APPM 4600 Homework 3

20 September 2024

1. (a) We have the continuous function $f(x) = 2x - 1 - \sin x$. Observe that

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

and that

$$f(\frac{\pi}{2}) = 2(\frac{\pi}{2}) - 1 - \sin\frac{\pi}{2} = \pi - 2 > 0.$$

By the intermediate value theorem, $f(x)$ has a root on the region $x \in (0, \frac{\pi}{2})$.

(b) Observe that $f(x)$ has derivative

$$f'(x) = 2 - cos(x) > 0 \ \forall x \in \mathbb{R},$$

that is, f is monotonically increasing. Thus, f has only one root on $\mathbb{R}$.

(c) The bisection code adapted from that provided in class is included below. Bisection yields an approximate root of

$$r \approx 0.88786221$$

```
#! /usr/bin/env python3
# APPM 4600 Homework 3 Question 1

import numpy as np
import math

def question1():

    f = lambda x: 2*x-1-np.sin(x)
    a = 0
    b = math.pi / 2

    tol = 1e-12

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


def bisection(f,a,b,tol):

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

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

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

    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]
      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]

question1()
```

2. The code used in this problem is listed below.

   (a) Bisection on $f(x) = (x-5)^9$ finds the root
   $$r \approx 5.00007,$$
   as expected.

   (b) Bisection on the expanded form of $f(x)$ finds the root
   $$r \approx 5.12875,$$
   which is not within the tolerance we specified to find the root.

   (c) The expanded form of $f(x)$ undergoes catastrophic cancellation and resulting loss of precision when evaluated (we showed this on Homework 1). Thus, bisection finds a point where this loss of precision during function evaluation creates an apparent false root.

```
#! /usr/bin/env python3
# APPM 4600 Homework 3 Question 2

import numpy as np
import math

def question2_a():

    f = lambda x: (x-5)**9
    a = 4.82
    b = 5.2

    tol = 1e-4

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

def question2_b():
    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

    tol = 1e-4

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


def bisection(f,a,b,tol):

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

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

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

    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]
      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]

question2_a()
question2_b()
```

3. (a) We wish to find the roots of $f(x) = x**3 + x - 4$ over $[1, 4]$ with an absolute accuracy of 10^-3. From Theorem 2.1, we have that the accuracy of the nth iteration is

$$|p_n - p| \leq \frac{b-a}{2^n},$$

so we need

$$n \leq \log_2\left(\frac{b-a}{|p_n - p|}\right) = \log_2\left(\frac{4-1}{10^-3}\right) \approx 11.5$$

iterations to find the root to the desired accuracy. Thus, we expect no more than 11 iterations.

(b) Bisection (code below) finds the root

$$r \approx 1.3787$$

after 11 iterations, which is consistent with the upper bound found in part (a).

```
#! /usr/bin/env python3
# APPM 4600 Homework 3 Question 3

import numpy as np
import math

def question3_b():

    f = lambda x: x**3 + x - 4
    a = 1
    b = 4

    tol = 1e-3

    [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)


def bisection(f,a,b,tol):

#    Inputs:
#     f,a,b       - function and endpoints of initial interval
#      tol  - bisection stops when relative 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]
```

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

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

    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]

question3_b()
```

4. (a) We have the iterative step $x_{n+1} = -16 + 6x_n + \frac{12}{x_n}$, $x^* = 2$. Observe that $f(x) = -16 + 6x + \frac{12}{x}$ has a fixed point at $f(2) = 2$, but that

$$|f'(x)| = \left|6 - \frac{12}{x^2}\right| > 1$$

in the neighborhood around $x = 2$, so the iteration will not converge to $x^* = 2$.

(b) We have the iterative step $x_{n+1} = \frac{2}{3}x_n + \frac{1}{x_n^2}$, $x^* = 3^{\frac{1}{3}}$. We have that $x^* = 3^{\frac{1}{3}}$ is a fixed point of the iteration and sufficient conditions for convergence ($\left|\frac{\partial x_{n+1}}{\partial x_n}\right| < 1$ around x^*). Observe that

$$\lim_{n\to\infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|} = \lim_{n\to\infty} \frac{|\frac{2}{3}x_n + \frac{1}{x_n^2} - 3^{\frac{1}{3}}|}{|x_n - 3^{\frac{1}{3}}|} = \lim_{n\to\infty} \frac{|2x_n^3 - 3^{\frac{4}{3}}x_n^2 + 3|}{|3x_n^3 - 3^{\frac{4}{3}}x_n^2|} = 0.$$

Thus, the iteration converges linearly with asymptotic error constant 0.

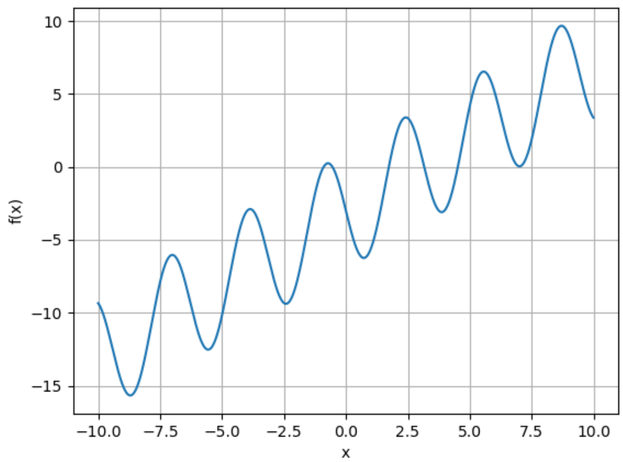
(c) We have iterative step $x_{n+1} = \frac{12}{1+x_n}$, $x^* = 3$. Observe that

$$\lim_{n\to\infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|} = \lim_{n\to\infty} \frac{\left|\frac{12}{1+x_n} - 3\right|}{|x_n - 3|} = \lim_{n\to\infty} \frac{\left|\frac{12-3-3x_n}{1+x_n}\right|}{|x_n - 3|} = \lim_{n\to\infty} \left|3\frac{3 - x_n}{(x_n + 1)(x_n - 3)}\right| = \frac{3}{4}.$$

Thus, the iteration converges linearly with asymptotic error constant $\frac{3}{4}$.

5. The code for this question is listed below.

   (a) A plot of $f(x) = x - 4\sin(2x) - 3$ is shown below. The function has five roots.



   (b) Fixed point iteration of

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

   gives only the roots

$$r_1 \approx -0.544442400,$$
$$r_2 \approx 3.161826486.$$

   These are the only roots that occur where

$$\left|\frac{\partial x_{n+1}}{\partial x_n}\right| = \left|-2\cos(2x_n) + \frac{5}{4}\right| \leq 1,$$

   which is the only region over which we can expect the fixed point iteration to converge.

```
#! /usr/bin/env python3
# APPM 4600 Homework 3 Question 5

import numpy as np
import matplotlib.pyplot as plt
import math

def question5_a():
  x = np.arange(-10, 10, 0.01)
  y = x - 4*np.sin(2*x)-3

  plt.plot(x, y)
  plt.xlabel("x")
  plt.ylabel("f(x)")
  plt.grid()
  plt.savefig("hw3_5_a.png")

def question5_b():
  rtol = 0.5e-10
```

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

    # attempt fixed point iteration starting at the approximate location of all the roots
    for guess in [-0.898, -0.55, 1.732, 3.16, 4.517]:
        [xstar, x_guess, ier] = fixedpt(f, guess, rtol, 100)
        print("xstar=", xstar)
        print("ier=", ier)
        print("number of iterations=", len(x_guess))

# run fixed point
def fixedpt(f,x0,tol,Nmax):

    ''' x0 = initial guess'''
    ''' Nmax = max number of iterations'''
    ''' tol = stopping tolerance'''
    x_guess = np.zeros(0)

    count = 0
    while (count <Nmax):
        x_guess = np.append(x_guess, x0)
        count = count +1
        x1 = f(x0)
        if (abs(x1-x0)/abs(x1) <tol):
            xstar = x1
            ier = 0
            return [xstar,x_guess,ier]
        x0 = x1

    xstar = x1
    ier = 1
    return [xstar, x_guess, ier]

question5_a()
question5_b()
```