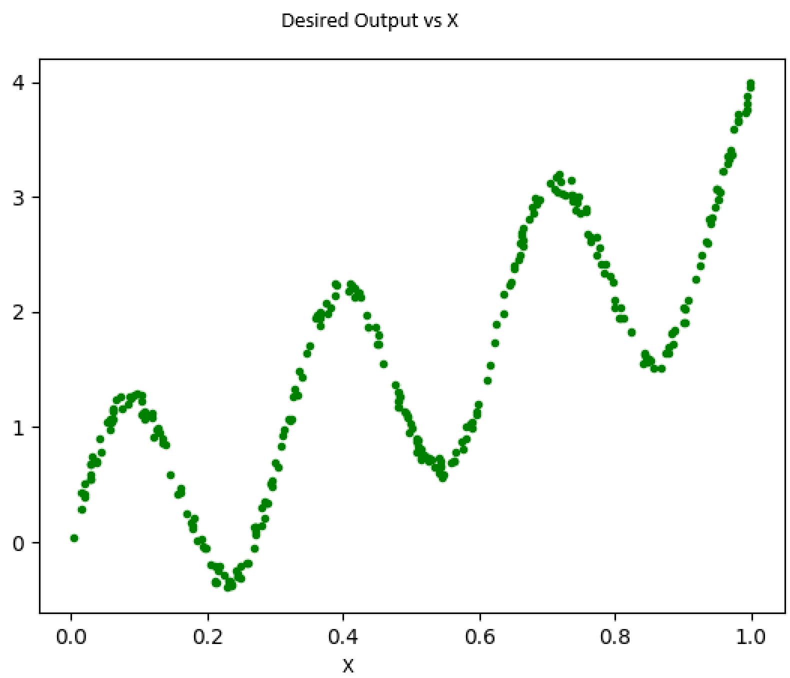
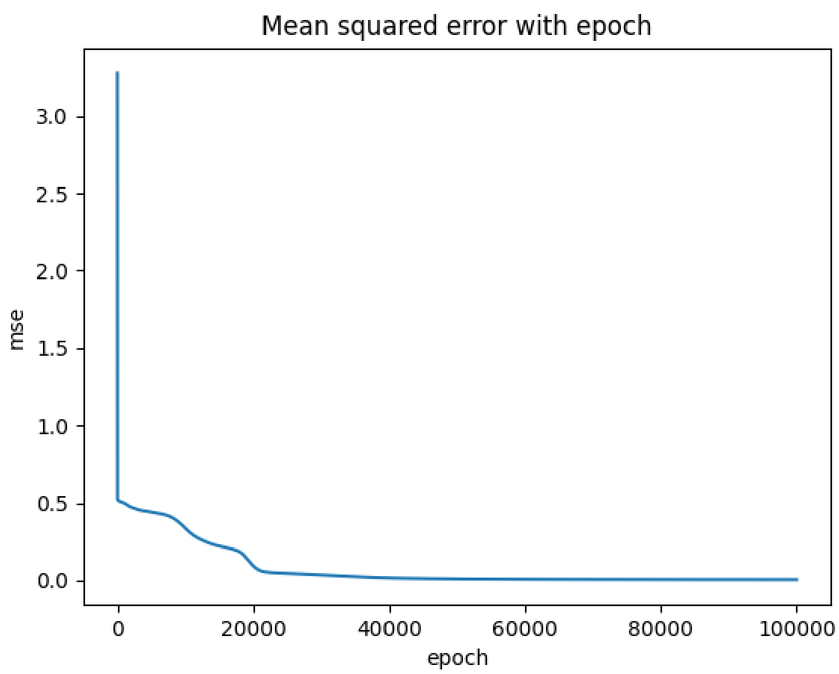
Neural Networks Assignment 4

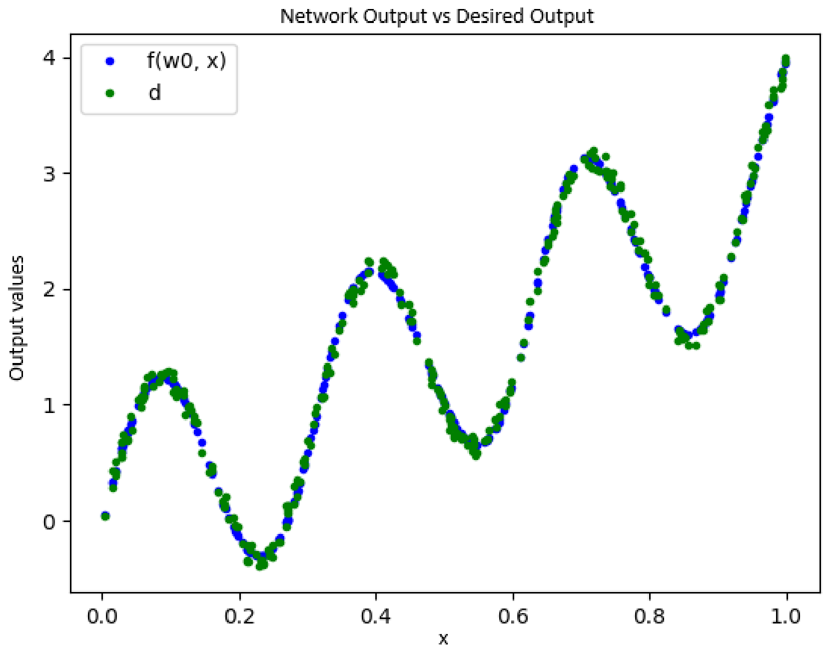
Here is the graph of Desired Outputs and x values:



Here is the graph of MSE vs Epochs:



Here is the graph of final output values ($f(w0, x)$) vs x values:



Pseudocode:

$$E(W) = \sum_{i=1}^{24} (d_i - f(w, x_i))^2$$

The energy function Mean squared error is defined above.

Let U be the weights between Hidden Layer and output Neuron.

Let W be the weights between input layer and Hidden Layer.

Let B be the biases for hidden layer

Let c be the bias for output neuron

Let x be the input values

Let D be the desired outputs

pseudocode:

Method backpropagation takes arguments u, w, b, c, x, d{

Repeat till E(w) is minimum{

For I in length of X{

    Output1 = W * X[i] + B

    Output2 = c + matrix_multiplication(u , tanh(output1))

```
        u = u + eta * (d[i] - output2) * np.tanh(output1)
        w = w + eta * (d[i] - output2) * (1 - tanh(output1)**2) * x[i] * u
        b = b + eta * (d[i] - output2) * (1 - tanh(output1)**2) * u
        c = c + eta * (d[i] - output2) * 1
        }
}
```

Code:

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

def get_tanh(index):
        global b
        global w
        global x
        return np.tanh(b[index] + w[index]*x[index])

def output(w, u, b, c, input):
        v = np.add(b, input*w)
        return np.add(c, np.matmul(u, np.tanh(v)))

def getmse(d, x, w, u, b, c):
        sum = 0
        for i in range(len(d)):
                sum += (d[i] - output(w, u, b, c, x[i]))**2
        return sum/300

def updateWeights(w, x, eta, d, c, u, b):
```

```
    epoch = 0
    mse = [getmse(d, x, w, u, b, c)]

    while(epoch<100000):
        curr_mse = getmse(d, x, w, u, b, c)
        for i in range(300):

            output1 = w * x[i] + b
            output2 = np.matmul(u, np.tanh(output1).T) + c

            u = u + eta * (d[i] - output2) * np.tanh(output1)
            w = w + eta * (d[i] - output2) * (1 - np.tanh(output1)**2) * x[i] * u
            b = b + eta * (d[i] - output2) * (1 - np.tanh(output1)**2) * u
            c = c + eta * (d[i] - output2) * 1

        if getmse(d, x, w, u, b, c) > curr_mse:
            eta = 0.9*eta
        epoch += 1
        mse.append(getmse(d, x, w, u, b, c))
        if(epoch%100 == 0):
            print(getmse(d, x, w, u, b, c))
            print('epoch=' + str(epoch))
            print('eta = '+str(eta))
    return w, u, b, c, mse


np.random.seed(121)


w = np.random.rand(24)
u = np.random.uniform(-0.1, 0.1, 24)
```

```
b = np.random.uniform(-0.1, 0.1, 24)
c = np.random.uniform(-0.1, 0.1, 1)
x = np.random.uniform(0, 1, 300)
v = np.random.uniform(-0.10, 0.10, 300)
d = np.empty([300])

for i in range(300):
    d[i] = math.sin(20*x[i]) + 3*x[i] + v[i]

plt.plot(x, d, 'g.')
plt.show()

w, u, b, c, mse = updateWeights(w, x, 0.001, d, c, u, b)

y = []
for input in x:
    y.append(output(w, u, b, c, input))

plt.plot(x, y, 'b.', label = 'f(w0, x)')
plt.plot(x, d, 'g.', label = 'd')
plt.legend()
plt.show()

plt.plot(mse)
plt.title('Mean squared error with epoch')
plt.xlabel('epoch')
plt.ylabel('mse')
plt.show()
```