on $\mathbb{R}$. Since the function $f(x)$ is strictly increasing on $\mathbb{R}$, we know that on either side of the root r, the function will not be equal to 0.

1.3 Using bisection to approximate r

For Part (c), we are asked to use the bisection code from class to approximate r to eight correct decimal places. Using python code which can be found in the **Homework 03** folder on the GitHub repository, the root was calculated to be $r = 0.88786221$. The following calling script was used.

```
# driver fxn
def driver():
    # use routines
    f = lambda x: 2*x-1-np.sin(x)
    a = 0
    b = 1

    tol = 1e-8

    [astar,ier, count] = bisection(f,a,b,tol)
    print('the approximate root is',astar)
    print('number of iterations: ', count)
    print('the error message reads:',ier)
    print('f(astar) =', f(astar))
    return -1
```

The above calling script, with the bisection method provided, produced the following output.

```
> python3 Homework_03.py
     the approximate root is 0.8878622129559517
     number of iterations:  26
     the error message reads: 0
     f(astar) = 1.8960828462866175e-09
```

2 Question 2

This question asks questions revolving around the function $f(x) = (x - 5)^9$. We are given that the function has a root with multiplicity of 9 when $x = 5$. We are asked to use a similar technique as **1.3** using $a = 4.82$, $b = 5.2$, and $tol = 1e - 4$.

2.1 Using bisection to approximate r

Similar to **1.3**, I used the provided python implementation of the bisection method to approximate the function $f(x)$. The following calling script was used.

```
    # Question 2 calling script
    print("———————— Question 2 Output ————————")
    f2 = lambda x: (x-5)**9
    a2 = 4.82
    b2 = 5.2

    tol2 = 1e-4

    [astar2, ier2, count2] = bisection(f2,a2,b2,tol2)
    print('the approximate root is',astar2)
    print('number of iterations: ', count2)
    print('the error message reads:',ier2)
    print('f(astar1) =', f2(astar2))
```

Using the above mentioned code and calling script, the following output was produced.

```
> python3 Homework_03.py
      ———————— Question 2 Output ————————
      the approximate root is 5.000073242187501
      number of iterations:  11
      the error message reads: 0
      f(astar1) = 6.065292655789404e−38
```

As shown above, the root was found to be $r = 5.000073242187501$ which took 11 iterations to find.

2.2 Using the expanded version of $f(x)$

For Part (b), we are asked to use the same methodology for the expanded version of $f(x)$ as shown below.

$$f(x) = x^9 - 45x^8 + 900x^7 - 10500x^6 + 78750x^5 - 393750x^4 + 1312500x^3 - 2812500x^2 + 3515625x - 1953125$$

Using the above expanded version of $f(x)$, the following calling script was used.

```
    print("———————— Question 2b Output ————————")
    f3 = lambda x: (x**9)-(45*x**8)+(900*x**7)-(10500*x**6)+(78750*x**5)-(393750*x**
    [astar3,ier3,count3] = bisection(f3,a2,b2,tol2)
    print('the approximate root is',astar3)
    print('number of iterations: ', count3)
    print('the error message reads:',ier3)
    print('f(astar3) =', f3(astar3))
```

Using the above mentioned code and calling script, the following output was produced.

```
> python3 Homework_03.py
      ———————— Question 2b Output ————————
      the approximate root is 5.12875
      number of iterations:  1
      the error message reads: 0
      f(astar3) = 0.0
```

As shown above, the root was found to be $r = 5.12875$ which was found in 1 iterations.

2.3 What is happening

The way the bisection method works is as following.

1. Check to see whether the function satisfies the **IVT** (There exists a root of the function in the interval).
2. Check to see if the endpoints are not roots.
3. Calculate the midpoint and check to see if the midpoint is not a root.

If any of the three checks are *False*, the method halts and returns to the call without entering the while loop. In the case of Part (b), the first midpoint calculated was a root as $f(d) = 0$ (d is the calculated midpoint). Since the midpoint, d, was a midpoint, the method returned the value of d which was found to be 5.12875 meaning $r = 5.12875$.

3 Question 3

For Question 3, we are directed to a theorem offered in class to find an upper bound on the number of iterations in the bisection needed to approximate the solution of $f(x) = x^3 + x - 4 = 0$ lying in the interval $[1, 4]$ with an accuracy of 10^{-3}. We are then asked to find an approximation of the root using the code similar to that in Question 1 and Question 2.

3.1 Finding an upper bound on the number of iterations n

From class, we know that the number of iterations of the bisection method, n, of a function $f(x)$ on the interval $[a, b]$ where $f(a)f(b) < 0$ is constrained by an upper bound as follows.

$$n \geq \frac{\log \frac{(b-a)}{\epsilon}}{\log 2}, \text{ where } \epsilon \text{ is the tolerance}$$

Using this, we can substitute in the above values for a, b, and ϵ as shown below.

$$n \geq \frac{\log(4-1) - \log(10^{-3})}{\log(2)}$$

$$n \geq \frac{\log(3) - \log(10^{-3})}{\log(2)}$$

$$n \geq 11.55074679$$

Since n is not an integer and is greater than the above value, we can set the upper bound of n as $n \leq 12$.

3.2 Comparing results

Part (b) asks us to use the code from class similar to before to see the number of iterations required. Like before, the code from class was used. The calling script can be found below as well as the output.

```
print("———————— Question 3b Output ————————")
f4 = lambda x: x**3+x-4
a4 = 1
b4 = 4

tol4 = 1e-3

[astar4,ier4,count4] = bisection(f4,a4,b4,tol4)
print('the approximate root is',astar4)
print('number of iterations: ', count4)
print('the error message reads:',ier4)
print('f(astar3) =', f4(astar4))
```

And the output shown below.

```
> python3 Homework_03.py
    ———————— Question 3b Output ————————
    the approximate root is 1.378662109375
    number of iterations:  11
    the error message reads: 0
    f(astar3) = -0.0009021193400258198
```

As shown above, the number of iterations to approximate the root of $f(x) = x^3 + x - 4$ with a tolerance $\epsilon = 10^{-3}$ is 11 which is less than the upper bound found in Part (a).

4 Question 4

Question 4 asks about a set of functions. The question asks whether or not the given iterations will converge to a fixed point x_*. If the function does converge to p_*, we are asked to give the order of convergence as well as the rate of linear convergence where applicable.

4.1

For Part (a), we are given the following iteration and x_*.

$$x_{n+1} = -16 + 6x_n + \frac{12}{x_n},\ x_* = 2$$

Plugging in $x_* = 2$ for x_n, we get the following.

$$x_{n+1} = -16 + 12 + \frac{12}{2}$$

$$x_{n+1} = -16 + 18$$

$$x_{n+1} = 2$$

Since $x_{n+1} = 2$, we know that x_* is a fixed point. To find the convergence, we can take the derivative of the function $f(x) = -16 + 6x - \frac{12}{x}$ as shown below.

$$f'(x) = 6 - \frac{12}{x^2}$$

Now evaluating the function at x_* by plugging in $x_* = 2$, we get the following.

$$f'(x_*) = 6 - 3 = 3$$

Since $f'(x_*) \geq 1$, we know the iteration does not converge around $x_* = 2$.

4.2

For Part (b), we are given the following iteration and x_*.

$$x_{n+1} = \frac{2}{3}x_n + \frac{1}{x_n^2},\ x_* = 3^{1/3}$$

Doing the same as before, we get the following.

$$x_{n+1} = \frac{2}{3}3^{1/3} + \frac{1}{(3^{1/3})^2}$$

$$x_{n+1} = \frac{2}{3^{2/3}} + \frac{1}{(3^{2/3})}$$

$$x_{n+1} = \frac{3}{3^{2/3}}$$

$$x_{n+1} = 3^{1/3}$$

Since $x_{n+1} = x_*$, we know x_* is a fixed point. Like before, we will now consider the derivative of the iteration function as shown below.

$$f'(x) = \frac{2}{3} - \frac{2}{x^3}$$

Plugging in $x_* = 3^{1/3}$ for x_n, we get the following.

$$f'(x) = \frac{2}{3} - \frac{2}{(3^{1/3})^3}$$

$$f'(x) = \frac{2}{3} - \frac{2}{3}$$

$$f'(x) = 0$$

Since $f'(x) = 0$, we now that the iteration converges to x_* faster than linear. To check for quadratic convergence, we can look at the second derivative as shown below.

$$f''(x) = \frac{6}{x^4}$$

Like before, we can substitute in x_*.

$$f''(x_*) = \frac{6}{(3^{1/3})^4}$$

$$f''(x_*) = \frac{2*3}{3^{4/3}}$$

$$f''(x_*) = \frac{2}{3^{1/3}}$$

Since, $f''(x_*) \neq 0$, we know that the iteration is quadratically convergent.

4.3

Like in the other parts, we are given an iteration as shown below.

$$x_{n+1} = \frac{12}{1+x_n},\ x_* = 3$$

Plugging in x_*, we get the following.

$$x_{n+1} = \frac{12}{1+3}$$

$$x_{n+1} = \frac{12}{4}$$

$$x_{n+1} = 3$$

Since $x_{n+1} = x_*$, we know that x_* is a fixed point. To evaluate convergence, we can take the derivative of the iteration function.

$$f'(x) = \frac{-12}{(1+x)^2}$$

Plugging in x_*, we get the following.

$$f'(x) = \frac{-12}{(1+3)^2}$$

$$f'(x) = \frac{-12}{(4)^2}$$

$$f'(x) = \frac{-12}{16}$$

$$f'(x) = \frac{-3}{4}$$

Since $|f'(x)| < 1$, we know that the iteration converges linearly with an order of convergence of $\alpha = \frac{3}{4}$.

5 Question 5

Question 5 gives us the following scalar equation and asks that the roots to be determined with at least 10 accurate digits. That is, roots with a relative error less than $0.5 * 10^{-n}$.

$$x - 4\sin(2x) - 3 = 0$$

5.1 Plotting in python

For Part (a), we are asked to plot the function $f(x) = x - 4\sin(2x) - 3$. Using python, numpy, and the pyplot library in matplotlib, the following figure was created. As shown in the graph above,

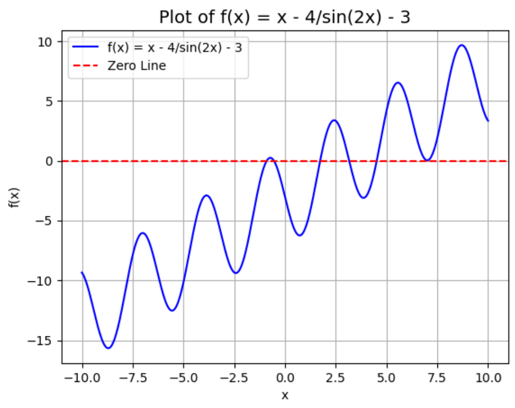

Figure 1: Plot of $f(x) = x - 4\sin(2x) - 3$

there are 4 roots of the scalar equation above.

5.2 Computing roots using FPI

For Part (b), we were asked to write a program to compute the roots using the following fixed point iteration.

$$x_{n+1} = -\sin(2x_n) + \frac{5x_n}{4} - \frac{3}{4}.$$

The roots are required to be calculated to 10 correct digits. That being said, I was struggling to fully implement the FPI method to work properly and to the required degree of accuracy. I will hopefully be able to get it figured out shortly. The code can be found in my GitHub repository. I will make sure to leave a comment in the code with the roots that I find.