# 250HW6 Q4

November 13, 2018

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from numpy.linalg import *
        import math
```

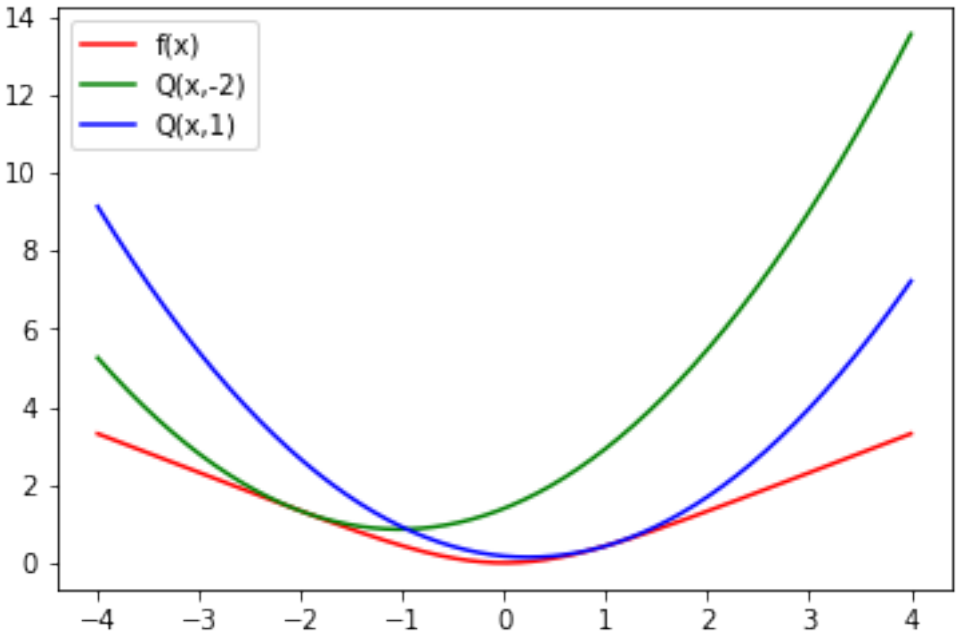
### 0.1 part (c)

```
In [4]: def f(x):
            return np.log(np.cosh(x))

        def f_p(x):
            return np.sinh(x)/np.cosh(x)

        def Q(x,y):
            return f(y)+f_p(y)*(x-y)+(x-y)*(x-y)/2

        x = np.linspace(-4, 4, 1000)
        fig = plt.figure()
        plt.plot(x, f(x),'r', label='f(x)')
        plt.plot(x, Q(x,-2),'g', label = 'Q(x,-2)')
        plt.plot(x, Q(x,1),'b', label = 'Q(x,1)')
        plt.legend(loc='upper left')
        plt.show()
```

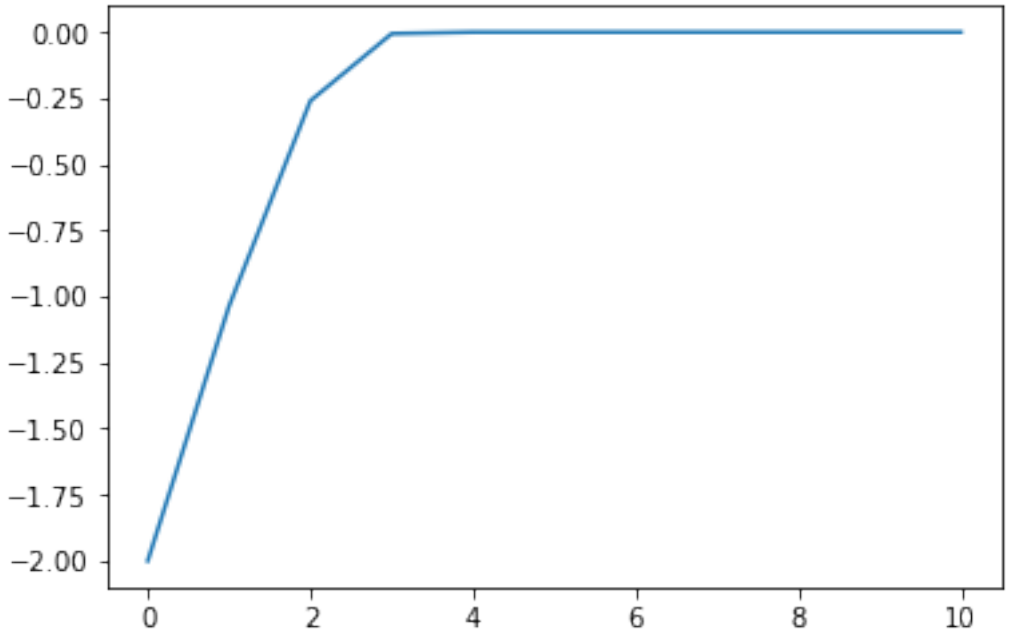



### 0.2 part (f)

```
In [5]: #update rule
        def CalcXn(x):
            x -= np.sinh(x)/np.cosh(x)
            return x
```

```
In [19]: # plot xn vs n, x0 = -2
         x = -2
         step = 10
         result = np.ones((step+1,1))
         n = np.linspace(0, step, step+1)
         result[0,0] = x
         for i in range(step):
             x = CalcXn(x)
             result[i+1,0] = x
         plt.plot(n,result)
```

```
Out[19]: [<matplotlib.lines.Line2D at 0x110a07a90>]
```

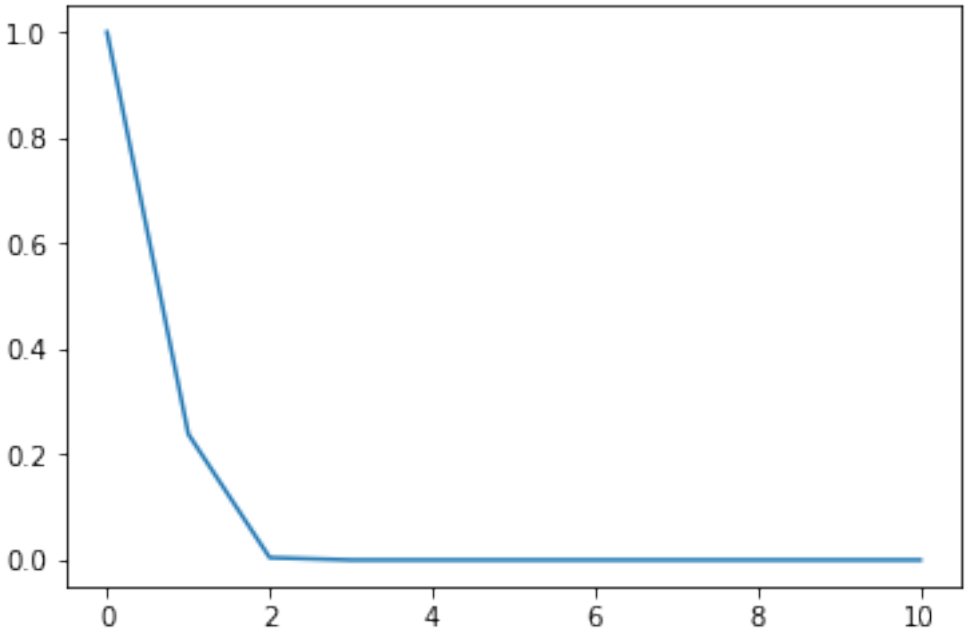



```
In [20]: # x0 = -2
         result
```

```
Out[20]: array([[-2.00000000e+00],
                [-1.03597242e+00],
                [-2.59679795e-01],
                [-5.68378530e-03],
                [-6.12048908e-08],
                [-6.61744490e-23],
                [ 0.00000000e+00],
                [ 0.00000000e+00],
                [ 0.00000000e+00],
                [ 0.00000000e+00],
                [ 0.00000000e+00]])
```

```
In [21]: # plot xn vs n, x0 = 1
         x = 1
         step = 10
         result = np.ones((step+1,1))
         n = np.linspace(0, step, step+1)
         result[0,0] = x
         for i in range(step):
             x = CalcXn(x)
             result[i+1,0] = x
         plt.plot(n,result)
```

```
Out[21]: [<matplotlib.lines.Line2D at 0x1109b6978>]
```




```
In [22]: # x0 = 1
         result

Out[22]: array([[1.00000000e+00],
                [2.38405844e-01],
                [4.41640558e-03],
                [2.87132405e-08],
                [9.92616735e-24],
                [0.00000000e+00],
                [0.00000000e+00],
                [0.00000000e+00],
                [0.00000000e+00],
                [0.00000000e+00],
                [0.00000000e+00]])
```

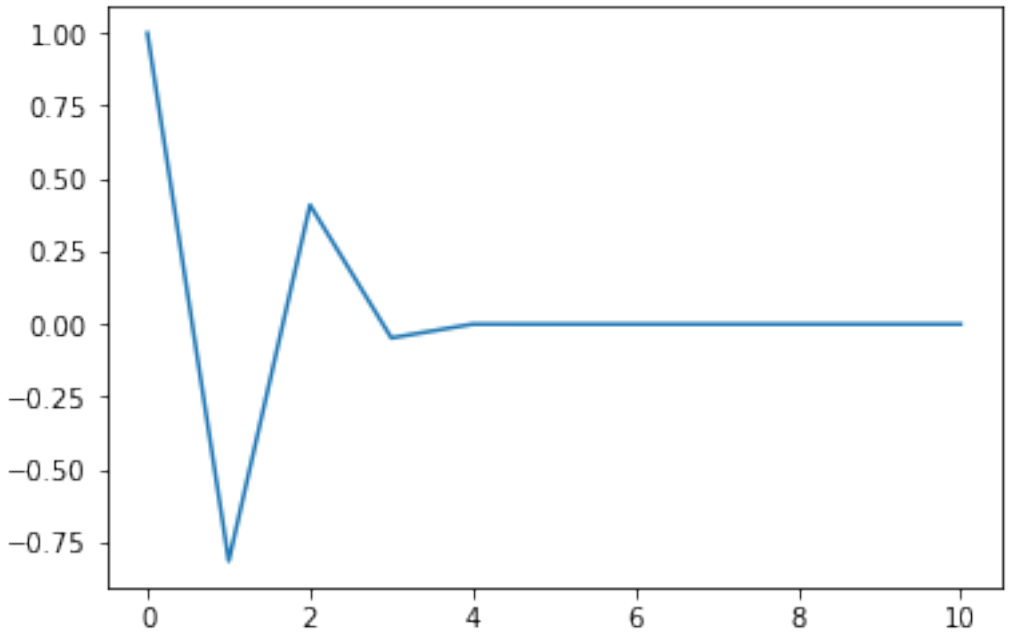
### 0.3 part(g)

```
In [23]: #update rule
         def CalcXnNewton(x):
             x -= np.sinh(x)*np.cosh(x)
             return x

In [25]: # plot xn vs n, x0 = 1
         x = 1
```

```
step = 10
result = np.ones((step+1,1))
n = np.linspace(0, step, step+1)
result[0,0] = x
for i in range(step):
    x = CalcXnNewton(x)
    result[i+1,0] = x
plt.plot(n,result)
```

Out[25]: [<matplotlib.lines.Line2D at 0x1173318d0>]



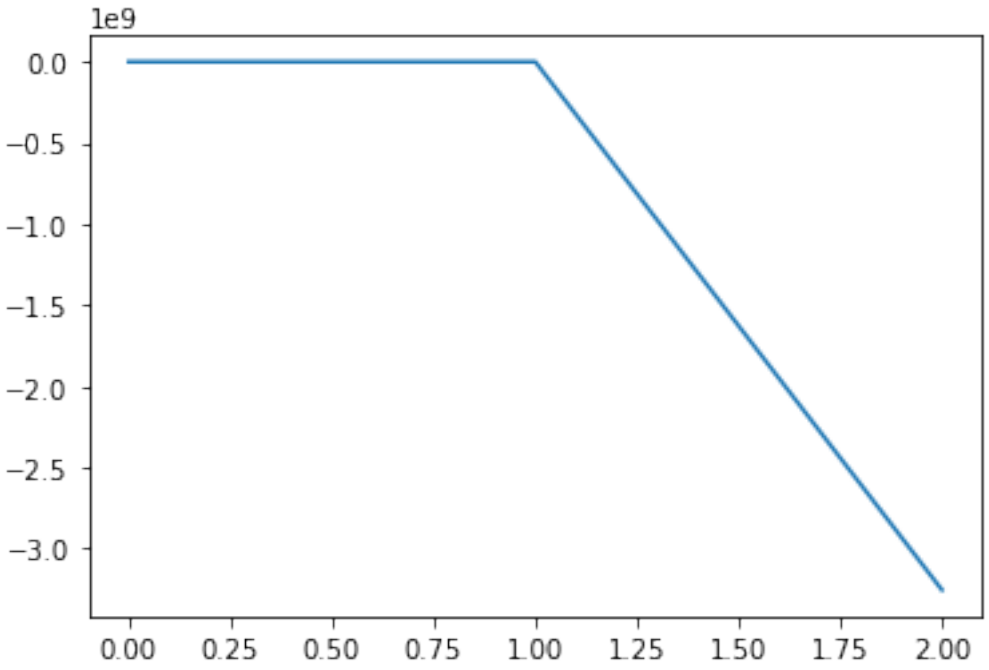

In [26]:
```
# x0 = 1
result
```

Out[26]:
```
array([[ 1.00000000e+00],
       [-8.13430204e-01],
       [ 4.09402317e-01],
       [-4.73049165e-02],
       [ 7.06028036e-05],
       [-2.34625145e-13],
       [ 0.00000000e+00],
       [ 0.00000000e+00],
       [ 0.00000000e+00],
       [ 0.00000000e+00],
       [ 0.00000000e+00]])
```

In [27]:
```
# plot xn vs n, x0 = -2
x = -2
step = 10
result = np.ones((step+1,1))
n = np.linspace(0, step, step+1)
result[0,0] = x
for i in range(step):
    x = CalcXnNewton(x)
    result[i+1,0] = x
plt.plot(n,result)
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: RuntimeWarning: overflow encou
  This is separate from the ipykernel package so we can avoid doing imports until
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: RuntimeWarning: overflow encou
  This is separate from the ipykernel package so we can avoid doing imports until
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: RuntimeWarning: invalid value
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[27]: [<matplotlib.lines.Line2D at 0x1173954a8>]




In [28]:
```
# x0 = -2
result
```

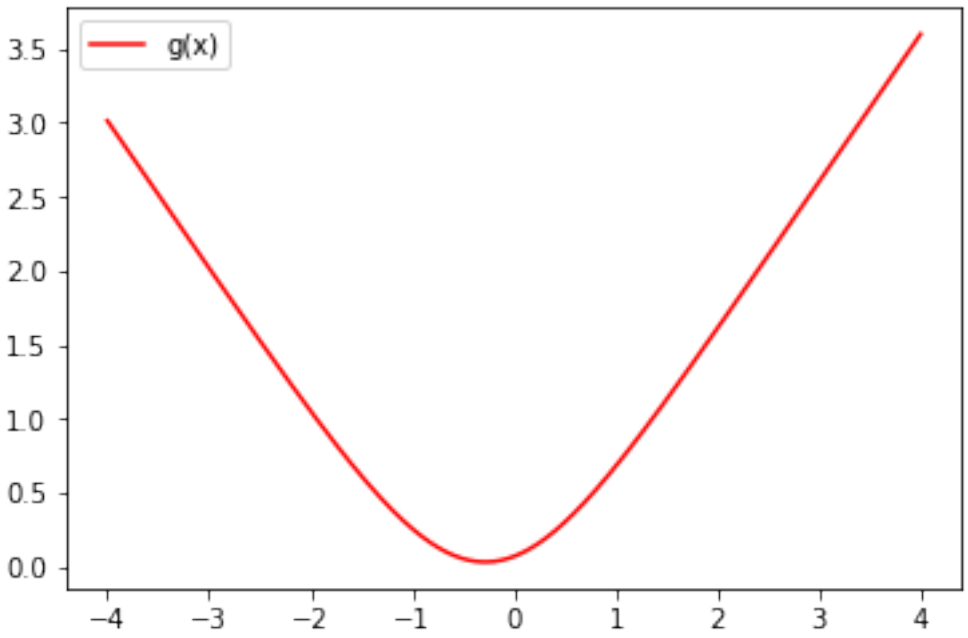
Out[28]:
```
array([[-2.00000000e+00],
       [ 1.16449586e+01],
```

```
       [-3.25553621e+09],
       [            inf],
       [            nan],
       [            nan],
       [            nan],
       [            nan],
       [            nan],
       [            nan],
       [            nan]])
```

### 0.4 part(h)

```
In [29]: def g(x):
             result = 0
             for k in range(10):
                 result += f(x+1/(k+1))
             result /= 10
             return result
```

```
In [32]: x = np.linspace(-4, 4, 1000)
         fig = plt.figure()
         plt.plot(x, g(x),'r', label='g(x)')
         plt.legend(loc='upper left')
         plt.show()
```

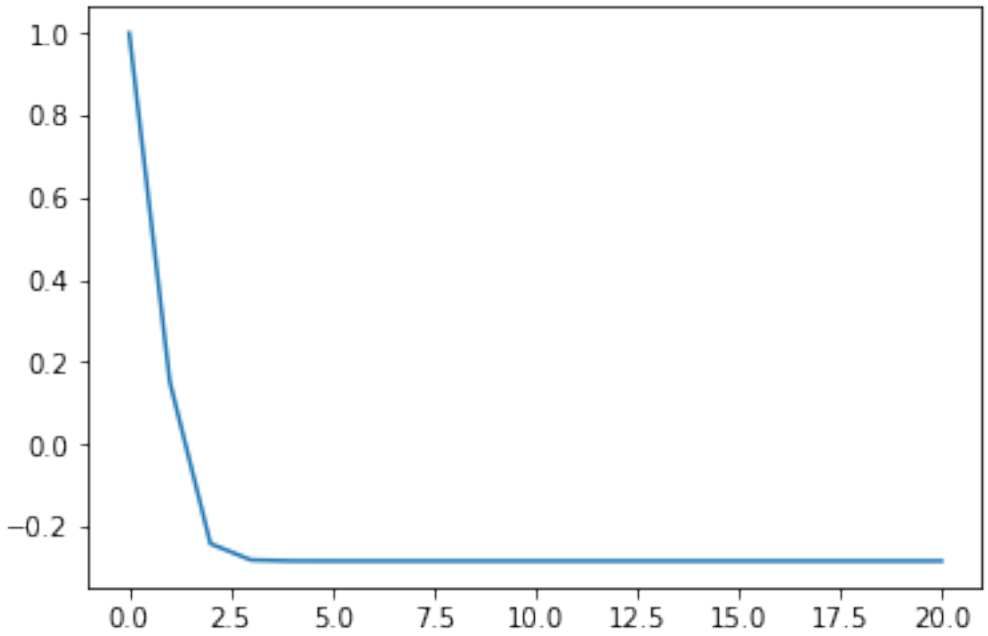


### 0.5 part(k)

```
In [33]: def CalcXnG(x):
             result = 0
             for k in range(10):
                 result += f_p(x+1/(k+1))
             result /= 10
             return x - result
```

```
In [36]: # plot of xn vs n, x0 = 1
         x = 1
         step = 20
         result = np.ones((step+1,1))
         n = np.linspace(0, step, step+1)
         result[0,0] = x
         for i in range(step):
             x = CalcXnG(x)
             result[i+1,0] = x
         plt.plot(n,result)
```

```
Out[36]: [<matplotlib.lines.Line2D at 0x117599668>]
```




```
In [38]: # minimum to 4 significant digits, x0 = 1
         result
```

```
Out[38]: array([[ 1.        ],
                [ 0.15259948],
```

```
[-0.24096661],
[-0.28071793],
[-0.28289933],
[-0.28301996],
[-0.28302663],
[-0.283027  ],
[-0.28302702],
[-0.28302702],
[-0.28302702],
[-0.28302702],
[-0.28302702],
[-0.28302702],
[-0.28302702],
[-0.28302702],
[-0.28302702],
[-0.28302702],
[-0.28302702],
[-0.28302702],
[-0.28302702]])
```