APPM 4600 HW #3 - Gustav Cedergrund

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

a) An interval $[a,b]$ on which equation has root r is on $[0,\frac{\pi}{2}]$ b/c

$x=0 \Rightarrow 2(0)-1-\sin(0) = 0-1-0 = \underline{-1} < 0$

$x=\frac{\pi}{2} \Rightarrow 2(\frac{\pi}{2})-1-\sin(\frac{\pi}{2}) = \pi-1-1 = \underline{\pi-2} > 0$

$\therefore$ by IVT, as equation is continuous and $-1 < 0 < \pi-2$, then $\exists$ root 'r' s.t. $\underline{2(r)-1-\sin(r)=0}$.

b) To prove that r from (a) is only root, we can look to 1st derivative,

$\frac{d}{dx}(2x-1-\sin(x)) = 2-\cos(x) > 0 \quad \forall x \in \mathbb{R}$.

$\therefore$ the equation is monotonically increasing $\forall x \in \mathbb{R}$ s.t. it continues to increase forever. Thus, 'r' is the only root b/c the equation only goes past 0 once.

c) Using bisection code from class, we can approximate r to 8 correct dec. places as

$\boxed{r = 0.88786221}$ calling script w/ # of iterations attached on next page

---

2) function: $f(x)=(x-5)^9$ is viable candidate for bisection

a) bisection method + results for $f(x)$ attached

b) bisection method + results for expanded $f(x)$ attached

c) the difference in computation + result for bisection in this example is caused by error in computation for the function. The second version of the function that is expanded becomes very inaccurate as a result of many computations being done near the root, so it is unable to check if the guess for the root at each iteration is close to the real root. As a result, it produces 5.128 rather than ~5.0 b/c the error propagation causes an input of 5.128 result in output under the tolerance of root, while actually, a root does not exist there.

Problem 1 Script + Output:

```python
import numpy as np

# imported bisection method from class example
# with added print statements at each iteration
# and tolerance checking for relative error
# instead of absolute error


def driver():
    # function decleration
    f = lambda x: 2 * x - 1 - np.sin(x)

    # endpoints for parts a,b,c
    a = 0
    b = np.pi / 2

    # tolerance for 8 correct digits
    tol = 0.5 * 10**-8

    print("(1):\n")
    [astar, ier] = bisection(f, a, b, tol)
    print("the approximate root is", astar)
    # print("the error message reads:", ier)
    print("f(root) =", f(astar))
    print("\n")

    return
```

```
(1):

iteration:  1  | curr_root =  1.1780972450961724
iteration:  2  | curr_root =  0.9817477042468103
iteration:  3  | curr_root =  0.8835729338221293
iteration:  4  | curr_root =  0.9326603190344698
iteration:  5  | curr_root =  0.9081166264282996
iteration:  6  | curr_root =  0.8958447801252145
iteration:  7  | curr_root =  0.889708856973672
iteration:  8  | curr_root =  0.8866408953979006
iteration:  9  | curr_root =  0.8881748761857863
iteration:  10  | curr_root =  0.8874078857918435
iteration:  11  | curr_root =  0.8877913809888149
iteration:  12  | curr_root =  0.8879831285873006
iteration:  13  | curr_root =  0.8878872547880577
iteration:  14  | curr_root =  0.8878393178884363
iteration:  15  | curr_root =  0.887863286338247
iteration:  16  | curr_root =  0.8878513021133416
iteration:  17  | curr_root =  0.8878572942257943
iteration:  18  | curr_root =  0.8878602902820206
iteration:  19  | curr_root =  0.8878617883101338
iteration:  20  | curr_root =  0.8878625373241904
iteration:  21  | curr_root =  0.8878621628171621
iteration:  22  | curr_root =  0.8878623500706763
iteration:  23  | curr_root =  0.8878622564439191
iteration:  24  | curr_root =  0.8878622096305406
iteration:  25  | curr_root =  0.8878622330372299
iteration:  26  | curr_root =  0.8878622213338853
iteration:  27  | curr_root =  0.8878622154822129
iteration:  28  | curr_root =  0.8878622125563768
Number of iterations:  28

the approximate root is 0.8878622125563768
f(root) = 1.3490933925552895e-09
```

Problem 2 Output:

```
(2):

normal version (part a):
iteration:  1  | curr_root =  4.915
iteration:  2  | curr_root =  4.9625
iteration:  3  | curr_root =  4.98625
iteration:  4  | curr_root =  4.998125
iteration:  5  | curr_root =  5.0040625
iteration:  6  | curr_root =  5.00109375
iteration:  7  | curr_root =  4.999609375
iteration:  8  | curr_root =  5.000351562500001
iteration:  9  | curr_root =  4.9999804687500005
iteration:  10  | curr_root =  5.000166015625
iteration:  11  | curr_root =  5.000073242187501
Number of iterations:  11

the approximate root is 5.000073242187501
f(root) = 6.065292655789404e-38


expanded version (part b):
iteration:  1  | curr_root =  5.105
iteration:  2  | curr_root =  5.1525
iteration:  3  | curr_root =  5.12875
Number of iterations:  3

the approximate root is 5.12875
f(root) = 9.721317766824793e-09
```

3)

a) Using theorem 2.1, we know that bisection method approximates zero p of function f w/

$$|p_n - p| < \frac{b-a}{2^n}, \text{ when } n \geq 1$$

$\therefore$ w/ $a=1, b=4$ on $f(x) = x^3 + x - 4$, to predict w/ accuracy of 10^{-3}

$$\hookrightarrow \quad 10^{-3} < \frac{4-1}{2^n}$$

$$2^n < 3 \cdot 10^3$$

$$\log_2(3000) = 11.55 < n \quad \text{so } n \geq 12$$

We need at most $\boxed{12}$ iterations to find p

b) Using bisection code (output attached), it takes 11 iterations to get root w/ this accuracy. This is less than upper bound, which checks out.

---

4) Evaluate convergence of following iterations

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

Let $x_{n+1} = g(x_n) = -16 + 6x_n + \frac{12}{x_n}$

first, $x_* = 2$ is a fixed point b/c $g(x_*) = -16 + 6(2) + \frac{12}{2} = 2 = x_*$

However, as $g'(x_n) = 6 - 12x_n^{-2}$ evaluated at x_* is

$$g'(2) = 6 - \frac{12}{2^2} = 6 - 3 = 3$$

$|g'(x_*)| \not< 1$, so by Thm 2.4 x_{n+1} does not converge to x_*

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

Let $x_{n+1} = g(x_n)$. $x_* = 3^{1/3}$ is a fixed point as $g(3^{1/3}) = \frac{2}{3}(3^{1/3}) + \frac{1}{3^{2/3}} = 3^{1/3}$

And $g'(x_n) = \frac{2}{3} - 2x_n^{-3}$ s.t $g'(3^{1/3}) = \frac{2}{3} - 2(3^{1/3})^{-3} = 0 < 1$

s.t $\{x_n\}$ converges when x_0 sufficiently close to x_* by Thm 2.4.

However as $g'(3^{1/3}) = 0$ rate of convergence is not linear.

Note: $\lim_{n\to\infty} \frac{|x_{n+1} - x_*|}{|x_n - x_*|^2} = \lim_{n\to\infty} \frac{|g''(x_n)|}{2}$, and b/c $g''(x_n) = 6x_n^{-4}$

$$\frac{g''(x_*)}{2} = \frac{6}{2 \cdot 3^{4/3}} = 3^{-1/3} < 1. \quad \therefore \boxed{\text{order of convergence is 2}}$$

4) contined

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

Let $x_{n+1} = g(x_n)$, then $g(3) = \frac{12}{1+3} = 3$ $\therefore$ x_* is fixed point,

Also, as $g'(x_n) = -12/(1+x_n)^2$ at x_* is $|g'(3)| = |-12/(1+3)^2| = 3/4$ and $0 < 3/4 < 1$. The sequence converges linearly w/ order 1 and asymptotic constant $3/4 = \lambda$.

5) Consider scalar equation $x - 4\sin(2x) - 3 = 0$

a) Plot for $f(x) = x - 4\sin(2x) - 3$ attached.
There are 5 zero crossings

b) Program output attached
Empirically only 2/5 roots can be found.
↳ 1 @ -0.5444424006, 1 @ 3.1618264865
The other roots can not be found by fixed root method, b/c, $|f'(x)|$ at fixed points are >1 s.t Fixed Point Theorem 2.4 doesn't guarantee the fixed point method work. Simply, this is b/c successive iterations will grow away from root.

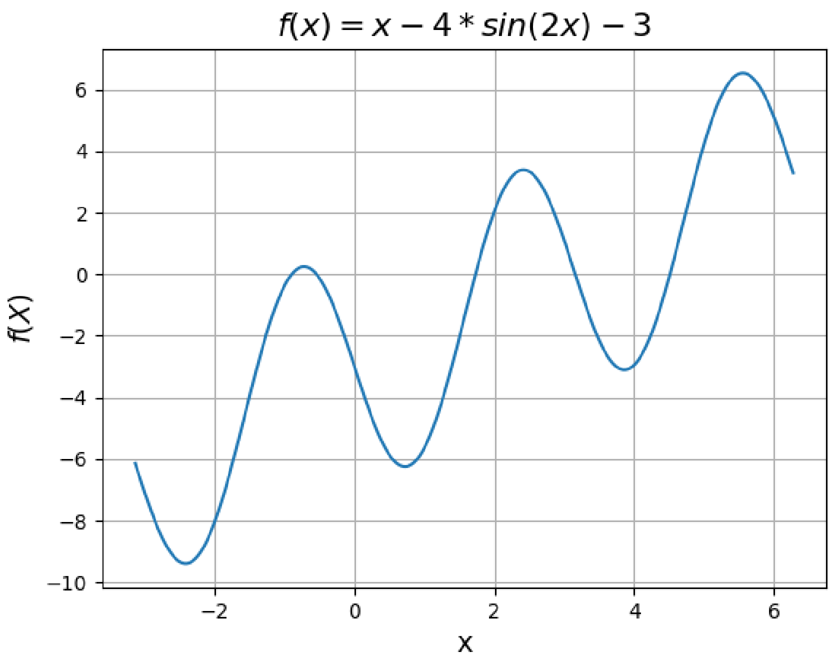
Problem 3 Output:

```
(3):

approximation:
iteration:  1  | curr_root =  1.75
iteration:  2  | curr_root =  1.375
iteration:  3  | curr_root =  1.5625
iteration:  4  | curr_root =  1.46875
iteration:  5  | curr_root =  1.421875
iteration:  6  | curr_root =  1.3984375
iteration:  7  | curr_root =  1.38671875
iteration:  8  | curr_root =  1.380859375
iteration:  9  | curr_root =  1.3779296875
iteration:  10  | curr_root =  1.37939453125
iteration:  11  | curr_root =  1.378662109375
Number of iterations:  11

the approximate root is 1.378662109375
f(root) = -0.0009021193400258198
```

Problem 5 Output:
(a):



(b):

```
(5):

looking for root at x=-0.898 with x0 = -0.9
the approximate fixed point is: -2761829351.191013
f(fixed_point): -3452286689.3512335
Error message reads: 1


looking for root at x=-0.544 with x0 = -0.4
the approximate fixed point is: -0.5444424006756098
f(fixed_point): -0.5444424006790539
Error message reads: 0


looking for root at x=1.732 with x0 = 1.7
the approximate fixed point is: -0.5444424006869433
f(fixed_point): -0.5444424006827152
Error message reads: 0


looking for root at x=3.162 with x0 = 3
the approximate fixed point is: 3.161826486605397
f(fixed_point): 3.1618264865119454
Error message reads: 0


looking for root at x=4.518 with x0 = 4.5
the approximate fixed point is: 3.161826486613379
f(fixed_point): 3.161826486505972
Error message reads: 0
```