M 125 HW 2

1 a) $K(x) = \frac{|x f'(x)|}{|f(x)|} = \frac{|x \cdot \frac{1}{x}|}{|\ln x|} = \boxed{\frac{1}{|\ln x|} \text{ so } f(x) \text{ is illconditioned for } x \approx 1.}$

b) $K(x) = \frac{|x f'(x)|}{|f(x)|} = \frac{\left|\frac{-x}{(x-1)^2}\right|}{\left|\frac{x}{x-1}\right|} = \left|\frac{-1}{x-1}\right| = \frac{1}{|x-1|}$

$\boxed{K(x) = \frac{1}{|x-1|} \text{ so } f(x) \text{ is ill-conditioned for } x \text{ close to } 1.}$

c) $a_i \xrightarrow{g} x_J \qquad g(a_i) = x_J$

$$\frac{\left|\frac{g(a_i+\Delta) - g(a_i)}{a_i}\right|}{\left|\frac{\Delta}{a_i}\right|} = \left|\frac{g(a_i+\Delta) - g(a_i)}{\Delta} \cdot \frac{a_i}{g(a_i)}\right|$$

$$= \left|g'(a_i) \cdot \frac{a_i}{x_J}\right|$$

$$= \left|\frac{a_i g'(a_i)}{x_J}\right|$$

$\frac{da_i}{dx_J}$: $a_0 + a_1 x_J + a_2 x_J^2 + \dots + a_n x_J^n = 0$

$a_1 + 2a_2 x_J + \dots + i a_i x_J^{i-1} + \frac{\partial a_i}{\partial x_J} x_J^i + \dots + a_n x_J^n$

$$\frac{\partial a_i}{\partial x_J} = \frac{-a_1 - 2a_2 - \dots - n a_n x_J^{n-1}}{x_J^i} = \frac{-p'(x_J)}{x_J^i}$$

Want reciprocal so $\frac{x_J^i}{-p'(x_J)}$

$$K(a_i) = \left|\frac{x_J^i}{-p'(x_J)} \cdot \frac{a_i}{x_J}\right| = \left|\frac{x_J^{i-1} a_i}{p'(x_J)}\right|$$

So condition number is $\left|\frac{x_J^{i-1} a_i}{p'(x_J)}\right|$ and so $P(x)$ is ill conditioned for large a_i or large x_J^{i-1}, meaning as i increases $p(x)$ is more ill conditioned.

Question 1D)

Using the Matlab code provided, along with some additional calculations, here are the roots of the Wilkinson Polynomial, along with the relative error for each root:

Root	True root	Relative Error
20.8469 + 0.0000i	20	0.7157
19.5024 + 1.9403i	19	1.8016
19.5024 - 1.9403i	18	2.2058
16.7307 + 2.8126i	17	2.9338
16.7307 - 2.8126i	16	3.0174
13.9924 + 2.5188i	15	3.3614
13.9924 - 2.5188i	14	3.1210
11.7939 + 1.6525i	13	3.0262
11.7939 - 1.6525i	12	2.4633
10.0955 + 0.6449i	11	1.9346
10.0955 - 0.6449i	10	1.1353
8.9158 + 0.0000i	9	0.1663
8.0078 + 0.0000i	8	0.0171
6.9996 + 0.0000i	7	0.0010
6.0000 + 0.0000i	6	0.0001
5.0000 + 0.0000i	5	0.0000
4.0000 + 0.0000i	4	0.0000
3.0000 + 0.0000i	3	0.0000
2.0000 + 0.0000i	2	0.0000
1.0000 + 0.0000i	1	0.0000

1d) Check back for write up of answers.

relative error is $\dfrac{\dfrac{|x-r|}{|r|}}{\left|\dfrac{2^{-23}}{210}\right|}$

2 $\tilde{f}(x) = \dfrac{|f(\tilde{x})-f(x)|}{|f(x)|} \cdot K(x)$

rel. error of $\tilde{x}$ is $\dfrac{|\tilde{x}-x|}{|x|} < \mathcal{E}_{machine}$

$$\frac{\dfrac{|f(\tilde{x})-f(x)|}{|f(x)|}}{\dfrac{|\tilde{x}-x|}{|x|}} \leftarrow K(x) \qquad \cdot \frac{|\tilde{x}-x|}{|x|} < \mathcal{E}$$

$\dfrac{|\tilde{x}-x|}{|x|} < \mathcal{E}_m \qquad K(x) \cdot \mathcal{E}_m < \mathcal{E}$

So this is bounded above by $K(x)\mathcal{E}$

and $\dfrac{|\tilde{f}(x)-f(x)|}{|f(x)|} = O(K(x)\mathcal{E})$

3 Since $f'(x) > 0 \;\forall x$, and some r s.t. $f(r)=0$ exists, then by definition, r is unique as f is increasing.

For convergence
$e_{i+1} = e_i^2 \dfrac{f''(c_i)}{2f'(x_i)}$ Since $f''(x) > 0 \;\forall x$ and $f'(x) > 0 \;\forall x$

$e_i^2 \dfrac{f''(c_i)}{2f'(x_i)} \geq 0$ so $e_{i+1} = x_{i+1} - r > 0$

$x_{i+1} > r$

So $\forall n > 0$, $x_n > r$ and $f(x_n) > f(r) = 0$

$x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$ $f(x_n) \geq 0 \;\forall x \geq 1$, so $\dfrac{f(x_n)}{f'(x_n)} > 1$

So x_{n+1} is decreasing each iteration, but bounded below by $f(r)=0$

So by monotonic convergence theorem, $x_{n+1} \to r$.

4a) Let $g(a) = \frac{1}{a} - x$ $g'(a) = \frac{-1}{a^2}$

$$a_{n+1} = a_n - \frac{g(a)}{g'(a)}, \quad a_{n+1} = a_n - \left(\frac{\frac{1}{a_n} - x}{-\frac{1}{a_n^2}}\right)$$

$$a_{n+1} = a_n + a_n^2\left(\frac{1}{a_n} - x\right)$$

$$a_{n+1} = a_n + a_n - a_n^2 x$$

$$a_{n+1} = 2a_n - a_n^2 x = \boxed{a_n(2 - a_n x) = a_{n+1}}$$

b) $\epsilon_{k+1} = -\epsilon_k^2$ $\quad \epsilon_k = a x_k - 1$ where a is value converge to

$$(a x_{k+1} - 1) = -(a x_{k} - 1)^2$$

$$a x_{k+1} - 1 = -(a^2 x_{k}^2 - 2a x_k + 1)$$

$$\cancel{a} x_{k+1} \cancel{-1} = -\cancel{a}^2 x_{k}^2 + 2\cancel{a} x_k \cancel{-1}$$

$$x_{k+1} = -a x_{k}^2 + 2 x_k$$

$$x_{k+1} = x_k(2 - a x_k)$$ which is equivalent to the Newton's method scheme derived above

c) $\epsilon_1 = -\epsilon_0^2$

$\epsilon_2 = -\epsilon_0^4$ $\quad$ Want $|\epsilon_k| \leq 2^{-d}$

$\epsilon_k = -\epsilon_0^{2k}$ $\quad$ $|-\epsilon_0^{2k}| < 2^{-d}$

$$\log_2 \epsilon_0^{2k} < 2^{-d}$$

$$2k \log_2 \epsilon_0 < -d$$

$$k < \frac{-d}{2k \log_2 \epsilon_0}$$

It takes $\frac{-d}{2 \log_2 \epsilon_0}$ iterations to get within d binary decimals

Note this is positive as $|\epsilon_0| < 1$ so $\log_2 \epsilon_0 < 0$

4d) No, $f''(x) = -\frac{1}{x^2}$, $f''(x) < 0$ for $x < 0$, so as shown in q3, we cannot guarantee convergence.

Question 5 Part 1 Newton's Method:

#Code

```
#Note: To run for each function, replace f with g or h and df with
#dg/dh
import numpy as np

#Function and derivative for part 1
f = lambda x : np.log(x)+x**2 - 3
df = lambda x: 1/x +2*x

#Function and derivative for part 2
g = lambda x: x**5+x-1
dg = lambda x: 5*x**4+1

#Function and derivative for part 3
h = lambda x: np.sin(x)-6*x-5
dh = lambda x: np.cos(x) - 6

#Note that
def newton(x0, epsilon): #Newton's method
    x1 = x0
    x = x1 + 50*epsilon #Just guarantees runs first iteration
    iter = 0
    while (abs(x1-x) > epsilon): #If > error bound, hasn't converged
        x = x1
        x1 = x1 - f(x1)/df(x1) #Updates new point with function f
        print("       ", iter, "|", x1, "|", f(x1)) #Display
        iter += 1
    return(x1)

x = 3 #Starting point
epsilon = .000001 #Error to end algorithm
print("Iterations|", "x |", "f(x)")
print("Final value:", newton(x, epsilon))
```

#END CODE

```
Part 1) Print out result for x^5+x =1
Iteration | x | h(x)
        0 | 2.396551724137931 | 3.6174910890527627
        1 | 1.9117137201111394 | 1.3026494231046835
        2 | 1.5215611067915087 | -0.2651149465482239
        3 | 1.2094340525058087 | -1.3471167476500898
        4 | 0.9703214785442607 | -2.088604069488606
        5 | 0.8174444097911604 | -2.5333570156794933
```

```
        6 | 0.7610052328597109 | -2.6939860804102604
        7 | 0.7549389861646737 | -2.711185473194238
        8 | 0.7548776724115093 | -2.7113592658509007
        9 | 0.7548776662466928 | -2.7113592833249083
Final value: 0.7548776662466928
```
Rounded to 8 decimal places, the root is at x = 0.75487766

```
Part 2) Print out result for sin(x)=6x+5
Iteration | x | h(x)
        0 | -0.27022954646338393 | -3.645575381080108
        1 | -0.994090786221374 | 0.1262811673085089
        2 | -0.9709400422003657 | 0.00022349899407547724
        3 | -0.9708989236326309 | 6.977751709769109e-10
        4 | -0.9708989235042558 | 0.0
Final value: -0.9708989235042558
```
Rounded to 8 decimal places, the root is at x = -0.97089892

```
Part 3) Print out result for ln(x)+x^2=3
Iteration | x | f(x)
        0 | 1.8791664807366144 | 1.1620949795418927
        1 | 1.6083124147681183 | 0.06185426317372
        2 | 1.592197796540965 | 0.0002091470728369238
        3 | 1.5921429376917977 | 2.4159136913226575e-09
        4 | 1.592142937058094 | 4.440892098500626e-16
Final value: 1.592142937058094
```
Rounded to 8 decimal places, the root is at x = 1.59214293

Question 5 Part 2 Secant Method:

#**CODE**

```
f = lambda x: x**3+x-1 #Function
def secant(x0, x1, epsilon):
    dummy = 0
    for i in range(1000):
        print(dummy,"| " ,x1)
        dummy = x1
        if abs(f(x1)-f(x0)) < epsilon: #If in epsilon bound, end
            break
        x1 = x1 - f(x1)*(x1-x0)/(f(x1)-f(x0)) #Updates new point
        x0 = dummy
    return x1

x0 = 0 #Starting value
x1 = 1
epsilon = 0.0000000000001 #Error bound
```

```
print("iteration| x0 | x1")
print("Final value:", secant(x0, x1, epsilon))
```

#END CODE

```
Print out result for f(x)= x^3+x-1:
iteration| x0 | x1
0 |  1
1 |  0.5
0.5 |  0.6363636363636364
0.6363636363636364 |  0.6900523560209424
0.6900523560209424 |  0.6820204196481856
0.6820204196481856 |  0.6823257814098928
0.6823257814098928 |  0.6823278043590257
0.6823278043590257 |  0.6823278038280184
0.6823278038280184 |  0.6823278038280193
Final value: 0.6823278038280193
```

Rounded to 8 decimal places, the root is at x = 0.68232780

Question 5 Part 3 Bisection Method:

#CODE

```
import numpy as np
import math

f = lambda x: (math.e)**x - np.sin(x) #Function

def bisection(a, b, epsilon):
    c1 = 0
    iter = 0
    while f(a)*f(b) < 0:
        c = (a+b)/2 #New point
        if (abs(c-c1) < epsilon): #If below error bound we are done
            break
        c1 = c #Temp var to see how close we are to convergence
        if f(c)*f(a) < 0: #Update value
            b = c
        elif f(c)*f(b) < 0: #Update value
            a = c
        elif f(c) == 0: #If we're at the root
            break
        print(iter, "|", c)
        iter += 1
    return c

a = -4 #Starting point
```

```
b = -1
epsilon = 0.000001 #Error bound

print("iteration | midpoint")
print(bisection(a, b, epsilon))
```
#END CODE

Print out of result for root of e^x = sin(x) closest to 0:

```
iteration | midpoint
0 | -2.5
1 | -3.25
2 | -2.875
3 | -3.0625
4 | -3.15625
5 | -3.203125
6 | -3.1796875
7 | -3.19140625
8 | -3.185546875
9 | -3.1826171875
10 | -3.18408203125
11 | -3.183349609375
12 | -3.1829833984375
13 | -3.18316650390625
14 | -3.183074951171875
15 | -3.1830291748046875
16 | -3.1830520629882812
17 | -3.183063507080078
18 | -3.1830577850341797
19 | -3.183060646057129
20 | -3.1830620765686035
```
Rounded to 8 decimal places, the root is -3.18306207

5) Check back for results and code for all parts.

6 a) $f'(x)=\frac{2}{3}(2x-1)^{-2/3}$, $f'(1)=\frac{2}{3}<1$, so $f(x)$ is locally convergent at $r=1$

b) $f'(x)=\frac{3x^2}{2}$, $f'(1)=3/2>1$, so $f(x)$ is not locally convergent at $x=1$.

c) $f'(x)=\cos x+1$ $f'(0)=2>1$, so $f(x)$ isn't locally convergent to the fixed point at $x=0$.

7 a) $x=\frac{4}{5}x+\frac{1}{x}$, $\rightarrow$ $\frac{1}{5}x=\frac{1}{x}$, $\frac{x^2}{5}=1$, $x=\sqrt{5}$ so this converges to $\sqrt{5}$

$g(x)=\frac{4}{5}x+\frac{1}{x}$, $g'(x)=\frac{4}{5}-\frac{1}{x^2}$, $g'(\sqrt{5})=\frac{3}{5}$

$\boxed{g(x) \text{ converges to } \sqrt{5} \text{ and } |g'(\sqrt{5})|=\frac{3}{5}<1}$

b) $x=\frac{x}{2}+\frac{5}{2x}$, $\frac{x}{2}=\frac{5}{2x}$, $x^2=5$, $x=\sqrt{5}$. So converges to $\sqrt{5}$.

$g(x)=\frac{x}{2}+\frac{5}{2x}$, $g'(x)=\frac{1}{2}-\frac{5}{2x^2}$, $g'(\sqrt{5})=0<1$

$\boxed{g(x) \text{ converges to } \sqrt{5} \text{ and } |g'(\sqrt{5})|=0<1.}$

c) $x=\frac{x+5}{x+1}$, $x^2+x=x+5$, $x^2=5$, $x=\sqrt{5}$, so converges to $\sqrt{5}$

$g'(x)=\frac{x+1-(x+5)}{(x+1)^2}=\frac{-4}{(x+1)^2}$ $g'(\sqrt{5})=\left|\frac{-4}{(\sqrt{5}+1)^2}\right|<1$

So, $g(x)$ converges to $\sqrt{5}$ and $|g'(\sqrt{5})|=.38<1$.

Scanned with CamScanner