Numerical Analysis: Homework #1

Due on January 12, 2015

Professor Mohler MWF 2:15

Rick Sullivan

Problem 1

a) Use three iterations of the bisection method to approximate the root of $f(x) = \sqrt{x} - \cos x$ in [0, 1].

a) How many iterations would be needed to approximate the root with accuracy 10^{-5}?

Part a
Initially, $a = 0$, $b = 1$, and $p = \frac{1}{2}$.

$$\begin{aligned} f(p) * f(a) &= (\sqrt{\frac{1}{2}} - \cos\frac{1}{2}) * (\sqrt{0} - \cos 0) \\ &= -\frac{1}{\sqrt{2}} + \cos\frac{1}{2} \approx 0.1705 \end{aligned}$$

Because this is greater than 0, set $a = p$ and continue. With $a = \frac{1}{2}$, $b = 1$, and $p = \frac{3}{4}$:

$$\begin{aligned} f(p) * f(a) &= (\sqrt{\frac{3}{4}} - \cos\frac{3}{4}) * (\sqrt{\frac{1}{2}} - \cos\frac{1}{2}) \\ &\approx -0.0229 \end{aligned}$$

Because this is less than 0, set $b = p$ and continue. With $a = \frac{1}{2}$, $b = \frac{3}{4}$, and $p = \frac{5}{8}$:

$$\begin{aligned} f(p) * f(a) &= (\sqrt{\frac{5}{8}} - \cos\frac{5}{8}) * (\sqrt{\frac{1}{2}} - \cos\frac{1}{2}) \\ &\approx 0.0034 \end{aligned}$$

Greater than 0, so $a = p$. After three iterations we have:

$$\begin{aligned} p &= \frac{\frac{5}{8} + \frac{3}{4}}{2} \\ &= \frac{11}{16} \approx .6875 \end{aligned}$$

Part b
Accuracy can be determined by $\frac{|b-a|}{2^n}$, where n is the number of iterations.
Solving for n with accuracy 10^{-5}:

$$\begin{aligned} \frac{|1-0|}{2^n} &= 10^{-5} \\ 2^n &= 10^5 \\ n\log 2 &= \log 10^5 \\ n &= \frac{\log 10^5}{\log 2} \\ &\approx 16.6 \end{aligned}$$

In order to meet that accuracy, we must iterate 17 times.

Problem 2

Write a program to approximate the root of $3x - e^x$ for $1 \leq x \leq 2$ within 10^{-5} of the exact solution. Have the program print the estimate at each iteration to the command line. Attach the code to your hw, as well as a print out of the command line.

Solution
I implemented a root approximation using the bisection method we discussed in class. The code can be found on page 6.

```
Iteration 1: p=1.75
Iteration 2: p=1.625
Iteration 3: p=1.5625
Iteration 4: p=1.53125
Iteration 5: p=1.515625
Iteration 6: p=1.5078125
Iteration 7: p=1.51171875
Iteration 8: p=1.513671875
Iteration 9: p=1.5126953125
Iteration 10: p=1.51220703125
Iteration 11: p=1.511962890625
Iteration 12: p=1.5120849609375
Iteration 13: p=1.51214599609375
Iteration 14: p=1.512115478515625
Iteration 15: p=1.5121307373046875
Iteration 16: p=1.5121383666992188
Iteration 17: p=1.5121345520019531
```

Figure 1: Command line output from running bisection.py

Problem 3

a) Use algebraic manipulation to show that $g(x) = \left(\frac{x+3}{x^2+2}\right)^{1/2}$ has a fixed point where the function $f(x) = x^4 + 2x^2 - x - 3$ has a root.

b) Perform three fixed point iterations with $g(x)$ and $p_0 = 1$ to approximate the root. Will the fixed point iteration converge?

Part a

First, I will solve $f(x) = 0$ for x.

$$\begin{aligned} x^4 + 2x^2 - x - 3 &= 0 \\ x^4 + 2x^2 &= x + 3 \\ x^2(x^2 + 2) &= x + 3 \\ x^2 &= \frac{x+3}{x^2+2} \\ x &= \left(\frac{x+3}{x^2+2}\right)^{1/2} = g(x) \end{aligned}$$

Therefore, $g(x)$ has a fixed point where $f(x)$ has a root.

Part b

Iteration one with $p_0 = 1$:

$$\begin{aligned} g(1) &= \left(\frac{1+3}{1^2+2}\right)^{1/2} \\ &= \left(\frac{4}{3}\right)^{1/2} \\ &= \left(\frac{2}{\sqrt{3}}\right) \\ &\approx 1.1547 \end{aligned}$$

Iteration two with $p_1 = 1.1547$:

$$\begin{aligned} g(p_1) &= \left(\frac{1.1547+3}{1.1547^2+2}\right)^{1/2} \\ &\approx 1.1164 \end{aligned}$$

Iteration three with $p_2 = 1.1164$:

$$\begin{aligned} g(p_2) &= \left(\frac{1.1164+3}{1.1164^2+2}\right)^{1/2} \\ &\approx 1.1261 \end{aligned}$$

This fixed point iteration will converge. It is approaching 1.1241230297, one of the roots of $f(x)$. We can prove that it will converge by calculating the derivative of $g(x)$ at the root.

$$g'(x) = \frac{-x^2 - 6x + 2}{2\sqrt{\frac{x+3}{x^2+2}}(x^2+2)^2}$$

$$|g'(1.124)| \approx 0.2509$$

This is less than 1, so the approximation will converge.

Problem 4

Rank the following fixed point iteration methods, with $p_0 = 1$, based upon their apparent speed of convergence for computing $21^{\frac{1}{3}}$.

a) $g(x) = (20x + 21/x^2)/21$

b) $g(x) = x - (x^3 - 21)/(3x^2)$

c) $g(x) = x - (x^4 - 21x)/(x^2 - 21)$

d) $g(x) = (21/x)^{1/2}$

Solution
In decreasing speed of convergence, the fixed point iteration methods are ranked b, d, a, and c.

I created a script to run each of the fixed-point iteration functions to experimentally determine speeds of convergence. The script can be found on page 7, and the outputs on page 8.

We can also analyze derivatives of each of the functions: the closer a derivative at the root is to 0, the faster the function will converge to $21^{1/3}$. For reference, $21^{1/3}$ is approximately 2.75892417.

$$\begin{aligned} a'(x) &= 20/21 + -2/x^3 \\ |a'(2.7589)| &\approx 0.80671 \end{aligned}$$

$$\begin{aligned} b'(x) &= 1 - (9x^4 - (x^3 - 21)6x)/(9x^4) \\ &= 1 - 1/3 - 126/9x^3 \\ &= 2/3 - 14/x^3 \\ |b'(2.7589)| &\approx 0.00001753 \end{aligned}$$

$$\begin{aligned} c'(x) &= 1 - ((4x^3 - 21)(x^2 - 21) - (x^4 - 21x)(2x))/(x^2 - 21)^2 \\ |c'(2.7589)| &\approx 5.7053 \end{aligned}$$

$$\begin{aligned} d'(x) &= -1/2(\sqrt{21}/x^{3/2}) \\ |d'(2.7589)| &\approx 0.5000 \end{aligned}$$

As expected, $b'(x)$ is closest to 0 at 2.7589, followed by d and a. $c'(2.7589)$ is actually greater than 1, so it will not converge.

Appendix

bisection.py

```
# Rick Sullivan
# 6 January 2015
#
# Requires python 3
import math

# Function used in question 1) of homework
# Used this to verify my answer
def f(x):
    return math.sqrt(x) - math.cos(x)

# Function used in question 2) of homework
def g(x):
    return 3*x - math.pow(math.e, x)

# Uses the bisection method to approximate the root of
# the function f over the interval a <= x <= b
# Precision can be set in two ways:
#   1) Set the number of iterations the algorithm will take
#   2) Give a number of decimal places, which will calculate
#       how many iterations are needed
def bisection(a, b, f, iterations=None, decimal_places=None):
    if iterations is None and decimal_places is None:
        raise Exception("Must provide either number of iterations or decimal
            places")
    if decimal_places is not None:
        iterations = math.ceil(math.log10(abs(b-a) * math.pow(10,
            decimal_places)) / math.log10(2))
    p = (b+a)/2
    for i in range(iterations):
        if f(p)*f(a) < 0:
            b = p
        else:
            a = p

        p = (b+a)/2
        print("Iteration " + str(i+1) + ": p=" + str(p))

    return p

# Verify question 1
# bisection(0, 1, f, iterations=3)
# Generate accurate answer for question 2
bisection(1, 2, g, decimal_places=5)
```

fixedpoint.py

```
# Rick Sullivan
# 6 January 2015
#
# Requires python 3
import math

def a(x):
    return (20*x + 21/math.pow(x, 2))/21

def b(x):
    return x - (math.pow(x, 3) - 21)/(3*math.pow(x, 2))

def c(x):
    return x - (math.pow(x, 4) - 21*x)/(math.pow(x, 2) - 21)

def d(x):
    return math.pow(21/x, 0.5)

# Provides a fixed point iteration on the function f starting at the point p
def fixedpoint(p, f, iterations=10):
    for i in range(iterations):
        p = f(p)
        print("Iteration " + str(i+1) + ": p=" + str(p))

    return p

# Function used in question 3
def g(x):
    return math.pow((x + 3)/(math.pow(x, 2) + 2), 0.5)

functions = [a, b, c, d]
for f in functions:
    print("Function " + f.__name__ + ":")
    fixedpoint(1, f)
    print("")
```

Fixed point output

```
Function a:
Iteration 1: p=1.95238095238
Iteration 2: p=2.12175427379
Iteration 3: p=2.24284969202
Iteration 4: p=2.33483967253
Iteration 5: p=2.4070933802
Iteration 6: p=2.46505928751
Iteration 7: p=2.51224346295
Iteration 8: p=2.55105709638
Iteration 9: p=2.5832377672
Iteration 10: p=2.61008144462

Function b:
Iteration 1: p=7.66666666667
Iteration 2: p=5.23020373871
Iteration 3: p=3.7426969187
Iteration 4: p=2.99485356828
Iteration 5: p=2.77702222587
Iteration 6: p=2.75904186642
Iteration 7: p=2.7589241814
Iteration 8: p=2.75892417638
Iteration 9: p=2.75892417638
Iteration 10: p=2.75892417638

Function c:
Iteration 1: p=0.0
Iteration 2: p=0.0
Iteration 3: p=0.0
Iteration 4: p=0.0
Iteration 5: p=0.0
Iteration 6: p=0.0
Iteration 7: p=0.0
Iteration 8: p=0.0
Iteration 9: p=0.0
Iteration 10: p=0.0

Function d:
Iteration 1: p=4.58257569496
Iteration 2: p=2.14069514293
Iteration 3: p=3.13207559492
Iteration 4: p=2.58936652742
Iteration 5: p=2.84782227445
Iteration 6: p=2.71552125263
Iteration 7: p=2.78088509471
Iteration 8: p=2.74800883838
Iteration 9: p=2.7643980934
Iteration 10: p=2.7561912839
```