Name: Venkata Meghana Achanta

USC ID : 2578990261

Q1. The solution for question 1 has been depicted in the picture shown. The output activation for the given input is [8 4].

Q1. Given,

$$W_1 = \begin{bmatrix} 1 & -2 \\ 3 & 4 \end{bmatrix} \quad W_2 = \begin{bmatrix} 2 & 2 \\ 2 & -3 \end{bmatrix}$$

$$b_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad b_2 = \begin{bmatrix} 0 \\ -4 \end{bmatrix}$$

$$a_1 = h_1[W_1 2 + b_1]$$

$$a_1 = h_1\left[\begin{bmatrix} 1 & -2 \\ 3 & 4 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right]$$

$$a_1 = h_1\left[\begin{bmatrix} 4 \\ -1 \end{bmatrix}\right] \quad ; h_1 = max(x, 0)$$

$$a_1 = \begin{bmatrix} 4 \\ 0 \end{bmatrix}$$

$$a_2 = h_2[W_2 a_1 + b_2]$$

$$a_2 = h_2\left[\begin{bmatrix} 2 & 2 \\ 2 & -3 \end{bmatrix}\begin{bmatrix} 4 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -4 \end{bmatrix}\right]$$

$$a_2 = h_2\begin{bmatrix} 8 \\ 4 \end{bmatrix}$$

$$a_2 = \begin{bmatrix} 8 \\ 4 \end{bmatrix} \quad ; h_2 = \text{identity activation function.}$$

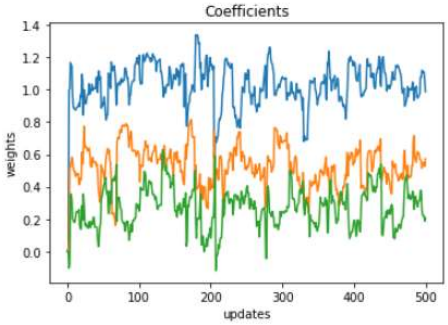
Q2. For this question, I manually typed the 25 binary sequences at random to generate a .hd5 file. The DEBUG flag was set to False and I modified the file name from:

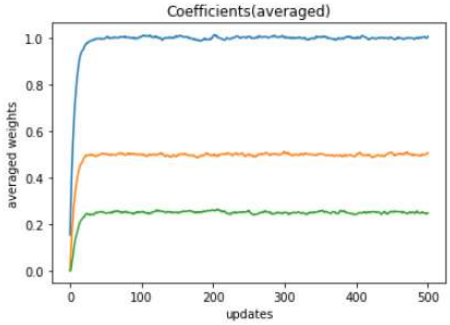
DATA_FNAME = 'brandon_franzke_hw1_1.hd5' to DATA_FNAME = 'meghana_hw3_2.hd5'

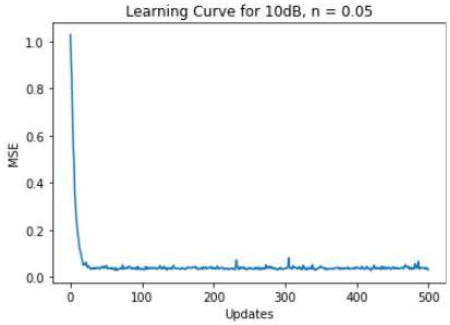
The code has been attached in the appendix. The .hd5 has been submitted to Autolab.

Q3.a) i. After programing the LMS algorithm for the SNR's 3dB and 10dB for learning rates 0.05 and 0.15, I observed that the filter learnt coefficients with less fluctuations and high convergence for 0.05 as learning rate compared to 0.15. The results for the same can be observed in ii.

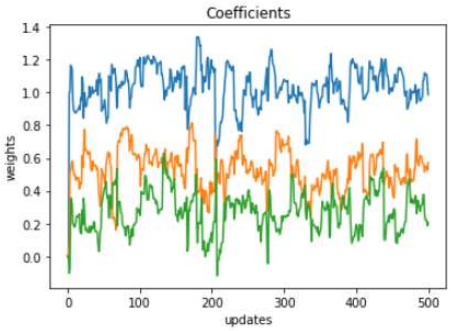
ii. The plots for 10dB noise and n = 0.05 is as follows:

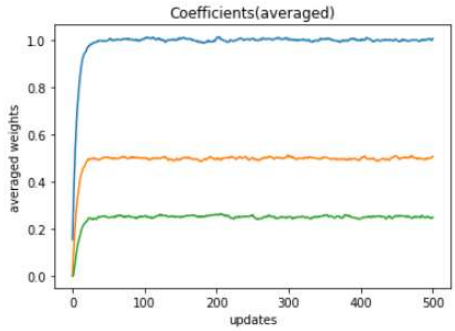


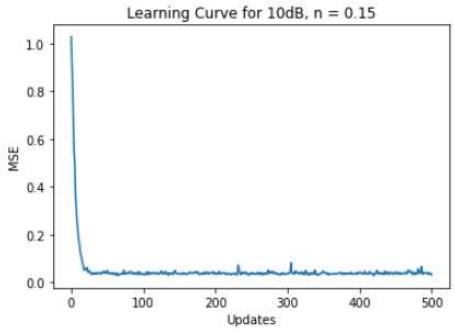




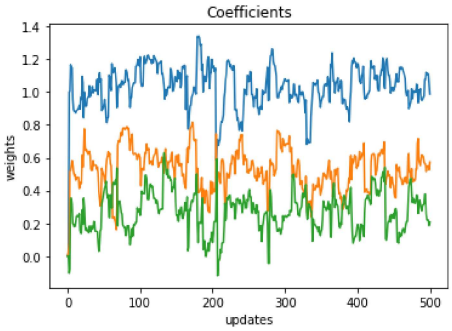
The plots for 10dB SNR for n = 0.15 is as follows:

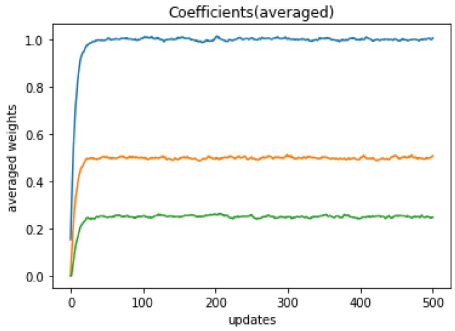


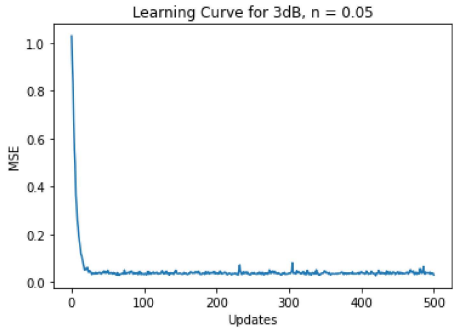




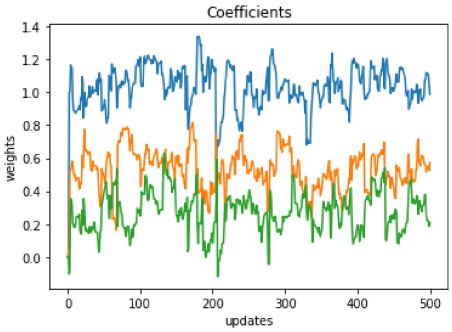
The plots for 3dB SNR for n = 0.05 is as follows:

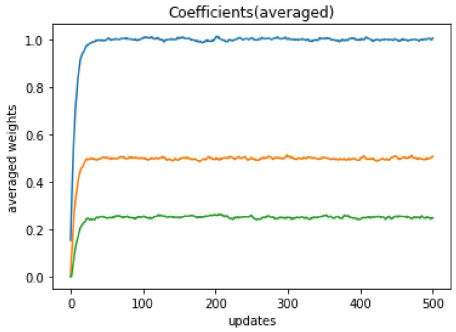


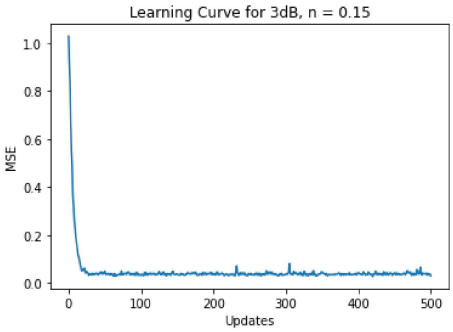




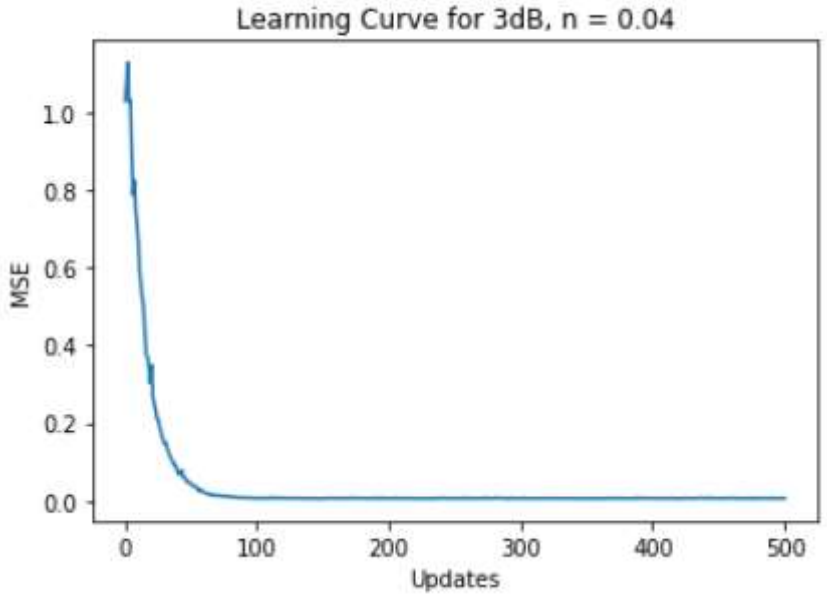
The plot for 3dB SNR at n = 0.15 is as follows:

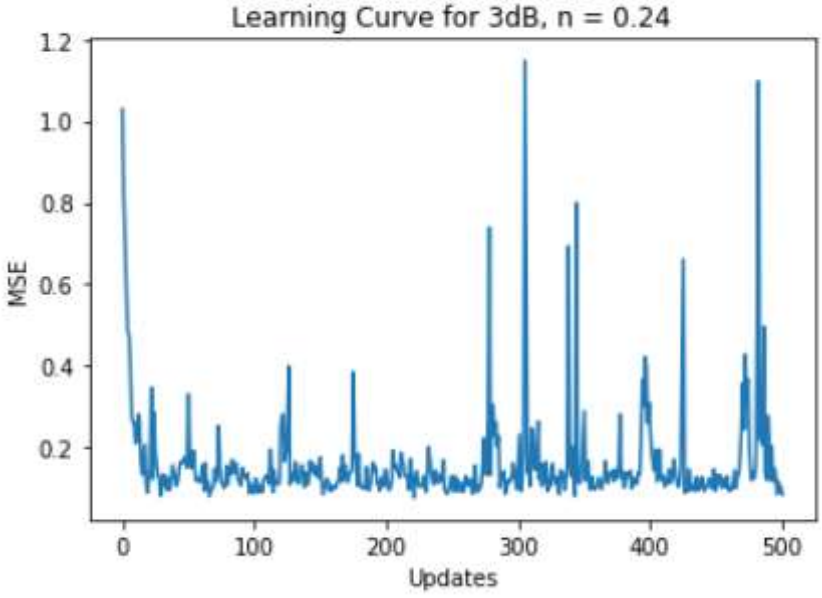


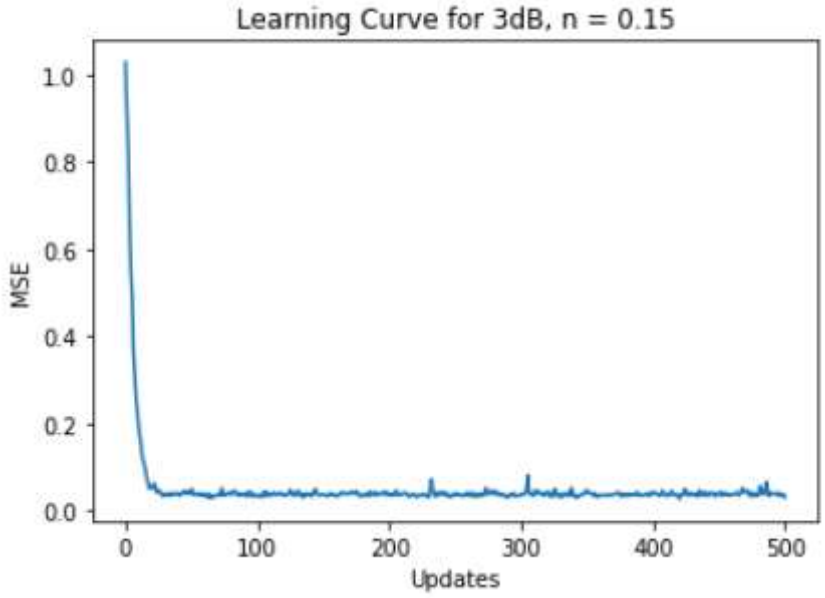




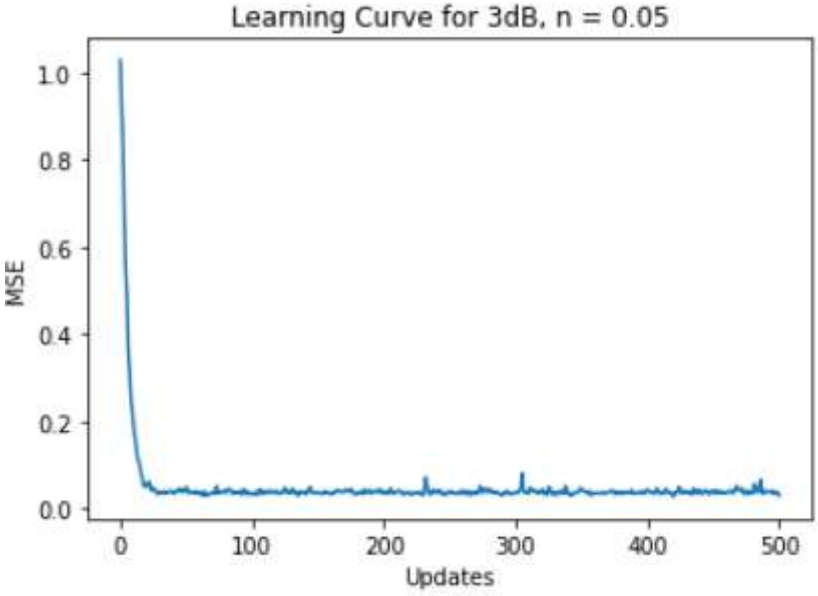
iii. To find the largest value of n that does not lead to divergence, I chose 4 learning rates in the range from 0.4 to 0.3 for both the SNRS. For SNR 3dB, I observed the learning curves for values 0.04, 0.05, 0.15,0.24. We can see that for learning rate 0.04, there are minimum fluctuations and the learning rate around 0.24 is the maximum value before theMSE starts to diverge.



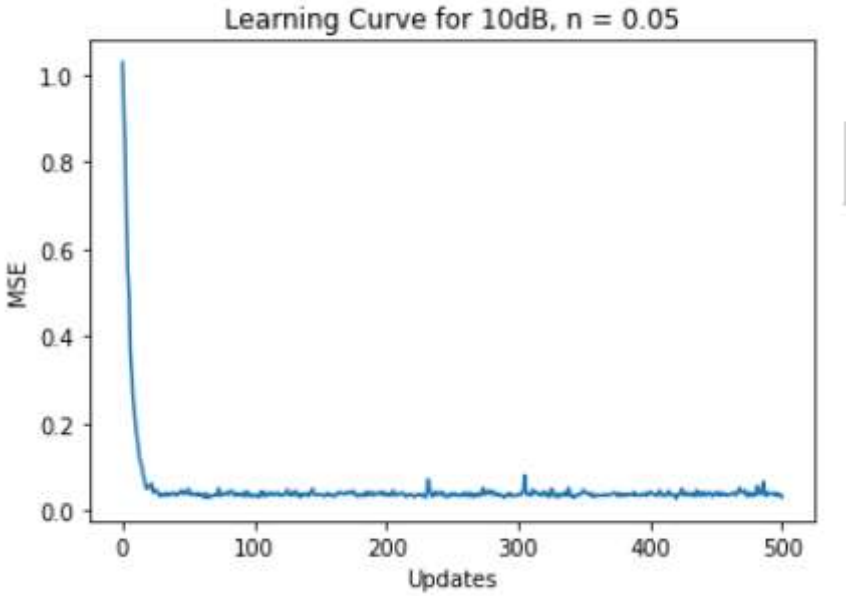


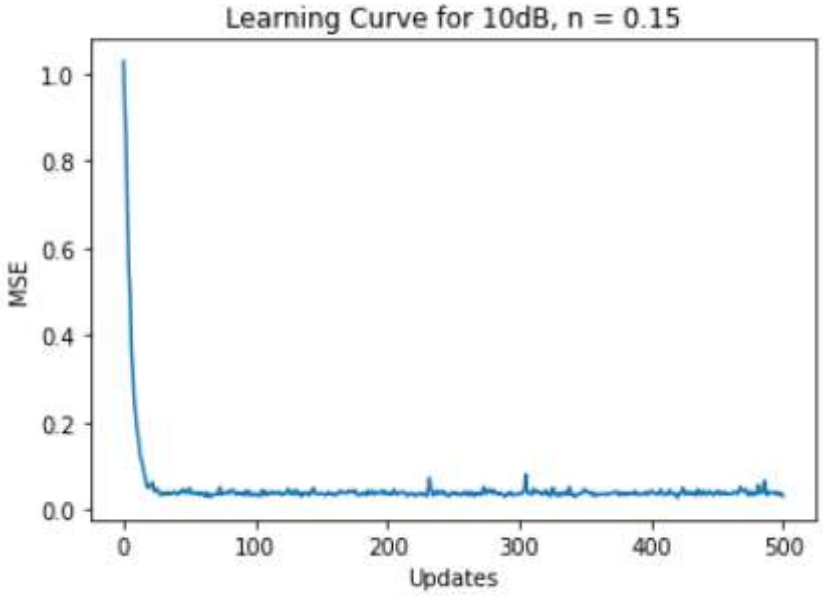


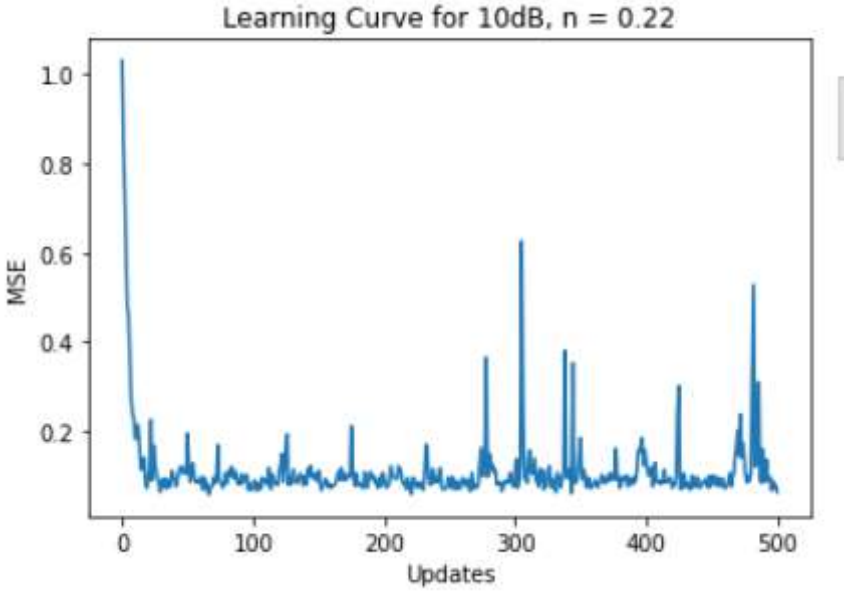
!



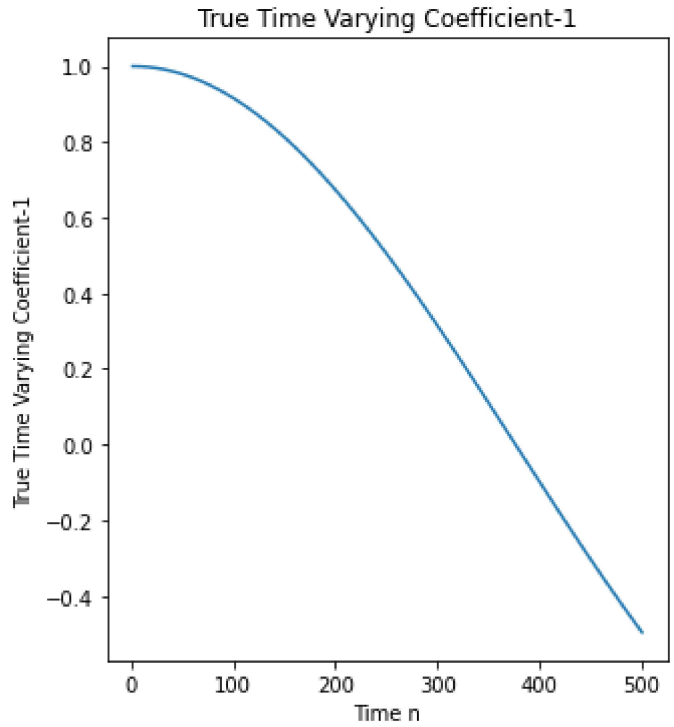
For SNR 10dB, I observed the learning curves for values 0.05, 0.15 and 0.22. We can see that for learning rate 0.05, there are minimum fluctuations and the learning rate around 0.22 is the maximum value before the MSE starts to diverge.The largest value of n that does not lead to divergence of n is around 0.22. When the n value goes above 0.22, the MSE starts to increase exponentially leading to divergence.

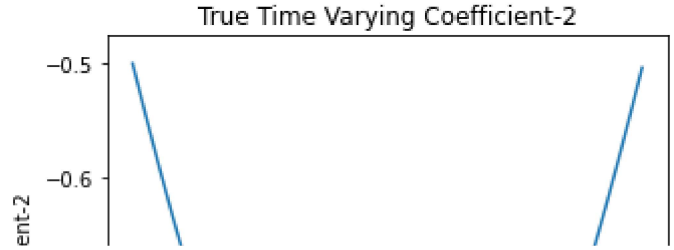

Learning Curve for 10dB, n = 0.05
MSE
Updates
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500

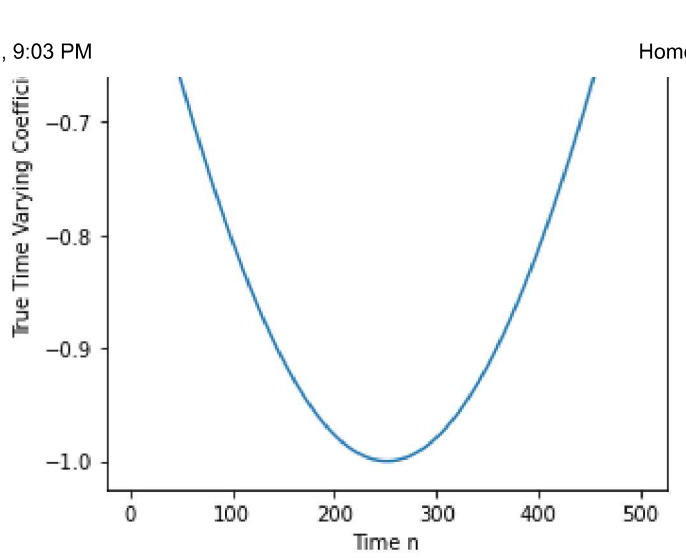

Learning Curve for 10dB, n = 0.15
MSE
Updates
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500

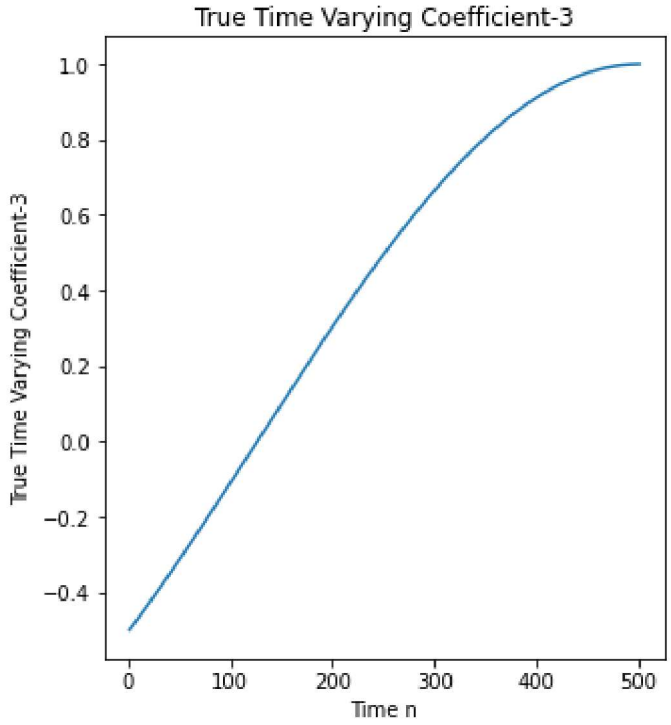


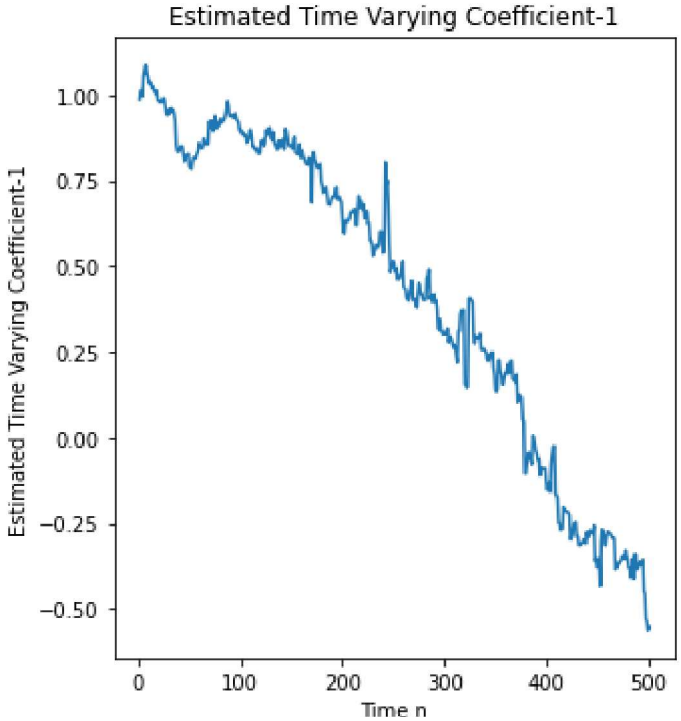
Q3b. After experimenting, I chose the learning rate as 0.08 to track the variations between true coefficients and estimated coefficients. I observed that this learning rate estimated the new coefficients which were close to the values of true coefficients. The plot for coefficients vs time and true coefficients vs estimated coefficients is as follows:

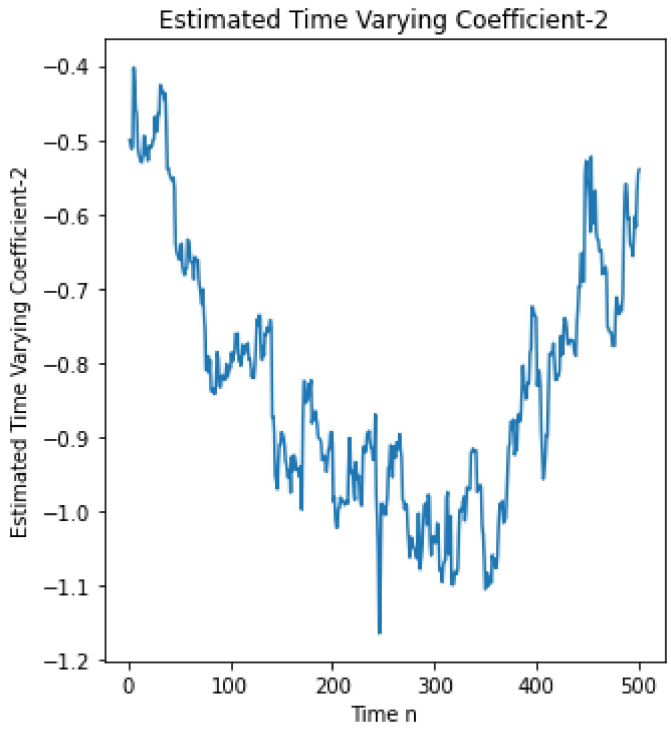


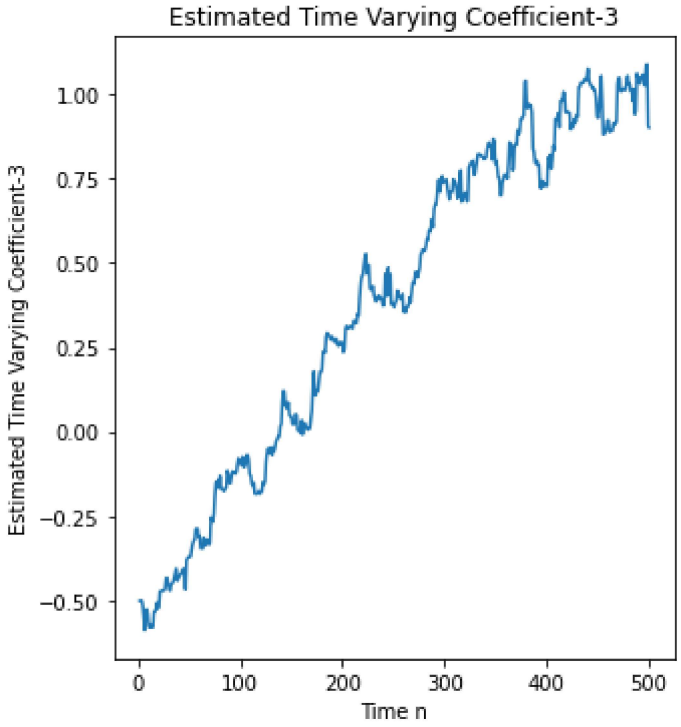


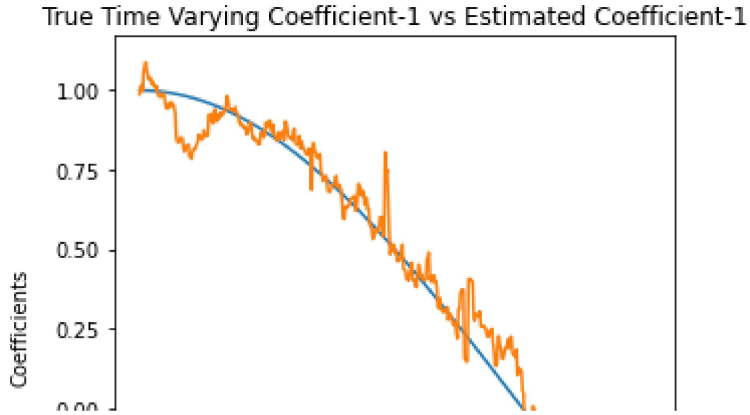

True Time Varying Coeffici
−0.7
−0.8
−0.9
−1.0
0
100
200
300
400
500
Time n

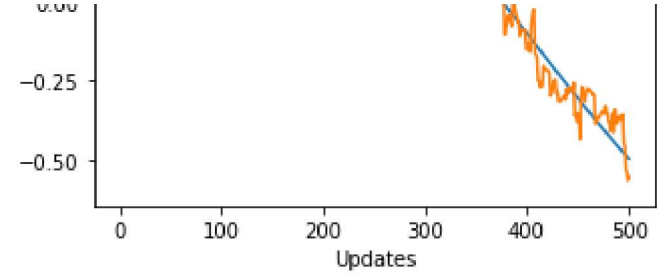

True Time Varying Coefficient-3
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0
100
200
300
400
500
Time n

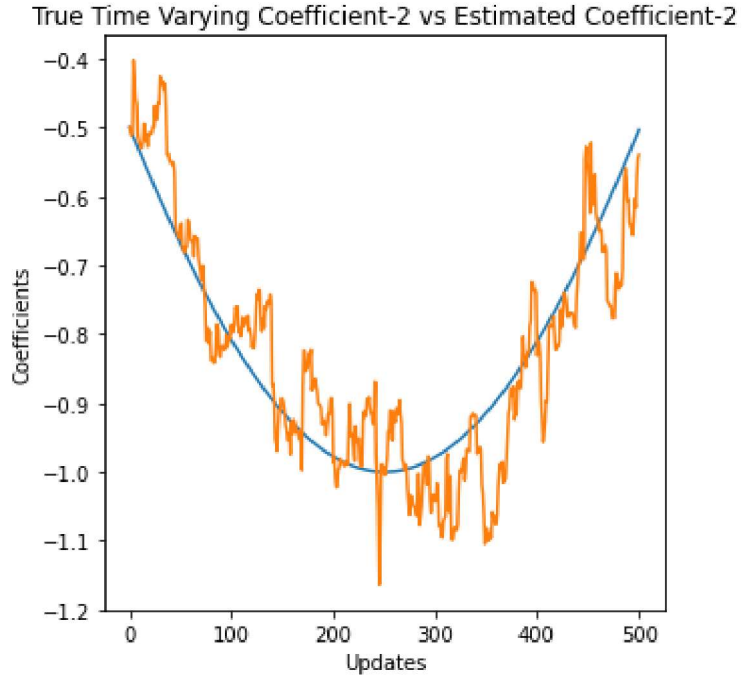

Estimated Time Varying Coefficient-1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
0
100
200
300
400
500
Time n

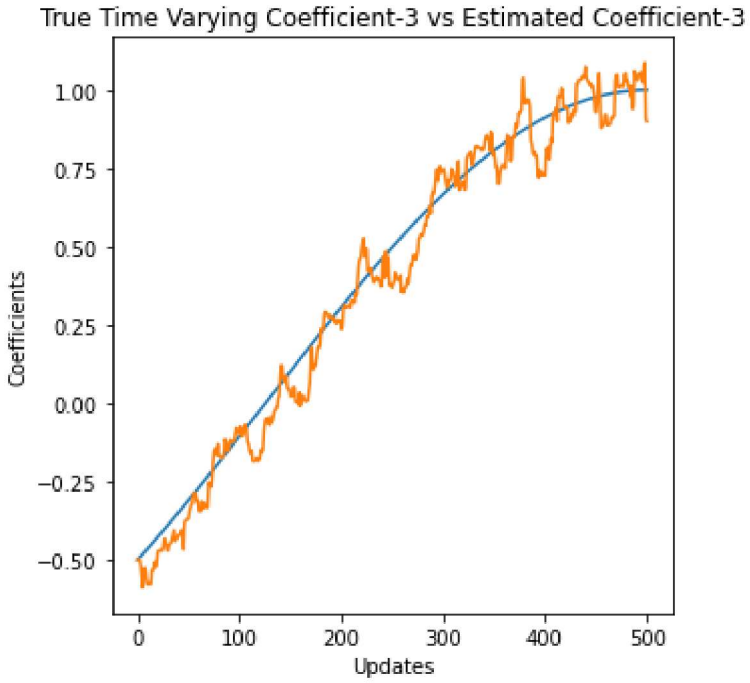

Estimated Time Varying Coefficient-2
Estimated Time Varying Coefficient-2
−0.4
−0.5
−0.6
−0.7
−0.8
−0.9
−1.0
−1.1
−1.2
0
100
200
300
400
500
Time n


Estimated Time Varying Coefficient-3
Estimated Time Varying Coefficient-3
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
0
100
200
300
400
500
Time n


True Time Varying Coefficient-1 vs Estimated Coefficient-1
Coefficients
1.00
0.75
0.50
0.25



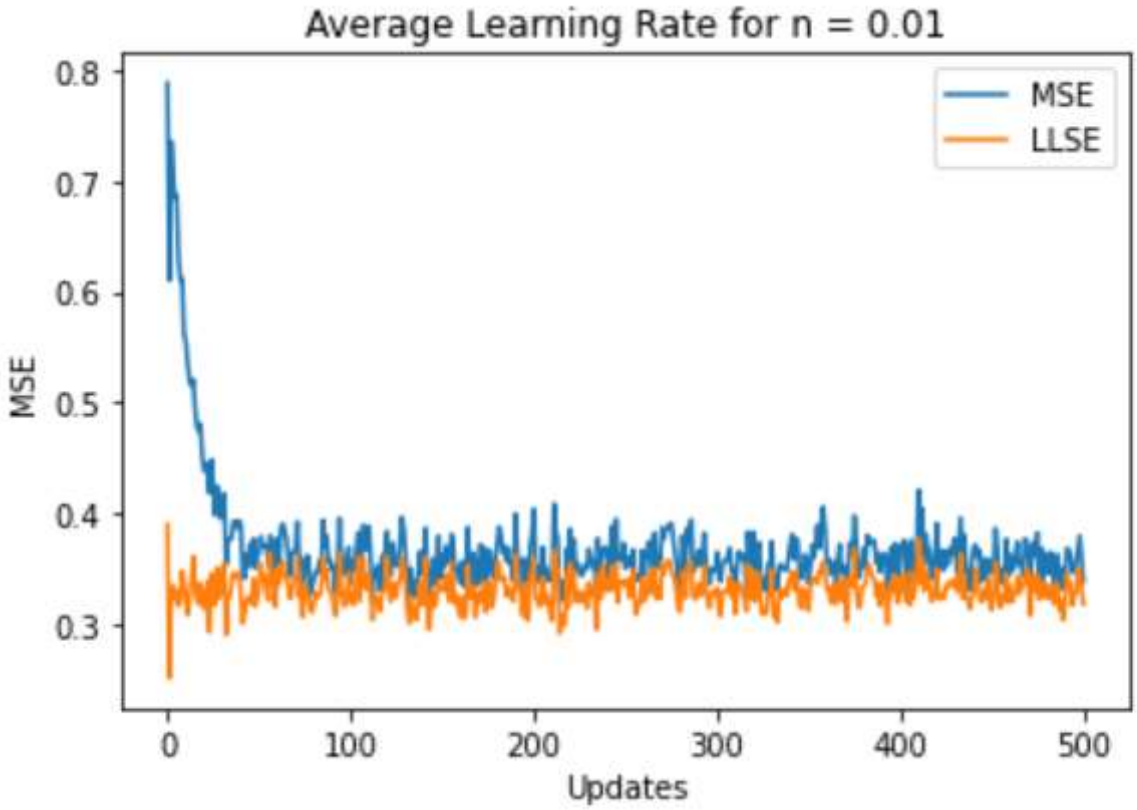




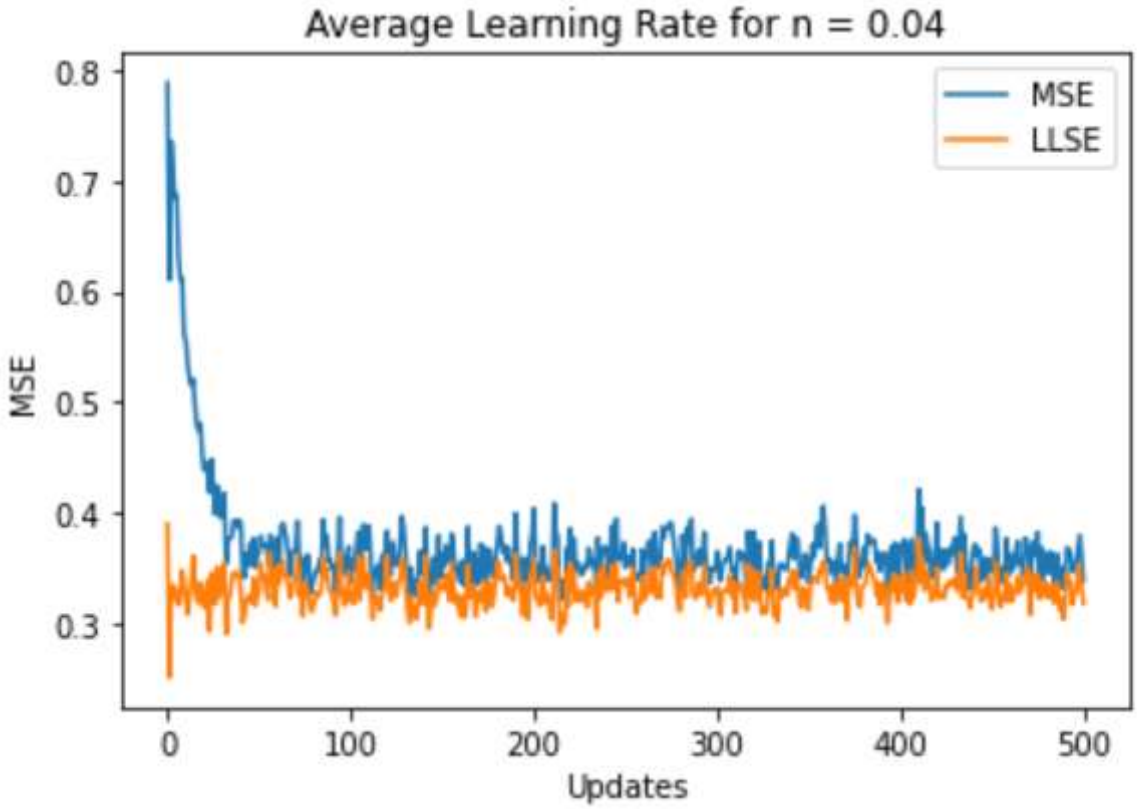
Q3c. For lower values of n i.e, about 0.01, 0.04, the MSE is below LLSE. However, as the values of n increases (above 0.05) the MSE goes above the values of LLSE. Hence for lower values of learning rates(n< 0.05), LLSE is not lesser that MSE after convergence whereas for higher values of learning rates (n > 0.05), LLSE is lower than MSE after convergence.

The various plots of LLSE and MSE for different values of n are as follows:

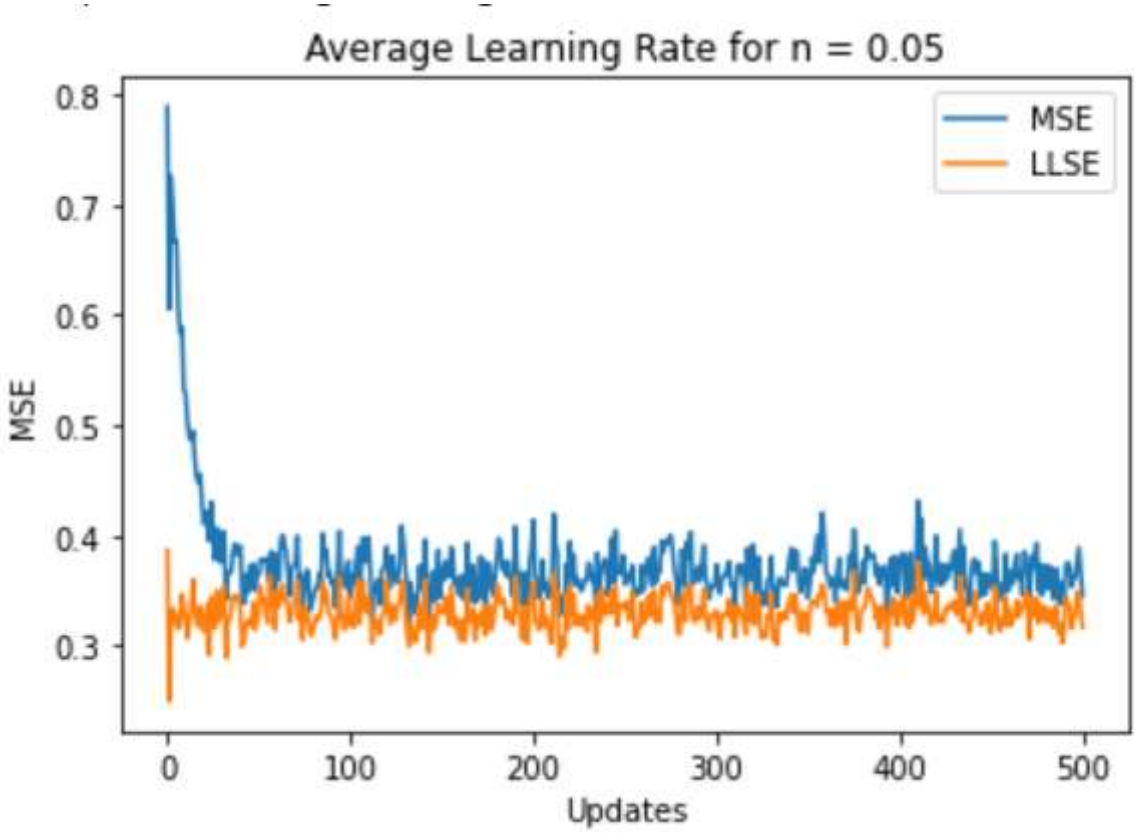
i. Plot of n = 0.01



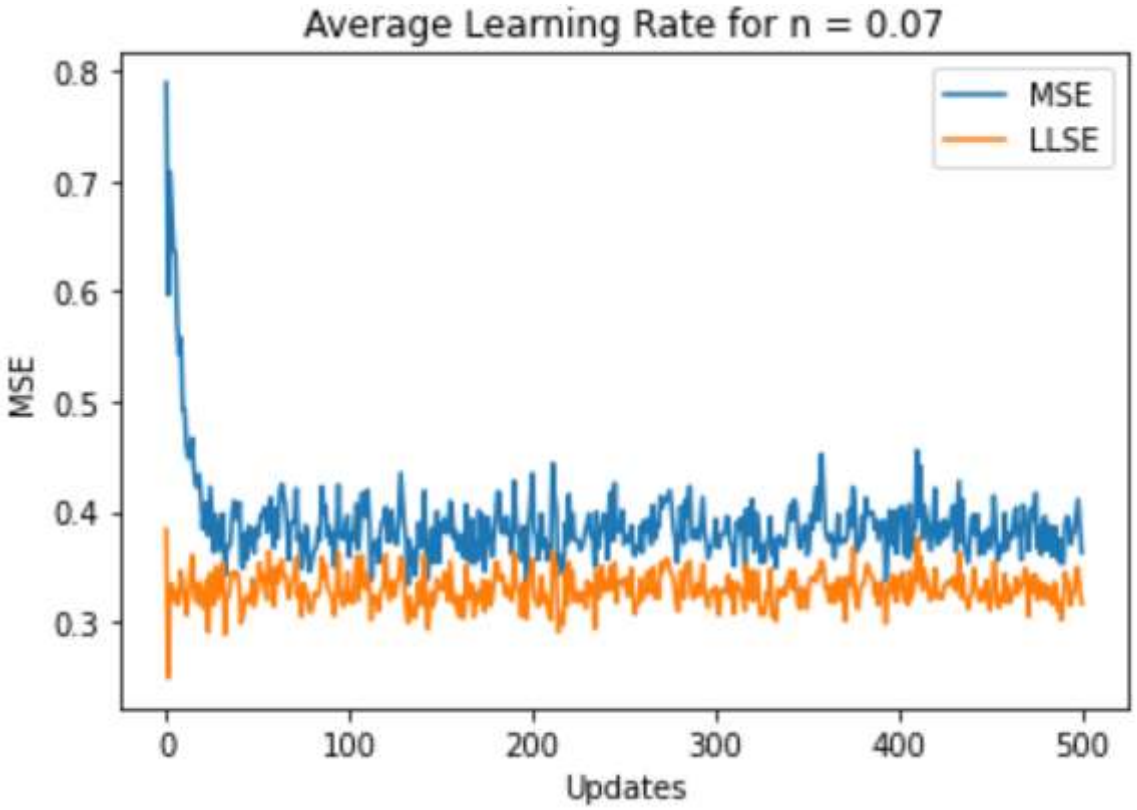
ii. Plot of n = 0.04



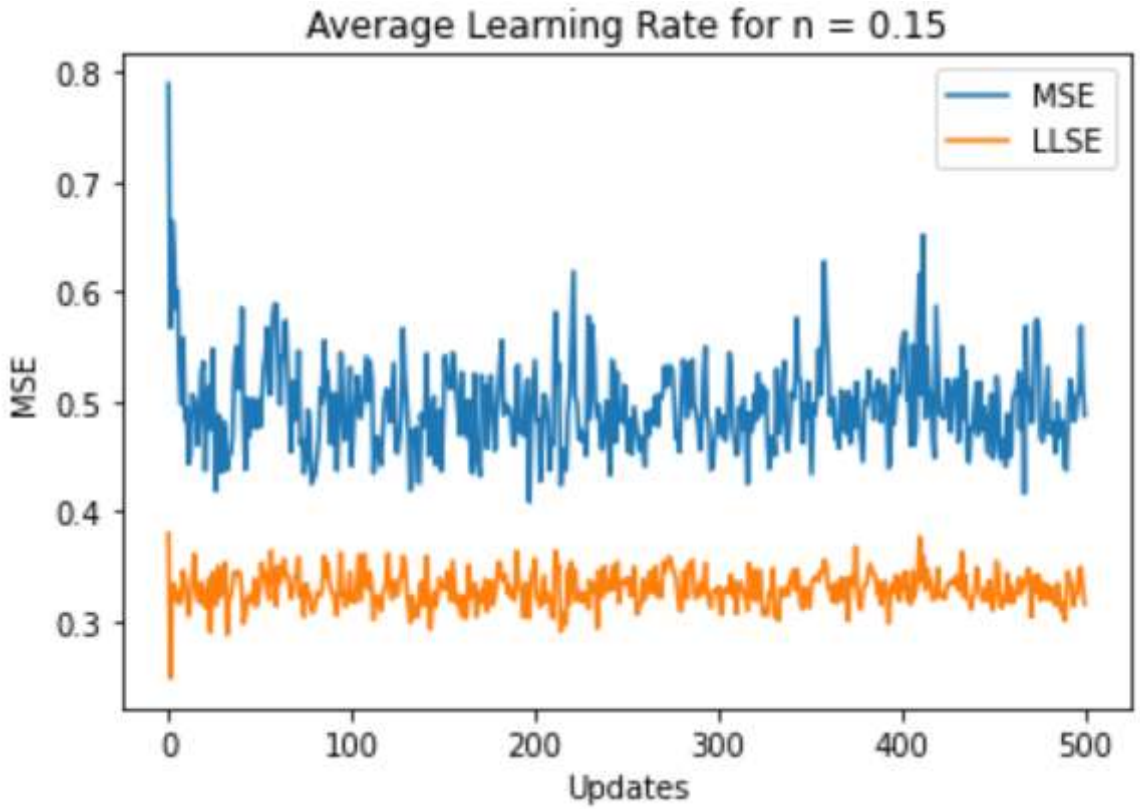
iii. Plot of n = 0.05



iv. Plot of n = 0.07



v. Plot of n = 0.15



Appendix

```
#Question 2

## copyright, Keith Chugg
##  EE599, 2020

#####################################################
## this is a template to illustrate hd5 files
##
## also can be used as template for HW1 problem
#####################################################

import h5py
import numpy as np
import matplotlib.pyplot as plt

DEBUG = False
DATA_FNAME = 'meghana_hw3_2.hd5'

if DEBUG:
    num_sequences = 3
    sequence_length = 4
else:
    num_sequences = 25
    sequence_length = 20
```

```
### Enter your data here...
### Be sure to generate the data by hand.  DO NOT:
###     copy-n-paste
###     use a random number generator
###
x_list = [[0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,1,1,0],
          [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
          [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
          [1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0],
          [0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1],
          [1,0,0,1,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0],
          [1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0],
          [0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1],
          [1,1,0,0,1,1,1,1,1,1,0,0,0,1,1,1,0,0,1,0],
          [0,0,0,1,1,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0],
          [1,0,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0],
          [1,1,1,1,1,1,0,0,0,0,0,0,1,1,1,1,1,1,1,1],
          [0,0,0,0,1,1,1,1,0,0,0,1,1,1,1,1,0,0,0,1],
          [1,1,1,0,0,1,1,1,0,0,1,1,1,0,0,1,1,1,0,0],
          [1,1,1,1,1,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0],
          [1,1,1,0,1,1,1,0,1,1,1,0,1,1,1,0,1,1,1,0],
          [0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0],
          [1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0],
          [1,1,0,0,1,1,0,0,0,0,0,0,1,1,1,1,0,0,1,1],
          [0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1],
          [0,0,1,1,0,0,1,1,1,1,0,0,0,1,1,1,0,0,0,1],
          [0,1,1,1,1,0,0,0,0,0,1,1,1,0,0,0,1,1,0,0],
          [0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,1,1,1,0,0],
          [1,1,1,1,1,1,0,1,1,1,1,1,1,0,1,1,1,1,1,1],
          [0,1,1,1,0,1,1,1,1,1,1,1,1,0,0,1,1,1,0,0]
          ]
# x_list = [
#     [ 0, 1, 1, 0],
#     [ 1, 1, 0, 0],
#     [ 0, 0, 0, 1]
# ]

# convert list to a numpy array...
human_binary = np.asarray(x_list)

### do some error trapping:

assert human_binary.shape[0] == num_sequences, 'Error: the number of sequences was entered in
assert human_binary.shape[1] == sequence_length, 'Error: the length of the seqeunces is incor

# the with statement opens the file, does the business, and close it up for us...
with h5py.File(DATA_FNAME, 'w') as hf:
    hf.create_dataset('human_binary', data = human_binary)
    ## note you can write several data arrays into one hd5 file, just give each a different n
```

```
####################
# Let's read it back from the file and then check to make sure it is as we wrote...
with h5py.File(DATA_FNAME, 'r') as hf:
    hb_copy = hf['human_binary'][:]

### this will throw and error if they are not the same...
np.testing.assert_array_equal(human_binary, hb_copy)



#Question 3a

import h5py
import numpy as np
import matplotlib.pyplot as plt

data = h5py.File('lms_fun_v3.hdf5', 'r')

#For SNR 10dB and n = 0.05
x_10 = np.asarray(data['matched_10_x'])
v_10 = np.asarray(data['matched_10_v'])
y_10 = np.asarray(data['matched_10_y'])
z_10 = np.asarray(data['matched_10_z'])


wn = np.array(wn)
w1 = []
y = np.zeros((600,501))
sum = np.zeros((3,1))
sum1 = 0
mse = np.zeros((600,501))

for i in range(600):
  wn = np.array([0, 0, 0])
  w2 = []
  for j in range(501):
    y[i][j] = np.matmul(wn.T,v_10[i][j])
    wn = wn + 0.05*(z_10[i][j]-y[i][j])*v_10[i][j]
    w2.append(wn)
  w1.append(w2)

print(y.shape)
w1 = np.array(w1)
mse = np.sum((y_10 - y)**2, axis = 0)/600

w_avg0 = np.sum( w1[:,:,0], axis = 0)/600
w_avg1 = np.sum( w1[:,:,1], axis = 0)/600
w_avg2 = np.sum( w1[:,:,2], axis = 0)/600

plt.rcParams['figure.figsize'] = [20,4]

fig, (ax1, ax2, ax3) = plt.subplots(1,3)
```

```
ax1.plot(w1[1,:,0])
ax1.plot(w1[1,:,1])
ax1.plot(w1[1,:,2])
ax1.set_title('Coefficients')
ax1.set_xlabel('updates')
ax1.set_ylabel('weights')

ax2.plot(w_avg0)
ax2.plot(w_avg1)
ax2.plot(w_avg2)
ax2.set_title('Coefficients(averaged)')
ax2.set_xlabel('updates')
ax2.set_ylabel('averaged weights')

ax3.plot(mse)
ax3.set_title('Learning Curve for 10dB, n = 0.05')
ax3.set_xlabel('Updates')
ax3.set_ylabel('MSE')

#For SNR 10dB and n = 0.15

wn = np.array(wn)
y = np.zeros((600,501))
sum = np.zeros((3,1))
sum1 = 0
mse = np.zeros((600,501))
w1 = []

for i in range(600):
  wn = np.array([0, 0, 0])
  w2 = []
  for j in range(501):
    y[i][j] = np.matmul(wn.T,v_10[i][j])
    wn = wn + 0.22*(z_10[i][j]-y[i][j])*v_10[i][j]

    w2.append(wn)
  w1.append(w2)

print(y.shape)

w1 = np.array(w1)
mse = np.sum((y_10 - y)**2, axis = 0)/600

w_avg0 = np.sum( w1[:,:,0], axis = 0)/600
w_avg1 = np.sum( w1[:,:,1], axis = 0)/600
w_avg2 = np.sum( w1[:,:,2], axis = 0)/600

plt.rcParams['figure.figsize'] = [20,4]
fig, (ax1, ax2, ax3) = plt.subplots(1,3)
ax1.plot(w1[1,:,0])
ax1.plot(w1[1,:,1])
```

```
ax1.plot(w1[1,:,2])
ax1.set_title('Coefficients')
ax1.set_xlabel('updates')
ax1.set_ylabel('weights')

ax2.plot(w_avg0)
ax2.plot(w_avg1)
ax2.plot(w_avg2)
ax2.set_title('Coefficients(averaged)')
ax2.set_xlabel('updates')
ax2.set_ylabel('averaged weights')

ax3.plot(mse)
ax3.set_title('Learning Curve for 10dB, n = 0.22')
ax3.set_xlabel('Updates')
ax3.set_ylabel('MSE')

x_3 = np.asarray(data['matched_3_x'])
v_3 = np.asarray(data['matched_3_v'])
y_3 = np.asarray(data['matched_3_y'])
z_3 = np.asarray(data['matched_3_z'])

#For SNR 3dB and n = 0.05

wn = np.array(wn)
y = np.zeros((600,501))
sum = np.zeros((3,1))
sum1 = 0
mse = np.zeros((600,501))
w1 = []

for i in range(600):
  wn = np.array([0, 0, 0])
  w2 = []
  for j in range(501):

    y[i][j] = np.matmul(wn.T,v_3[i][j])
    wn = wn + 0.05*(z_3[i][j]-y[i][j])*v_3[i][j]
    w2.append(wn)
  w1.append(w2)

print(y.shape)
w1 = np.array(w1)
mse = np.sum((y_3 - y)**2, axis = 0)/600

w_avg0 = np.sum( w1[:,:,0], axis = 0)/600
w_avg1 = np.sum( w1[:,:,1], axis = 0)/600
w_avg2 = np.sum( w1[:,:,2], axis = 0)/600

plt.rcParams['figure.figsize'] = [20,4]
```

```
fig, (ax1, ax2, ax3) = plt.subplots(1,3)
ax1.plot(w1[1,:,0])
ax1.plot(w1[1,:,1])
ax1.plot(w1[1,:,2])
ax1.set_title('Coefficients')
ax1.set_xlabel('updates')
ax1.set_ylabel('weights')

ax2.plot(w_avg0)
ax2.plot(w_avg1)
ax2.plot(w_avg2)
ax2.set_title('Coefficients(averaged)')
ax2.set_xlabel('updates')
ax2.set_ylabel('averaged weights')

ax3.plot(mse)
ax3.set_title('Learning Curve for 3dB, n = 0.05')
ax3.set_xlabel('Updates')
ax3.set_ylabel('MSE')

#For SNR 3dB and n = 0.15
wn = np.array(wn)
y = np.zeros((600,501))
sum = np.zeros((3,1))
sum1 = 0
mse = np.zeros((600,501))

w1 = []
for i in range(600):
  wn = np.array([0, 0, 0])

  w2 = []
  for j in range(501):

    y[i][j] = np.matmul(wn.T,v_10[i][j])
    wn = wn + 0.15*(z_10[i][j]-y[i][j])*v_10[i][j]
    w2.append(wn)
  w1.append(w2)

print(y.shape)
w1 = np.array(w1)
mse = np.sum((y_10 - y)**2, axis = 0)/600

w_avg0 = np.sum( w1[:,:,0], axis = 0)/600
w_avg1 = np.sum( w1[:,:,1], axis = 0)/600
w_avg2 = np.sum( w1[:,:,2], axis = 0)/600

plt.rcParams['figure.figsize'] = [20,4]

fig, (ax1, ax2, ax3) = plt.subplots(1,3)
ax1.plot(w1[1,:,0])
```

```
ax1.plot(w1[1,:,1])
ax1.plot(w1[1,:,2])
ax1.set_title('Coefficients')
ax1.set_xlabel('updates')
ax1.set_ylabel('weights')

ax2.plot(w_avg0)
ax2.plot(w_avg1)
ax2.plot(w_avg2)
ax2.set_title('Coefficients(averaged)')
ax2.set_xlabel('updates')
ax2.set_ylabel('averaged weights')

ax3.plot(mse)
ax3.set_title('Learning Curve for 3dB, n = 0.15')
ax3.set_xlabel('Updates')
ax3.set_ylabel('MSE')



#Question 3b
tv_c = np.asarray(data['timevarying_coefficents'])
tv_x = np.asarray(data['timevarying_x'])
tv_y = np.asarray(data['timevarying_y'])
tv_z = np.asarray(data['timevarying_z'])
tv_v = np.asarray(data['timevarying_v'])

wn = tv_c[0]
y = np.zeros((501))
mse = np.zeros((600,501))

tv = []

for j in range(501):
  y[j] = np.matmul(wn.T,tv_v[j])
  wn = wn + 0.08*(tv_z[j]-y[j])*tv_v[j]
  tv.append(wn)
tv = np.array(tv)

plt.rcParams['figure.figsize'] = [5,40]

fig, (ax1, ax2, ax3, ax4, ax5, ax6) = plt.subplots(6,1)
ax1.plot(n,tv_c[:,0])
ax2.plot(n,tv_c[:,1])
ax3.plot(n,tv_c[:,2])

ax1.set_title('True Time Varying Coefficient-1')
ax1.set_xlabel('Time n')
ax1.set_ylabel('True Time Varying Coefficient-1')

ax2.set_title('True Time Varying Coefficient-2')
ax2.set_xlabel('Time n')
```

```
ax2.set_ylabel('True Time Varying Coefficient-2')

ax3.set_title('True Time Varying Coefficient-3')
ax3.set_xlabel('Time n')
ax3.set_ylabel('True Time Varying Coefficient-3')


ax4.plot(n,tv[:,0])
ax5.plot(n,tv[:,1])
ax6.plot(n,tv[:,2])

ax4.set_title('Estimated Time Varying Coefficient-1')
ax4.set_xlabel('Time n')
ax4.set_ylabel('Estimated Time Varying Coefficient-1')

ax5.set_title('Estimated Time Varying Coefficient-2')
ax5.set_xlabel('Time n')
ax5.set_ylabel('Estimated Time Varying Coefficient-2')

ax6.set_title('Estimated Time Varying Coefficient-3')
ax6.set_xlabel('Time n')
ax6.set_ylabel('Estimated Time Varying Coefficient-3')

plt.rcParams['figure.figsize'] = [5,18]

fig, (ax1, ax2, ax3) = plt.subplots(3,1)

ax1.plot(tv_c[:,0], Label = 'True Coefficients')
ax1.plot(tv[:,0], Label = 'Estimated Coefficients')
ax1.set_title('True Time Varying Coefficient-1 vs Estimated Coefficient-1')
ax1.set_xlabel('Updates')
ax1.set_ylabel('Coefficients')

ax2.plot(tv_c[:,1], Label = 'True Coefficients')
ax2.plot(tv[:,1], Label = 'Estimated Coefficients')
ax2.set_title('True Time Varying Coefficient-2 vs Estimated Coefficient-2')
ax2.set_xlabel('Updates')
ax2.set_ylabel('Coefficients')


ax3.plot(tv_c[:,2], Label = 'True Coefficients')
ax3.plot(tv[:,2], Label = 'Estimated Coefficients')
ax3.set_title('True Time Varying Coefficient-3 vs Estimated Coefficient-3')
ax3.set_xlabel('Updates')
ax3.set_ylabel('Coefficients')



#Question 3c

x_mm = np.asarray(data['mismatched_x'])
y_mm = np.asarray(data['mismatched_y'])
```

```
v_mm = np.asarray(data['mismatched_v'])
y_mm = np.asarray(data['mismatched_y'])

#For n = 0.01
wn = np.array(wn)
y = np.zeros((600,501))
sum = np.zeros((3,1))
sum1 = 0
mse = np.zeros((600,501))

vx = v_mm.T
y2 = np.zeros((600,501))

for i in range(600):
  wn = np.array([0, 0, 0])

  for j in range(501):
    y[i][j] = np.matmul(wn.T,v_mm[i][j])
    wn = wn + 0.01*(y_mm[i][j]-y[i][j])*v_mm[i][j]
  Rv = (1/600)*(np.matmul(v_mm[i].T,v_mm[i]))
  rvy = (1/600)*(np.matmul(v_mm[i].T,y[i]))
  w = np.matmul(rvy, inv(Rv))
  y2[i] = np.matmul(v_mm[i],w)

mse = np.sum((y_mm - y)**2, axis = 0)/600

plt.plot(mse)
plt.xlabel('Updates')
plt.ylabel('MSE')
plt.title('Average Learning Rate for n = 0.01')
wllse = np.sum((y_mm - y2)**2, axis = 0)/600
plt.plot(wllse)
plt.legend(['MSE', 'LLSE'])
print(wllse)
print(Rv)
print(rvy)

#For n = 0.04

mse = np.zeros((600,501))
vx = v_mm.T
y2 = np.zeros((600,501))

for i in range(600):
  wn = np.array([0, 0, 0])
  for j in range(501):
    y[i][j] = np.matmul(wn.T,v_mm[i][j])
    wn = wn + 0.04*(y_mm[i][j]-y[i][j])*v_mm[i][j]
  Rv = (1/600)*(np.matmul(v_mm[i].T,v_mm[i]))
  rvy = (1/600)*(np.matmul(v_mm[i].T,y[i]))
  w = np.matmul(rvy, inv(Rv))
  y2[i] = np.matmul(v_mm[i],w)
```

```
print(Rv)
print(rvy)

mse = np.sum((y_mm - y)**2, axis = 0)/600
wllse = np.sum((y_mm - y2)**2, axis = 0)/600
print(wllse)

plt.plot(mse)
plt.xlabel('Updates')
plt.ylabel('MSE')
plt.title('Average Learning Rate for n = 0.04')
plt.plot(wllse)
plt.legend(['MSE', 'LLSE'])

#For n = 0.05

mse = np.zeros((600,501))
vx = v_mm.T
y2 = np.zeros((600,501))

for i in range(600):
  wn = np.array([0, 0, 0])
  for j in range(501):
    y[i][j] = np.matmul(wn.T,v_mm[i][j])
    wn = wn + 0.05*(y_mm[i][j]-y[i][j])*v_mm[i][j]
  Rv = (1/600)*(np.matmul(v_mm[i].T,v_mm[i]))
  rvy = (1/600)*(np.matmul(v_mm[i].T,y[i]))
  w = np.matmul(rvy, inv(Rv))
  y2[i] = np.matmul(v_mm[i],w)

print(Rv)
print(rvy)

mse = np.sum((y_mm - y)**2, axis = 0)/600
wllse = np.sum((y_mm - y2)**2, axis = 0)/600
print(wllse)

plt.plot(mse)
plt.xlabel('Updates')
plt.ylabel('MSE')
plt.title('Average Learning Rate for n = 0.05')
plt.plot(wllse)
plt.legend(['MSE', 'LLSE'])

#For n = 0.07

mse = np.zeros((600,501))
vx = v_mm.T
y2 = np.zeros((600,501))

for i in range(600):
```

```
  for i in range(600):
    wn = np.array([0, 0, 0])
    for j in range(501):
      y[i][j] = np.matmul(wn.T,v_mm[i][j])
      wn = wn + 0.07*(y_mm[i][j]-y[i][j])*v_mm[i][j]
    Rv = (1/600)*(np.matmul(v_mm[i].T,v_mm[i]))
    rvy = (1/600)*(np.matmul(v_mm[i].T,y[i]))
    w = np.matmul(rvy, inv(Rv))
    y2[i] = np.matmul(v_mm[i],w)

print(Rv)
print(rvy)

mse = np.sum((y_mm - y)**2, axis = 0)/600
wllse = np.sum((y_mm - y2)**2, axis = 0)/600
print(wllse)

plt.plot(mse)
plt.xlabel('Updates')
plt.ylabel('MSE')
plt.title('Average Learning Rate for n = 0.07')
plt.plot(wllse)
plt.legend(['MSE', 'LLSE'])

#For n = 0.15

mse = np.zeros((600,501))
vx = v_mm.T
y2 = np.zeros((600,501))

for i in range(600):
  wn = np.array([0, 0, 0])
  for j in range(501):
    y[i][j] = np.matmul(wn.T,v_mm[i][j])
    wn = wn + 0.15*(y_mm[i][j]-y[i][j])*v_mm[i][j]
  Rv = (1/600)*(np.matmul(v_mm[i].T,v_mm[i]))
  rvy = (1/600)*(np.matmul(v_mm[i].T,y[i]))
  w = np.matmul(rvy, inv(Rv))
  y2[i] = np.matmul(v_mm[i],w)

print(Rv)
print(rvy)

mse = np.sum((y_mm - y)**2, axis = 0)/600
wllse = np.sum((y_mm - y2)**2, axis = 0)/600

print(wllse)

plt.plot(mse)
plt.xlabel('Updates')
plt.ylabel('MSE')
plt.title('Average Learning Rate for n = 0.15')
```

```
    plt.plot(wllse)
    plt.legend(['MSE', 'LLSE'])
```