HOMEWORK 3
APPM 4600
MATVIY KHOTYAINTSEV

1.a $2x - 1 = \sin x$

We can define

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

for which $f(x^*) = 0$

for some $x^* \in [a, b]$.

- $f(x)$ is continuous for all $x \in \mathbb{R}$ since its composed of continuous parts

MVP states that if $f(a) \cdot f(b) < 0$

$\exists$ a x^* st $f(x^*) = 0$.

Let's evaluate $a = 0$ and $b = 1$

$f(0) = -1$

$f(1) = 1 - \sin(1) \simeq 0,1585$.

Since $f(a) \cdot f(b) < 0$ $\exists$ a x^* st

$f(x^*) = 0$ ∎

b For this function we can prove that there exists only 1 root on $\mathbb{R}$ by analyzing the derivative:

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

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

Since $\cos x \in [-1, 1]$ for all $x \in \mathbb{R}$

$\Rightarrow f'(x) \in [1, 3]$ for all $x \in \mathbb{R}$

Since the derivative is always positive There can be only one root. Strictly increasing.

```
# import libraries
import numpy as np

def driver():

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

#    f = lambda x: np.sin(x)
#    a = 0.1
#    b = np.pi+0.1

    tol = 1e-9
    print('')
    [astar,ier,count] = bisection(f,a,b,tol)
    print('the approximate root is',astar)
    print('the error message reads:',ier)
    print('f(astar) =', f(astar))

    print('Iterations:', count)

# define routines
def bisection(f,a,b,tol):

#    Inputs:
#     f,a,b       - function and endpoints of initial interval
#      tol  - bisection stops when interval length < tol

#    Returns:
#      astar - approximation of root
#      ier   - error message
#            - ier = 1 => Failed
#            - ier = 0 == success

#     first verify there is a root we can find in the interval

    fa = f(a)
    fb = f(b);
    if (fa*fb>0):
       ier = 1
       astar = a
       return [astar, ier, count]

#   verify end points are not a root
    if (fa == 0):
      astar = a
      ier =0
      return [astar, ier, count]

    if (fb ==0):
      astar = b
      ier = 0
      return [astar, ier, count]
```

```
    count = 0
    d = 0.5*(a+b)
    while (abs(d-a)> tol):
      fd = f(d)
      if (fd ==0):
        astar = d
        ier = 0
        return [astar, ier, count]
      if (fa*fd<0):
         b = d
      else:
        a = d
        fa = fd
      d = 0.5*(a+b)
      count = count +1
#      print('abs(d-a) = ', abs(d-a))

    astar = d
    ier = 0
    return [astar, ier, count]

driver()
```

Terminal print:

```
the approximate root is 0.8878622120246291
the error message reads: 0
f(astar) = 6.211691161439603e-10
Iterations: 29
MK@cu-tcom-7-10 homework %
```

2 c. for (a) $x^* = 5,000073...$

(b) $x^* = 5,12875$

For (b) the first midpoint is $c = 5,01$ and calculates $f(5,01) = -2,495 \cdot 10^{-5}$, which is incorrect and prabably depends on errors.

```
# import libraries
import numpy as np

def driver():

# use routines
    f = lambda x: (x**9 - 45*x**8 + 900*x**7 - 10500*x**6 + 78750*x**5
            - 393750*x**4 + 1312500*x**3 - 2812500*x**2
            + 3515625*x - 1953125)
    a = 4.82
    b = 5.2

#    f = lambda x: np.sin(x)
#    a = 0.1
#    b = np.pi+0.1

    tol = 1e-3
    print('')
    [astar,ier,count] = bisection(f,a,b,tol)
    print('the approximate root is',astar)
    print('the error message reads:',ier)
    print('f(astar) =', f(astar))

    print('Iterations:', count)

# define routines
def bisection(f,a,b,tol):

#    Inputs:
#     f,a,b       - function and endpoints of initial interval
#      tol  - bisection stops when interval length < tol

#    Returns:
#      astar - approximation of root
#      ier   - error message
#            - ier = 1 => Failed
#            - ier = 0 == success

#     first verify there is a root we can find in the interval

    fa = f(a)
    fb = f(b);
    if (fa*fb>0):
       ier = 11
       astar = a
       return [astar, ier, 0]

#   verify end points are not a root
    if (fa == 0):
      astar = a
      ier =0
      return [astar, ier, 0]

    if (fb ==0):
      astar = b
```

```
    count = 0
    d = 0.5*(a+b)
    while (abs(d-a)> tol):
      fd = f(d)
      if (fd ==0):
        astar = d
        ier = 0
        return [astar, ier, count]
      if (fa*fd<0):
         b = d
      else:
        a = d
        fa = fd
      d = 0.5*(a+b)
      count = count +1
#      print('abs(d-a) = ', abs(d-a))

    astar = d
    ier = 0
    return [astar, ier, count]

driver()
```

Terminal print

```
the approximate root is 5.000073242187501
the error message reads: 0
f(astar) = 6.065292655789404e-38
Iterations: 11
```

```
# import libraries
import numpy as np

def driver():

# use routines
    f = lambda x: (x**9 - 45*x**8 + 900*x**7 - 10500*x**6 + 78750*x**5
            - 393750*x**4 + 1312500*x**3 - 2812500*x**2
            + 3515625*x - 1953125)
    a = 4.82
    b = 5.2

#    f = lambda x: np.sin(x)
#    a = 0.1
#    b = np.pi+0.1

    tol = 1e-3
    print('')
    [astar,ier,count] = bisection(f,a,b,tol)
    print('the approximate root is',astar)
    print('the error message reads:',ier)
    print('f(astar) =', f(astar))

    print('Iterations:', count)

# define routines
def bisection(f,a,b,tol):

#    Inputs:
#     f,a,b       - function and endpoints of initial interval
#      tol  - bisection stops when interval length < tol

#    Returns:
#      astar - approximation of root
#      ier   - error message
#            - ier = 1 => Failed
#            - ier = 0 == success

#     first verify there is a root we can find in the interval

    fa = f(a)
    fb = f(b);
    if (fa*fb>0):
       ier = 11
       astar = a
       return [astar, ier, 0]

#   verify end points are not a root
    if (fa == 0):
      astar = a
      ier =0
      return [astar, ier, 0]

    if (fb ==0):
      astar = b
```

```
      ier = 0
      return [astar, ier, 0]

    count = 0
    d = 0.5*(a+b)
    while (abs(d-a)> tol):
      fd = f(d)
      if (fd ==0):
         astar = d
         ier = 0
         return [astar, ier, count]
      if (fa*fd<0):
         b = d
      else:
         a = d
         fa = fd
      d = 0.5*(a+b)
      count = count +1
#      print('abs(d-a) = ', abs(d-a))

    astar = d
    ier = 0
    return [astar, ier, count]

driver()
```

Terminal print

```
the approximate root is 5.12875
the error message reads: 0
f(astar) = 0.0
Iterations: 3
```

3.a $N_{it} \geq \frac{\log(3-a) - \log tol}{\log 2}$

$a = 1$
$b = 4$
$tol = 10^{-3}$

$N_{it} \geq \frac{\log 3 - \log 10^{-3}}{\log 2} \approx 11,552$

N_{it} must be integer and in this case includes 12. Finally:

$N_{it} \geq 12.$

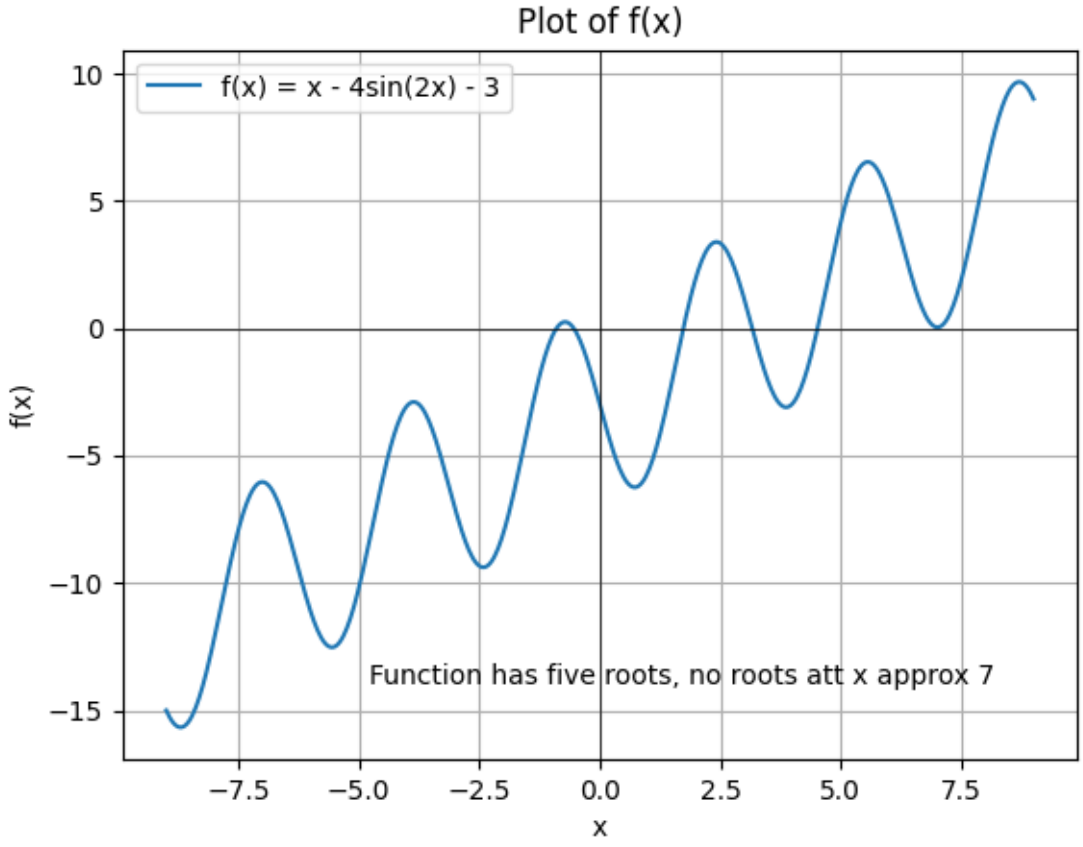
3.b The bisection code used 11 which is less than the specified minimum of 12 iterations; Could have to do with width of interval for final tolerance. Where our code uses endpoint as approx. instead of midpoint.

```python
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return x - 4 * np.sin(2 * x) - 3

# Plot the function
x_vals = np.linspace(-9, 9, 500)
y_vals = f(x_vals)

plt.plot(x_vals, y_vals, label="f(x) = x - 4sin(2x) - 3")
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.title("Plot of f(x)")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.text(-4.8,-14,"Function has five roots, no roots att x approx 7")
plt.grid(True)
plt.legend()
plt.show()
```


Plot of f(x)
f(x) = x - 4sin(2x) - 3
10
5
0
−5
−10
−15
f(x)
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
x
Function has five roots, no roots att x approx 7