Problem3

(d)

iteration	Number of mistakes M	Log-likelihood L
0	175	-0.95809
1	56	-0.49592
2	43	-0.40822
4	42	-0.36461
8	44	-0.34750
16	40	-0.33462
32	37	-0.32258
64	37	-0.31483
128	36	-0.31116
256	36	-0.31016

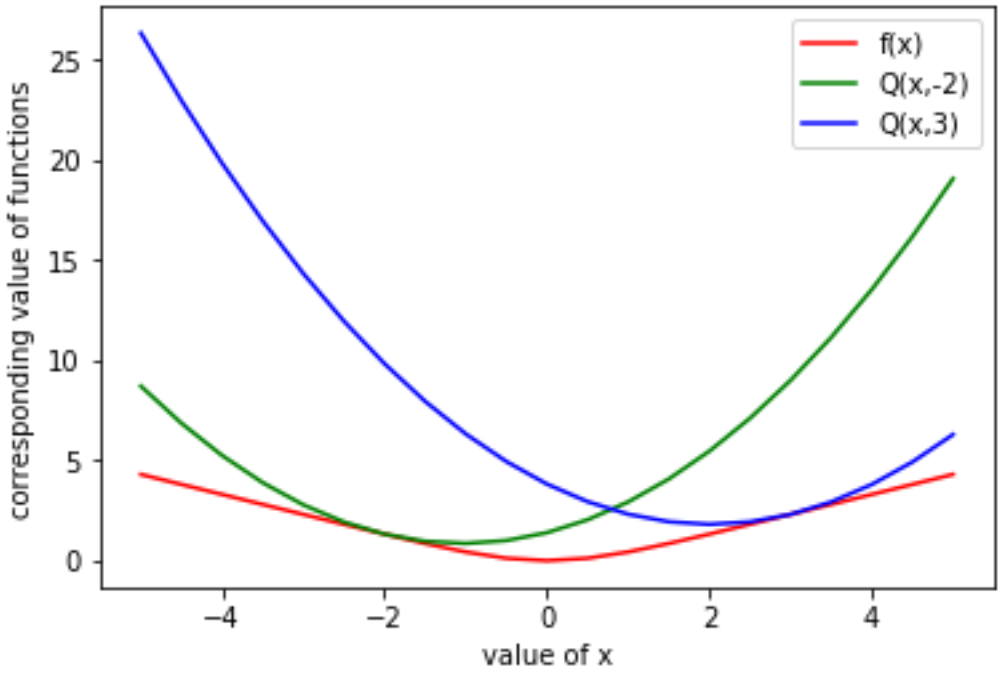
Sourcecode:

```
from math import log
from math import pow
#import the data
X_value=[]
with open('hw6_spectX.txt') as inputfile:
    for line in inputfile:
        X_value.append(line.strip().split(' '))
X_value = [list(map(int,x)) for x in X_value]
Y_value=[]
with open('hw6_spectY.txt') as inputfile:
    Y_value = inputfile.readlines()
Y_value = [int(x.strip()) for x in Y_value]

def posterior(X): #calculate P(Y(t)=0 | X(t))
    result = 1
    for i in range(len(X)):
        result *= pow(1-parameter[i],X[i])
    return result
def likelihood():
    L=0
    for i in range(len(X_value)):
        if Y_value[i]==1:
            L += log(1-posterior(X_value[i]))
        else:
            L += log(posterior(X_value[i]))
    L = L/len(X_value)
    print('likelihood = ',L)
```

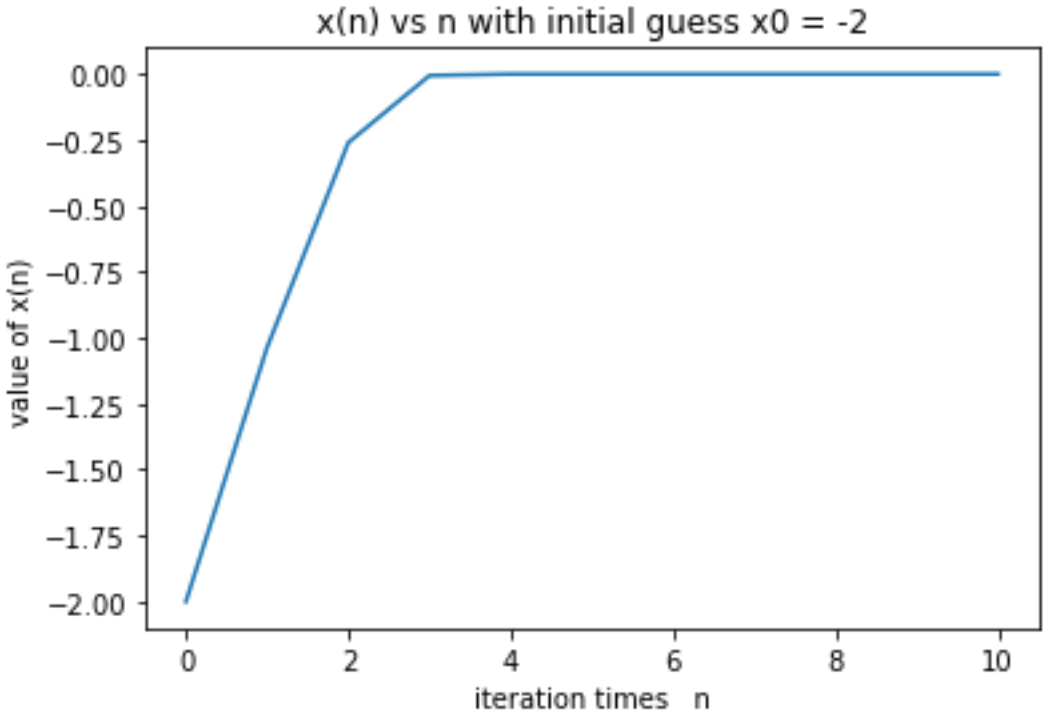
```
    return L

    def mistake():
        wrong_num = 0
        for i in range(len(X_value)):
            prob = 1-posterior(X_value[i])
            if Y_value[i]==0 and prob >= 0.5:
                wrong_num +=1
            if Y_value[i]==1 and prob <=0.5:
                wrong_num +=1
        print('number of mistakes M = ',wrong_num)
        return wrong_num

def posterior_update(index,parameter): #calculate sigma P(zi=1,xi=1| x=x(t),y=y(t))
    result = 0
    T=0
    for i in range(len(X_value)):
        if X_value[i][index]==1:
            T = T+1
        if Y_value[i]==1:
            result += X_value[i][index]*parameter[index]/(1-posterior(X_value[i]))

    p_index = result/T
    return p_index

def update(parameter):
    parameter_new=[0.0]*23
    for i in range(len(parameter)):
        parameter_new[i]=posterior_update(i,parameter)
    return parameter_new


parameter = [0.05]*23
likelihood()
mistake()
iteration = 256 #time of iteration
for i in range(iteration):
    parameter = update(parameter)
    print('iteration:',i+1)
    mistake()
    likelihood()
```

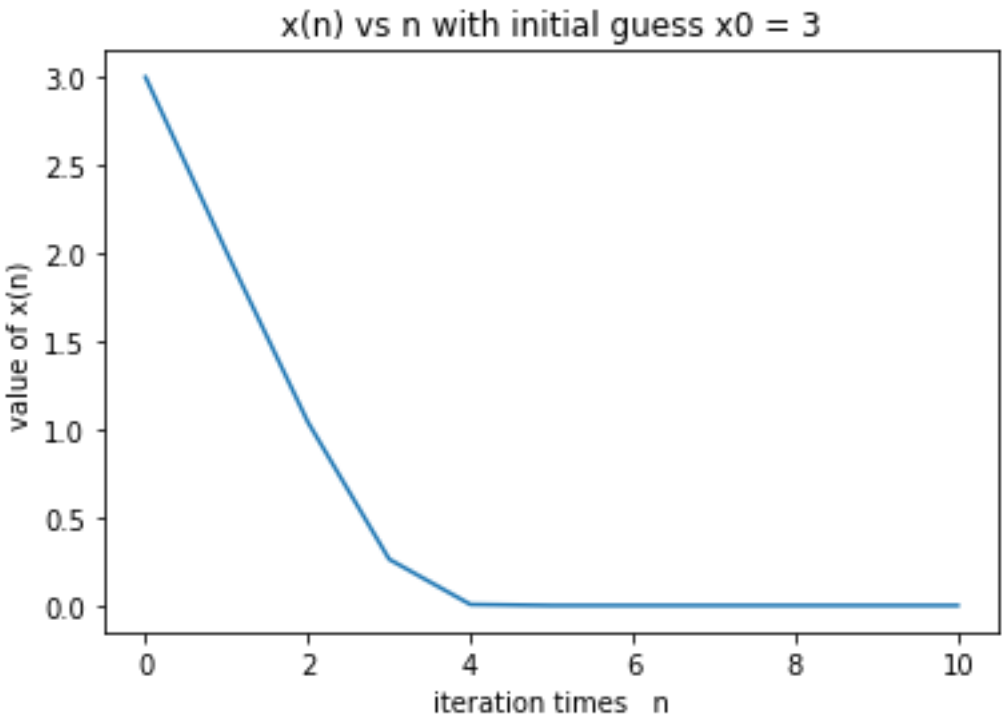
Problem4

(c)


f(x)
Q(x,-2)
Q(x,3)
corresponding value of functions
value of x
25
20
15
10
5
0
−4
−2
0
2
4

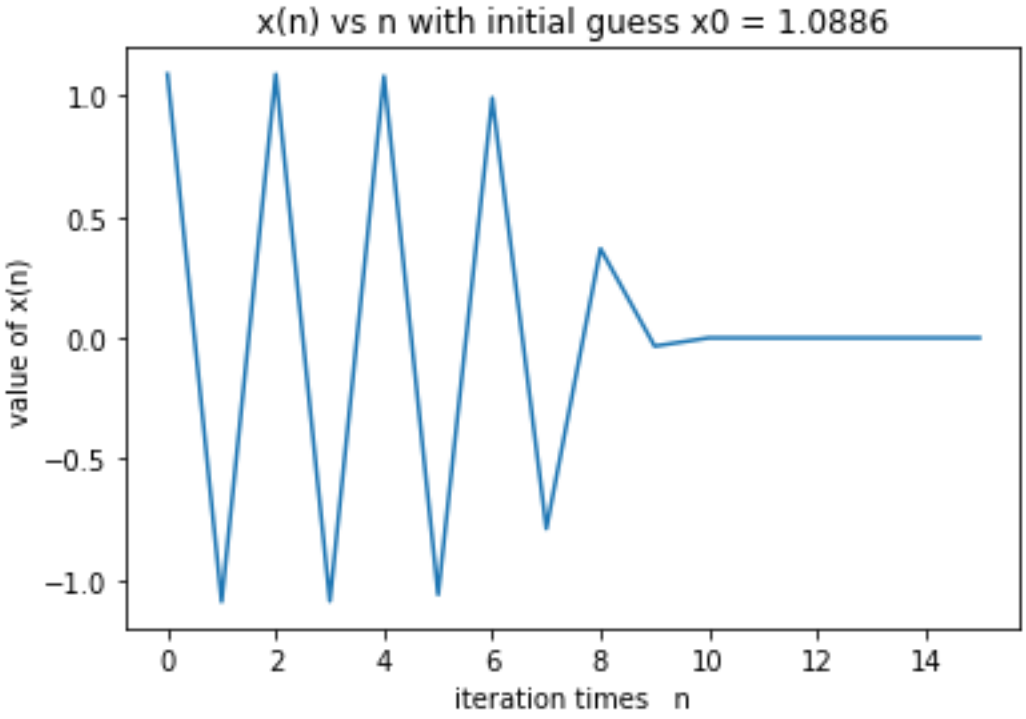
(f)


x(n) vs n with initial guess x0 = -2
value of x(n)
iteration times n
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
−1.50
−1.75
−2.00
0
2
4
6
8
10

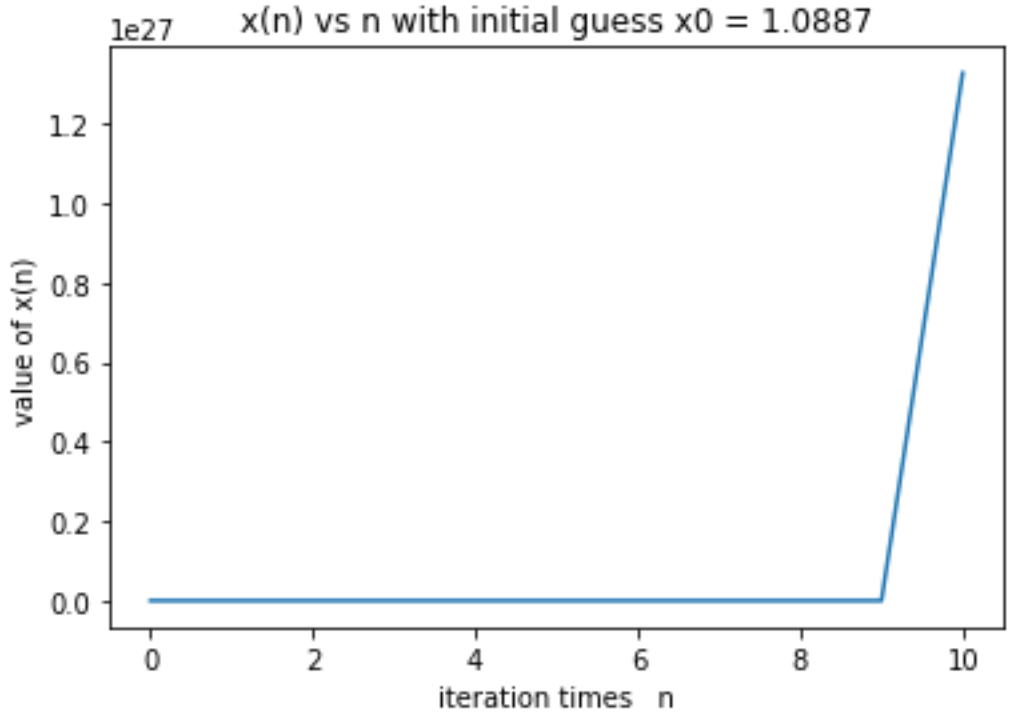


(g)

I find the upper bound is 1.0886. and I draw the plot of x(n) versus n

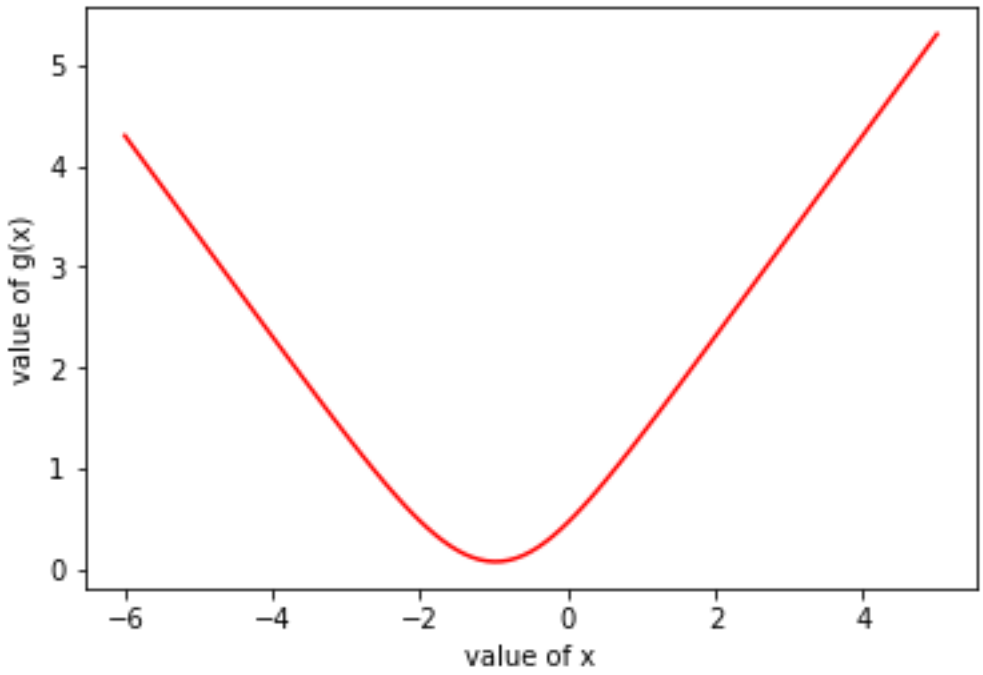


When I change the value to 1.0887, the results is following. It doesn't converge.



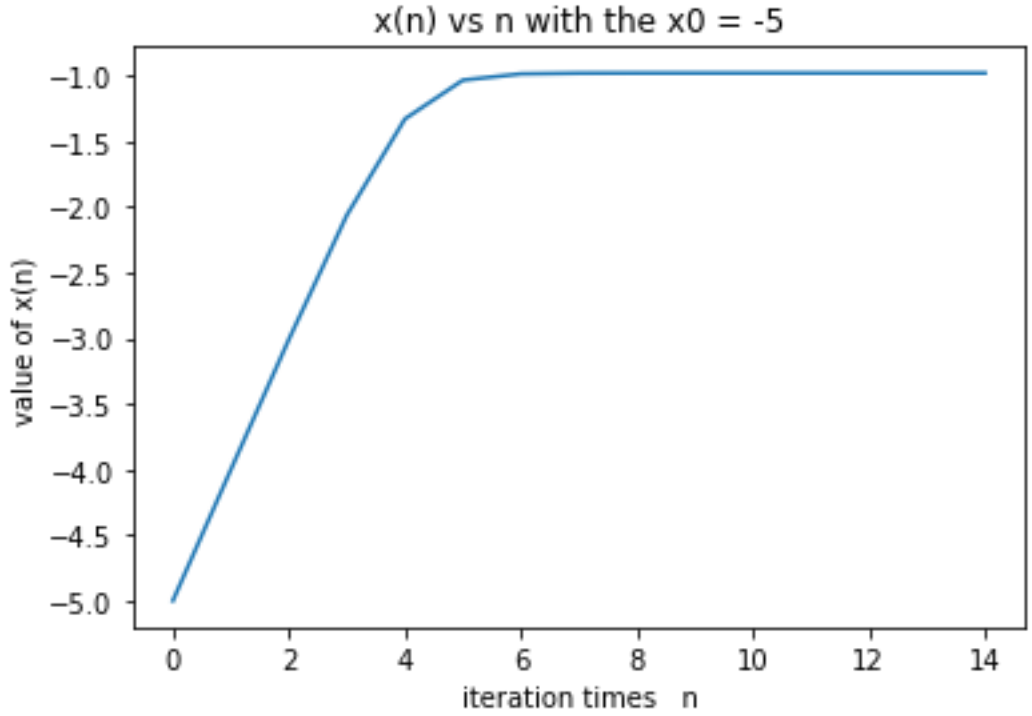
(h)

g(x)



(K)

random choose a start point x0 = -5, minimum value is at x = -0.9800



Sourcecode:

```
import matplotlib.pyplot as plt
import numpy as np
from numpy import cosh
from math import log
from math import exp
#part (c)
def function(t):
    return np.log(cosh(t))
def order1(x):
    return (np.exp(x)-np.exp(-x))/(np.exp(x)+np.exp(-x))
def order2(x):
    return 4/(np.exp(x)+np.exp(-x))**2
def Q(x,y):
    return function(y)+order1(y)*(x-y)+0.5*(x-y)**2
t = np.arange(-5,5.1,0.5)
```

```
plt.plot(t,function(t).astype(np),'r',label='f(x)')
plt.plot(t,Q(t,-2).astype(np),'g',label='Q(x,-2)')
plt.plot(t,Q(t,3).astype(np),'b',label='Q(x,3)')
plt.legend(loc='upper right')
plt.xlabel('value of x')
plt.ylabel('corresponding value of functions')
plt.show()
#part (f)
x_begin = -2  #generate a random x0 to begin
def update(x):
   x_new = x-order1(x)
   return x_new
iteration_time = 10
xval=[x_begin]
for i in range(iteration_time):
   x_begin = update(x_begin)
   xval.append(x_begin)
n=list(range(iteration_time+1))
plt.plot(n,xval)
plt.xlabel('iteration times   n')
plt.ylabel('value of x(n)')
plt.title('x(n) vs n with initial guess x0 = -2')
plt.show()

x_begin = 3  #generate a random x0 to begin
def update(x):
   x_new = x-order1(x)
   return x_new
iteration_time = 10
xval=[x_begin]
for i in range(iteration_time):
   x_begin = update(x_begin)
   xval.append(x_begin)
n=list(range(iteration_time+1))
plt.plot(n,xval)
plt.xlabel('iteration times   n')
plt.ylabel('value of x(n)')
plt.title('x(n) vs n with initial guess x0 = 3')
plt.show()

#(g)
t = np.arange(1.0875,1.09,0.0001)
plt.plot(t,(abs(order1(t)/order2(t))/abs(t)).astype(np),'r',label='f(x)')
plt.show()
```

```
x_begin = 1.0886 #generate a random x0 to begin
def update(x):
    x_new = x-order1(x)/order2(x)
    return x_new
iteration_time = 15
xval=[x_begin]
for i in range(iteration_time):
    x_begin = update(x_begin)
    xval.append(x_begin)
n=list(range(iteration_time+1))
plt.plot(n,xval)
plt.xlabel('iteration times   n')
plt.ylabel('value of x(n)')
plt.title('x(n) vs n with initial guess x0 = 1.0886')
plt.show()
x_begin = 1.0887 #generate a random x0 to begin
def update(x):
    x_new = x-order1(x)/order2(x)
    return x_new
iteration_time = 15
xval=[x_begin]
for i in range(iteration_time):
    x_begin = update(x_begin)
    xval.append(x_begin)
n=list(range(iteration_time+1))
plt.plot(n,xval)
plt.xlabel('iteration times   n')
plt.ylabel('value of x(n)')
plt.title('x(n) vs n with initial guess x0 = 1.0887')
plt.show()
```

#(h)

```
def g(x):
    result=0
    for i in range(10):
        k=i+1
        result += np.log(cosh(x+2/k**(1/2.0)))
    result = result/10
    return result
t = np.arange(-6,5.1,0.1)
plt.plot(t,g(t).astype(np),'r')
plt.xlabel('value of x')
plt.ylabel('value of g(x)')
plt.show()
```

```
#(K)
def g1(x):
    result = 0
    for i in range(10):
        k=i+1
        result += (np.exp(x+2/k**(1/2.0))-np.exp(-x-
2/k**(1/2.0)))/(np.exp(x+2/k**(1/2.0))+np.exp(-x-2/k**(1/2.0)))
    result = result/10
    return result
x_begin = -5  #generate a random x0 to begin
def update(x):
    x_new = x-g1(x)
    return x_new
iteration_time = 14
xval=[x_begin]
for i in range(iteration_time):
    x_begin = update(x_begin)
    #print(x_begin)
    xval.append(x_begin)
n=list(range(iteration_time+1))
plt.plot(n,xval)
plt.xlabel('iteration times   n')
plt.ylabel('value of x(n)')
plt.title('x(n) vs n with the x0 = -5')
plt.show()
```