APPM 4600 — HOMEWORK # 3 Solutions

1. Consider the equation $2x - 1 = \sin x$.

   (a) Find a closed interval $[a, b]$ on which the equation has a root r, and use the Intermediate Value Theorem to prove that r exists.

   (b) Prove that r from (a) is the only root of the equation (on all of $\mathbb{R}$).

   (c) Use the bisestion code from class (or your own) to approximate r to eight correct decimal places. Include the calling script, the resulting final approximation, and the total number of iterations used.

   Soln:

   (a) Let $f(x) = 2x - 1 - \sin x$. Our goal is to find the roots of $f(x)$. Plotting the function we observe that the root is between 0 and $\pi/2$; i.e. the root lies in the interval $[0, \pi/2]$. Now we will prove that this is true.
   $f(0) = -1$ and $f(2) = \pi - 1 - \sin(\pi/2) = \pi - 2 > 0$. Thus by the Intermediate Value Theorem, there exists a point $c \in [0, \pi/2]$ such that $f(c) = 0$.

   (b) To show uniqueness, we will assume there are two roots, r and s, $r \neq s$; $f(r) = 0$ and $f(s) = 0$.
   This means that $2r-1-\sin r+2s-1-\sin s = 0$. Rewriting we find $2(r+s) = \sin r+\sin s$. This is impossible since $\sin r < r < 2r$ and $\sin s < s < 2s$. This is our contradiction.

   (c) Driver code

```
import mypkg.prini as prini
import mypkg.my2DPlotB
import scipy
import numpy as np
import math

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

    tol = 1e-8
    a = 0
    b = math.pi/2
    Nmax = 100

    [astar,ier,its] = bisection(f,a,b,tol,Nmax)
    tmp = prini('real','the approximate root is',astar)
    tmp.print()
    tmp = prini('real','the number of iterations is:',its)
    tmp.print()
    tmp = prini('inter','the error message reads:',ier)
    tmp.print()
```

```
def bisection(f,a,b,tol,Nmax):
    '''
    Inputs:
      f,a,b       - function and endpoints of initial interval
      tol, Nmax   - bisection stops when interval length < tol
                  - or if Nmax iterations have occured
    Returns:
      astar - approximation of root
      ier   - error message
            - ier = 1 => cannot tell if there is a root in the interval
            - ier = 0 == success
            - ier = 2 => ran out of iterations
            - ier = 3 => other error ==== You can explain
    '''

    '''     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]

    ''' verify end point is not a root '''
    if (fa == 0):
      astar = a
      ier =0
      return [astar, ier]

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

    count = 0
    while (count < Nmax):
      c = 0.5*(a+b)
      fc = f(c)

      if (fc ==0):
        astar = c
        ier = 0
        return [astar, ier,count]

      if (fa*fc<0):
         b = c
```

```
          elif (fb*fc<0):
            a = c
            fa = fc
          else:
            astar = c
            ier = 3
            return [astar, ier]

          if (abs(b-a)/abs(a)<tol):
            astar = a
            ier =0
            return [astar, ier,count]

          count = count +1

        astar = a
        ier = 2
        return [astar,ier,count]




    if __name__ == '__main__':
        # run the drivers only if this is called from the command line
        driver()
```

Results: $r = 0.88786220963054$ and $iter = 27$

2. The function $f(x) = (x-5)^9$ has a root (with multiplicity 9) at $x = 5$ and is monotonically increasing (decreasing) for $x > 5$ ($x < 5$) and should thus be a suitable candidate for your function above. Use `a=4.82` and `b=5.2` and `tol = 1e-4` and use `bisection` with:
   (a) $f(x) = (x-5)^9$.
   (b) The expanded expanded version of $(x-5)^9$, that is, $f(x) = x^9 - 45x^8 + \ldots - 1953125$.
   (c) Explain what is happening.

   Soln:

   (a) The driver code reads

```
    def driver():
        f = lambda x: (x-5)**9

        tol = 1e-4
        a =  4.82
        b = 5.2
        [astar,ier] = bisection(f,a,b,tol,Nmax)
        tmp = prini('real','the approximate root is',astar)
        tmp.print()
        tmp = prini('inter','the error message reads:',ier)
        tmp.print()
```

The code returns $r = 5.000073242187501$ after 2 iterations.

(b) The driver code reads

```
def driver():
    f = lambda x: -1953125 + 3515625*x - 2812500*x**2 + \
          1312500*x**3 - 393750*x**4 -78750*x**5 \
          - 10500*x**6 + 900*x**7 - 45*x**8 + x**9

    tol = 1e-4
    a =  4.82
    b = 5.2

    [astar,ier] = bisection(f,a,b,tol,Nmax)
    tmp = prini('real','the approximate root is',astar)
    tmp.print()
    tmp = prini('inter','the error message reads:',ier)
    tmp.print()
```

Bissection fails.

(c) Using the expanded form returns the values $f(a) = -4.097469686693044e{+}08$ and $f(b) = -5.988213503999996e + 08$. This is clearly not correct. It is the result of evaluating the expanded form in floats. A value becomes so large that it cannot accurately do addition and subtraction of the pieces.

3. (a) Use a theorem from class (Theorem 2.1 from text) to find an upper bound on the number of iterations in the bissection needed to approximate the solution of $x^3 + x - 4 = 0$ lying in the interval $[1, 4]$ with an accuracy of 10^{-3}.

(b) Find an approximation of the root using the bisection code from class to this degree of accuracy. How does the number of iterations compare with the upper bound you found in part (a)?

Soln:

(a) We want for $\frac{b-a}{2^{N_{\max}}} < 10^{-3}$. In other words we need to find a value of $N_{\max}$ such that $\frac{3}{2^{N_{\max}}} < 10^{-3}$. After a little algebra, we find $\frac{\log(3/10^{-3}}{\log(2)} < N_{\max}$. $\frac{\log(3/10^{-3}}{\log(2)} = 11.550746785383243$. I would take $N_{\max} = 13$ for good measure.

(b) The driver code reads

```
def driver():
    f = lambda x: x**3+x-4
```

```
tol = 1e-3
a =  1
b = 4

[astar,ier] = bisection(f,a,b,tol,Nmax)
tmp = prini('real','the approximate root is',astar)
tmp.print()
tmp = prini('inter','the error message reads:',ier)
tmp.print()
```

The algorithm converges in two iterations to $r = 1.378662109375000$. This is well below the number we found in part a. This makes sense. It is an upper bound.

4. **Definition 1** *Suppose $\{p_n\}_{n=0}^{\infty}$ is a sequence that converges to p with $p_n \neq p$ for all n. If there exists postive constants λ and α such that*

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^{\alpha}} = \lambda$$

then $\{p_n\}_{n=1}^{\infty}$ converges to p with an order α and asympotic error constant λ. If $\alpha = 1$ and $\lambda < 1$ then the sequence converges linearly. *If $\alpha = 2$, the sequence is* quadratically *convergent.*

Which of the following iterations will converge to the indicated fixed point x_* (provided x_0 is sufficiently close to x_*)? If it does converge, give the order of convergence; for linear convergence, give the rate of linear convergence.

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

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

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

Soln:

i. $g(x) = -16+6x+12/x$ We need $|g'(x_*)| < 1$ for the iteration to converge. The derivative is $g'(x) = 6 - 12/x^2$. So $|g'(x_*)| = |6 - 12/x_*^2| = 6 - 3 > 1$. Therefore the iteration will not converge.

ii. $g(x) = 2/3x + 1/x^2$. We need $|g'(x_*)| < 1$ for the iteration to converge. The derivative is $g'(x) = 2/3 - 2/x^3$. So $|g'(x_*)| = |2/3 - 2/3| = 0$. Therefore the iteration will converge. In order to determine the order of convergence, lets look at the Taylor theorem with the expansion about x_*, see that there exist an α in the interval between α and x such that

$$\begin{aligned} g(x) &= g(x_*) + g'(x_*)(x - x_*) + \frac{g''(\alpha)}{2!}(x - x_*)^2 \\ &= x_* + 0 + \frac{g''(\alpha)}{2!}(x - x_*)^2 \end{aligned}$$

So

$$x_{n+1} - x_* = \frac{g''(\alpha_n)}{2!}(x_n - x_*)^2$$

Rewriting this in terms of order we get

$$\lim_{n\to\infty} \frac{|x_{n+1} - x_*|}{|x_n - x_*|^2} = \lim \frac{g''(\alpha_n)}{2!} = \frac{g''(x_*)}{2!}$$

Thus the method is second order.

iii. Let $g(x) = \frac{12}{1+x}$. Thus $g'(x) = -\frac{12}{(1+x)^2}$. Plugging in x_*, we find $|g'(x_*)| = 12/16 \leq 1$. Thus the iteration converges linearly.

To determine the constant, we use Taylor's theorem taking the expansion about x_* which states there exist an α_k between x_* and x_k such that

$$g(x_k) = g(x_*) + g'(\alpha_k)(x_* - x_k).$$

We know $g(x_*) = x_*$, $g(x_k) = x_{k+1}$ and $\lim_{k\to\infty} x_k = x_*$.
Thus $\lim_{k\to\infty} \frac{|x_{k+1}-x_*|}{|x_k-x_*|} = \lim_{k\to\infty} |g'(\alpha_k)| = 12/16$.

5. All the roots of the scalar equation

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

are to be determined with at least 10 accurate digits[1].

(a) Plot $f(x) = x - 4\sin(2x) - 3$ (using your `Python` toolbox). All the zero crossings should be in the plot. How many are there?

(b) Write a program or use the code from class to compute the roots using the fixed point iteration

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

Use a stopping criterium that gives an answer with ten correct digits. (*Hint: you may have to change the error used in determing the stopping criterion.*) Find, empirically which of the roots that can be found with the above iteration. Give a theoretical explanation.

Soln:

(a)

(b) The driver code is below.

```
import mypkg.prini as prini
import numpy as np

def driver():
```

[1]n accurate digits is equivalent to a relative error smaller than 0.5×10^{-n}.

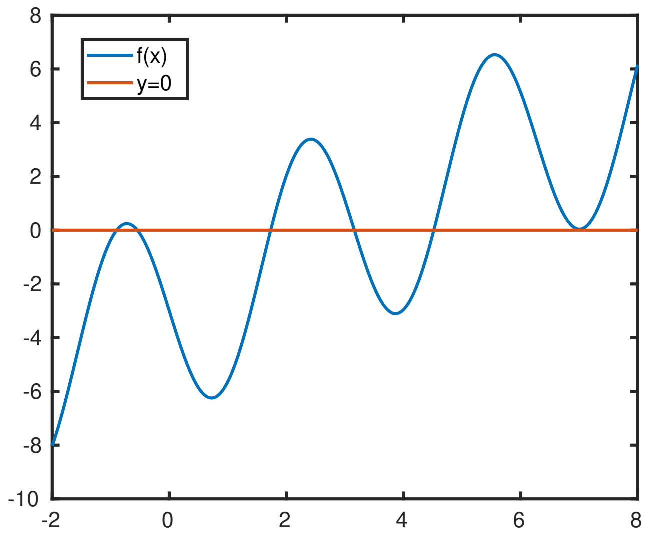

Figure 0.1: Plot of $f(x) = x - 4\sin(2x) - 3$.

```
f = lambda x: -np.sin(2*x)+5*x/4-3/4

Nmax = 100
tol = 1e-9

p0 = 3.2
[pstar,ier] = fixedpt(f,p0,tol,Nmax)
tmp = prini('real','the approximate fixed point is:',pstar)
tmp.print()
tmp = prini('real','f(pstar):',f(pstar))
tmp.print()
tmp = prini('inter','Error message reads:',ier)
tmp.print()
```

If $g(x) = -\sin(2x) + 5x/4 - 3/4$, $g'(x) = 5/4 - 2\cos(2x)$. A plot of $g'(x)$ is given in figure 0.2.

The first root of $f(x)$ is near -0.9. There $|g'(x)|$ is larger than 1 so the fixed point iteration is not going to converge to that root.

The second root of $f(x)$ is near -0.5. There $|g'(x)|$ is less than 1 so the fixed point iteration can converge to this root.

The next root of $f(x)$ is near 1.7. There $|g'(x)|$ is larger than 1 so the fixed point iteration is not going to converge to this root.

The fourth root of $f(x)$ is near 3.2. There $|g'(x)|$ is less than 1 so the fixed point iteration can converge to this root.

The final root of $f(x)$ is near 4.5. There $|g'(x)|$ is greater than 1 so the fixed point iteration cannot converge to this root.

Taking the initial guess of $p_0 = -0.3$, the fixed point iteration converges to $p^* = -5.4444240093562979e - 01$ in 17 iterations.

Take the initial guess of $p_0 = 3.2$, the fixed point iteration converges to $p^* = 3.1618264876109521$ in 60 iterations.

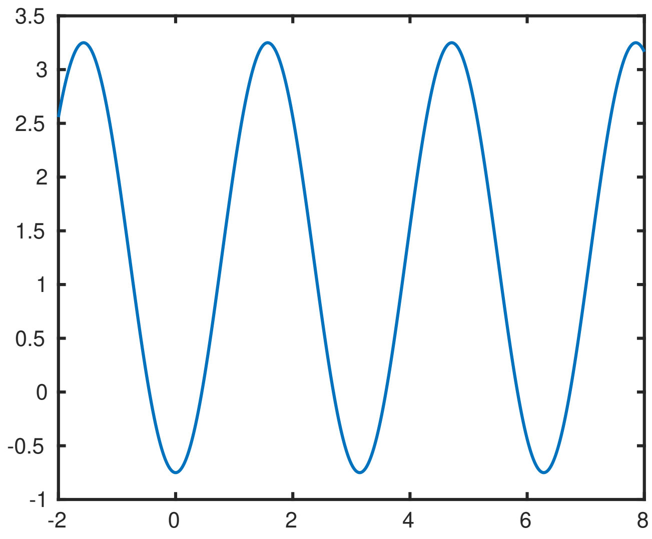

Figure 0.2: Plot of $g'(x) = 5/4 - 2 * \cos(2x)$.