Homework 3

Question 1:

1.1

In this question, I test range of filter order from 2 to 30 and range of regularization parameters from 0 to 0.5. From 0 to 0.5 as a regularization parameter, I choose parameters by step of 0.01. After use training data set for a weight vector, I calculate the mean square error for each parameter of filter order and regularization parameters. The 3-d plot figure is shown in Figure 1.

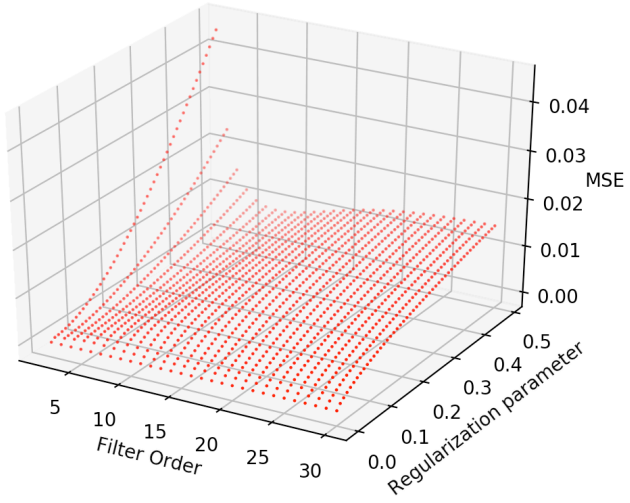
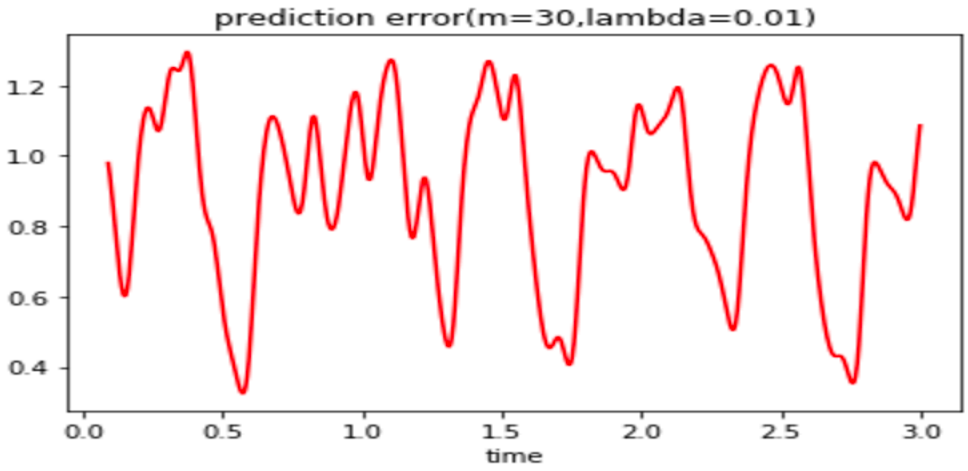
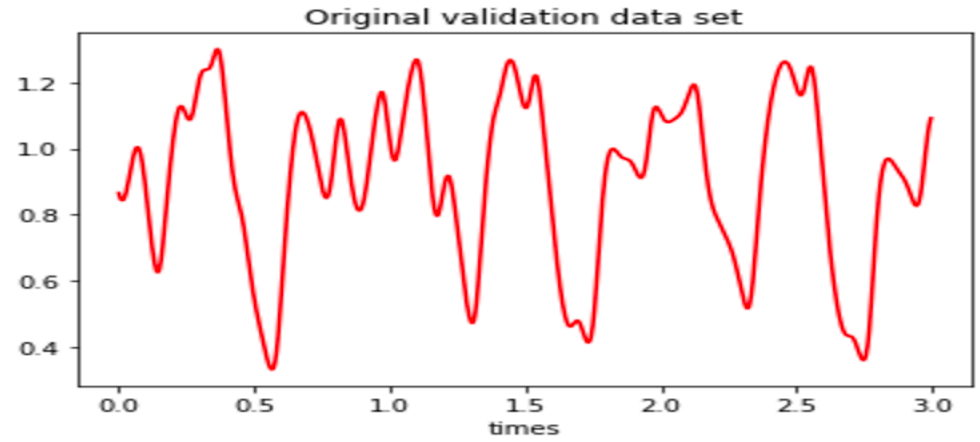


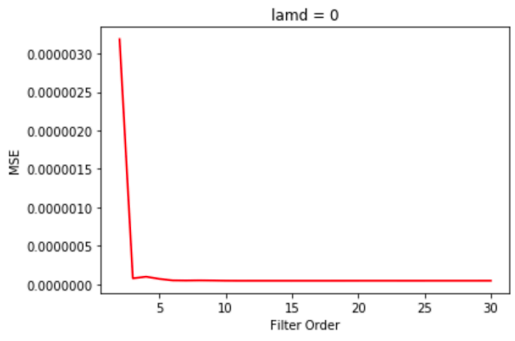
Figure 1

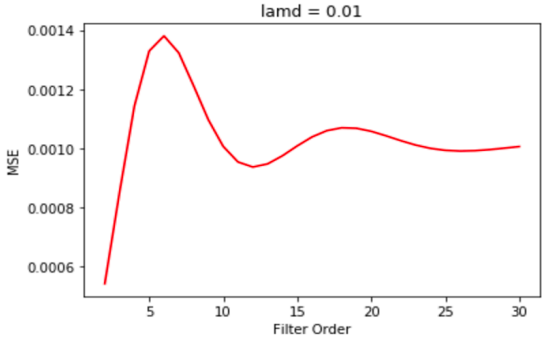
As we can infer from Figure 1, for each value of filter order, the value of MSE is going up along with the value of regularization parameter increasing. MSE is minimized when filter order equal to 30 and regularization parameter equal to 0.



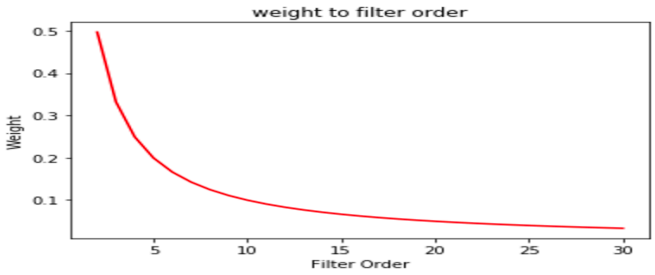


As we can see from the figures above, when filter order equal to 30 and regularization parameter equal to 0, the predict data set are almost the same as the original data set.

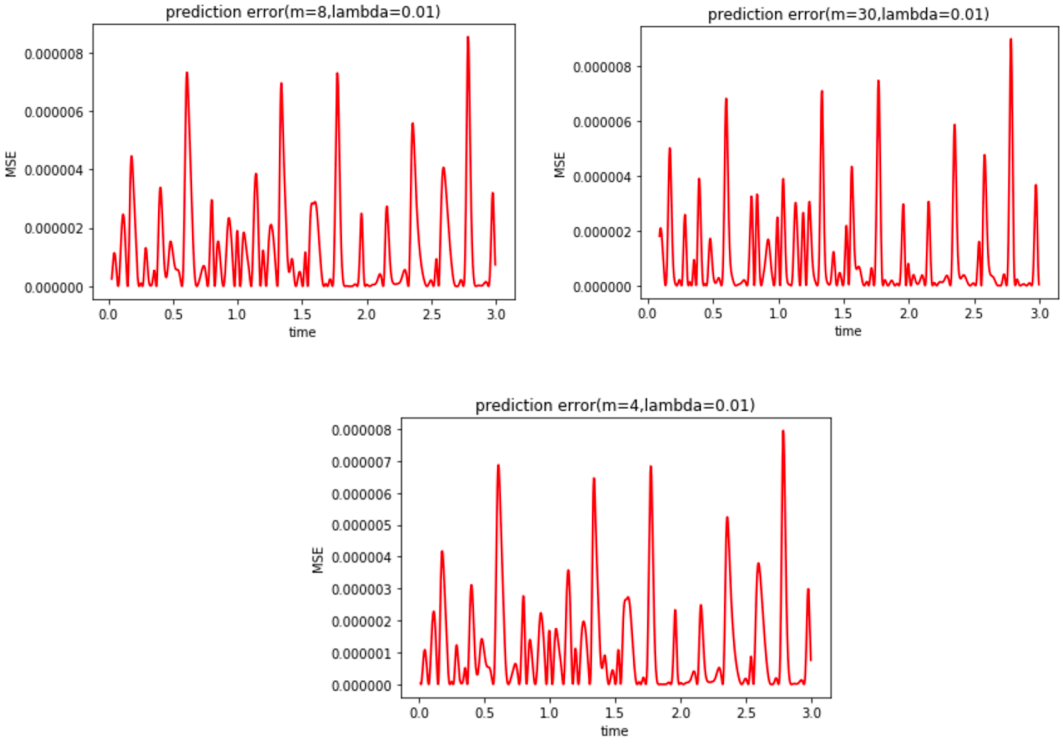




When lambda = 0, the MSE is near to zero.



weight changing with Filter order increase



1.2

When filter order M = {4,8,30} MSE value = {11.1842, 10.1933, 106.8979}. So, among the 3 filter orders, M=4 can provide better performance for noise data. Because the value of MSE is the smallest.

Question 2

2.1

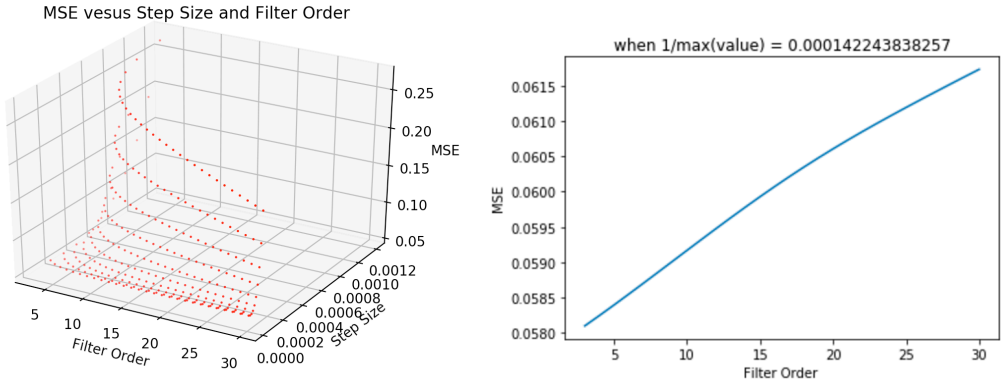
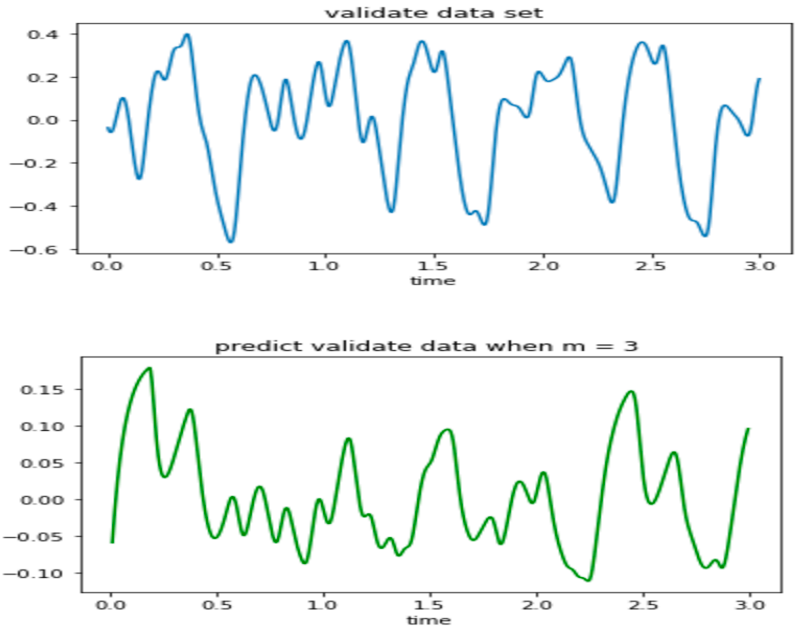


Figure 2. figure(left) is a 3-d plot of filter order, step size and MSE. Figure on the right shows the smallest MSE situation.

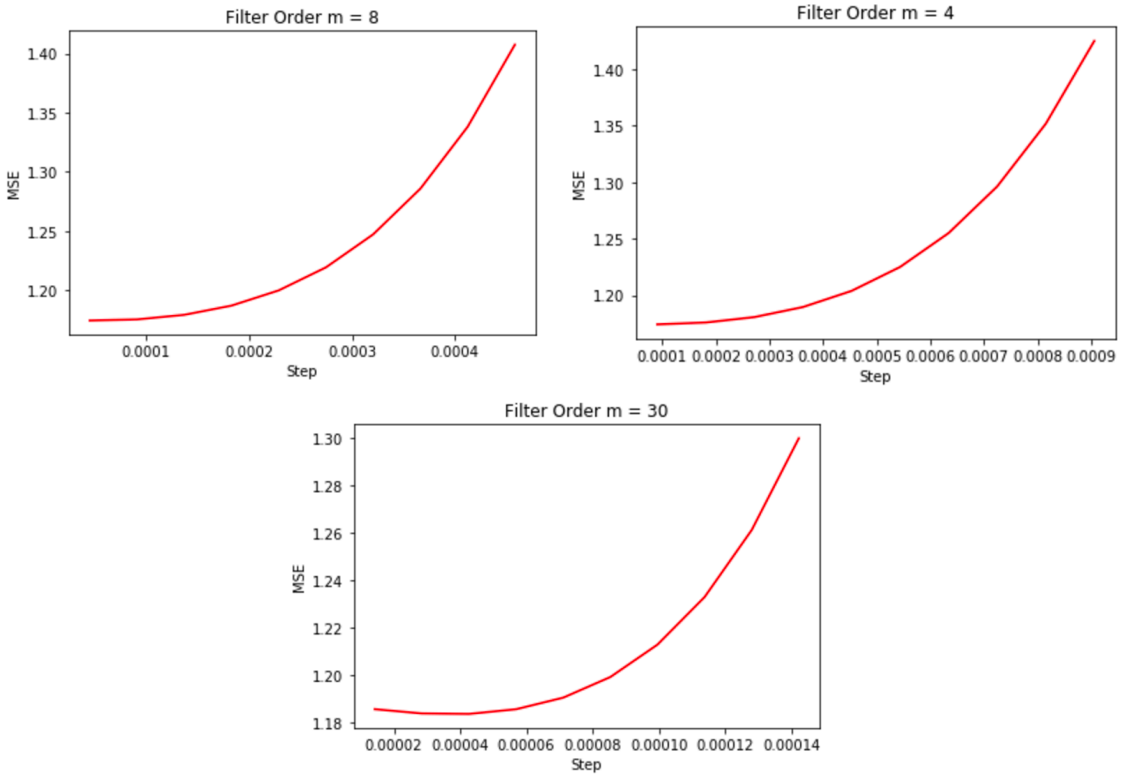
As we can infer from the 3-D picture and Order-MSE figure, when we chose filter order of 3, the mean square error is the smallest. Filter shows best performance when order equal to 3.

When we use order of 3, we can compare the predict figure with original data set.



As we can see from the two figures above the difference between the two figures is huge.

For noise data set:



Filter Order	4	8	30
Min(MSE)	1.17410313637	1.17447123695	1.1836350132

So, when the filter order equal to 4, the MSE is the smallest. So, use this value of filter order, we can get a better performance.

2.2

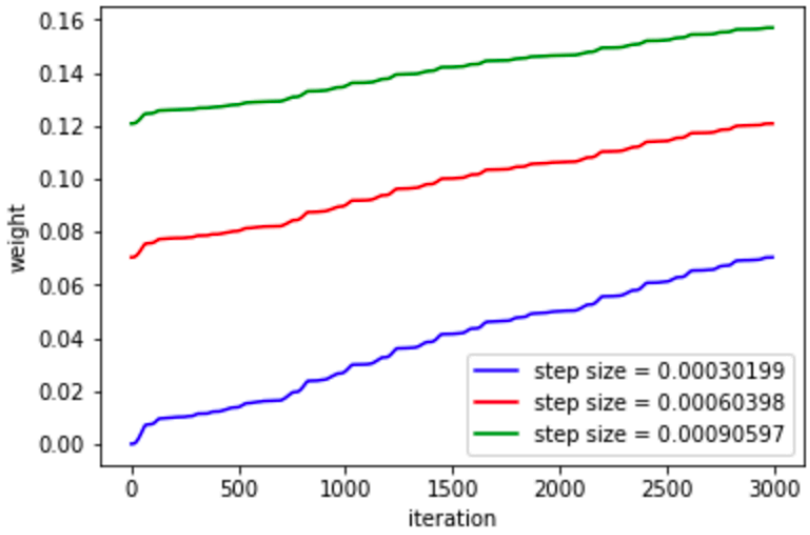


Figure 3. Plot weight with different step size.

As we can see from the figure, along with the increase of step size, the weight value is increased.

If the step size is too large, the value of weight and value of J are going to increase by iteration. If the step size is too small, the slope of learning curve is going to be very low and the weight value might be trained well even after 3000 iterations. So, we should maintain the step size smaller than the 1/max(λ) $which\ \lambda i$ is the eigenvalue of X*X.T and step size should also be larger than 0.

Code for Question 1:

```
import numpy as np
import scipy.io as sio
from mpl_toolkits.mplot3d import Axes3D
import matplotlib
import matplotlib.pyplot as plt
import math
import array
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
%matplotlib inline
#load data from file
train = np.loadtxt('training.txt')
validate = np.loadtxt('validate.txt')
x_t = np.arange(0,3,0.001)
x_v = np.arange(0,3,0.003)
N = train.shape[0]
Nv = validate.shape[0]
#function to return the EMR of this method
# training data adn validation
lamd = np.arange(0,0.5,0.01)
MSE_v = np.zeros(shape=(50,32))
x = []
y = []
z = []
y_v = []
MSE_1 = []
for m in range(2,31):
    for l in range(50):
        X = np.zeros(shape=(m,N-m))
```

```
            Y = train[m:N]
            for i in range(0,N-m):
                temp = train[i:i+m]
                X[:,i] = temp[::-1]
            r = (X@X.T)/(N-m)
            R = r+lamd[l]*np.identity(m)
            P = (X@Y)/(N-m)
            W = np.linalg.inv(R)@P
            Xv = np.zeros(shape=(m,Nv-m))
            Yv = validate[m:Nv]
            for j in range(0,Nv-m):
                temp = validate[j:j+m]
                Xv[:,j] = temp[::-1]
            yv = W.T@Xv
            MSE = np.zeros(shape=(Nv-m,1))
            for n in range(0,Nv-m):
                MSE[n] = (Yv[n] - yv[n])**2
            MSE_v[l,m] = MSE.sum()/(Nv-m)
            x.append(m)
            y.append(lamd[l])
            z.append(MSE.sum()/(Nv-m))
            if lamd[l]==0.01:
                MSE_1.append(MSE.sum()/(Nv-m))

ax = plt.subplot(111,projection='3d')
ax.scatter(x,y,z,c='r',s=0.5)
plt.show()
```

Code for question 2

```
import numpy as np
import scipy.io as sio
from mpl_toolkits.mplot3d import Axes3D
import matplotlib
import matplotlib.pyplot as plt
import math
import array
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
%matplotlib inline
train = np.loadtxt('training.txt')
validate = np.loadtxt('validate.txt')
x_t = np.arange(0,3,0.001)
x_v = np.arange(0,3,0.003)
train = train-np.mean(train)
validate = validate - np.mean(validate)
N = train.shape[0]
Nv = validate.shape[0]
x = []
y = []
z = []
y_v = []
j = []
plt.plot(x_v,validate)
plt.xlabel('time')
plt.title('validate data set')
for m in range(3,31):
    X = np.zeros(shape=(m,N-m))
    Y = train[m:N]    # Y is N-m*1 matrix
```

```
        #print(Y.shape)
        for i in range(0,N-m):
            temp = train[i:i+m]
            X[:,i] = temp[::-1] # X is m*N-m matrix
        R = X@X.T
        value, vector = np.linalg.eig(R)
        step_up = 1/max(value)
        step = np.arange(step_up/10,step_up+step_up/11,step_up/10)
        for j in range(10):
            sum_rms = 0
            w = np.zeros(shape=(m,1)) # w is m*1 matrix
            for iteration in range(0,N-m):
                #w = w + step[j]*(X[:,iteration].T@w-Y[iteration])*X[:,iteration]
                error = X[:,iteration].T@w-Y[iteration]
                J = step[j]*error*X[:,iteration]
                w[:,0] = w[:,0] + J.T
            #print(w)
            Xv = np.zeros(shape=(m,Nv-m))
            Yv = validate[m:Nv]
            for p in range(0,Nv-m):
                temp = train[p:p+m]
                Xv[:,p] = temp[::-1]
            yv = w.T@Xv
            MSE = np.zeros(shape=(Nv-m,1))
            for n in range(0,Nv-m):
                MSE[n] = (Yv[n] - yv.T[n])**2
            x.append(m)
            y.append(step[j])
            z.append(MSE.sum()/(Nv-m))
ax = plt.subplot(111,projection='3d')
```

```
ax.scatter(x,y,z,c='r',s=0.5)
ax.set_title('MSE vesus Step Size and Filter Order')
ax.set_xlabel('Filter Order')
ax.set_ylabel('Step Size')
ax.set_zlabel('MSE')
plt.show()
```