PROBLEM 1

PART A: Load the data from "curve80.txt" and split it into 75% / 25% training and test ratio.

In order to do that, I used the following code to load data and split into training and test data set. As 75% of the data are set to be training data, the first 60 rows of data are considered as training data. X of the data set is locate in the first column; therefore xTrain can be found by "Xtr = data_tr(:,1)" and xTest can be found by "Xte = data_te(:,1)". Similarly for yTrain and yTest where Y value of the data set locate in the second column.

```
load 'data/curve80.txt'

% PART A: (Split the data into 75% for training, 25% for testing

data_tr = curve80(1:60,:);
data_te = curve80(61:80,:);
Xtr = data_tr(:,1);
Xte = data_te(:,1);
Ytr = data_tr(:,2);
Yte = data_te(:,2);
```

PART B: Use the provided linearRegress class to create a linear regression predictor of y given x. You can plot the resulting function by simply evaluating the model at a large number of x values, xs

The following code are written and it serves as degree one Linear Regression

```
lr = linearRegress(Xtr,Ytr); % Create and train model
xs = [0: .05: 10]'; % Densely sample possible x-values: note transpose
ys = predict( lr, xs);

% Plot the training data with predition function in a single plot
figure (1)
hold on;
scatter(Xtr,Ytr,'r')
plot(xs,ys)
hold off;

% Calculate and report MSE
MSE_te = mse(lr,Xte,Yte)   %MSE on Test Data
MSE_tr = mse(lr,Xtr,Ytr)   %MSE on Train Data
```

Learner is defined by " lr = linearRegress(Xtr, Ytr)", it is trained with training data. Xs is an sampling array, it defines at which x values we are taking samples from the prediction function: ls. Please see the following figure for the plot of the learner.

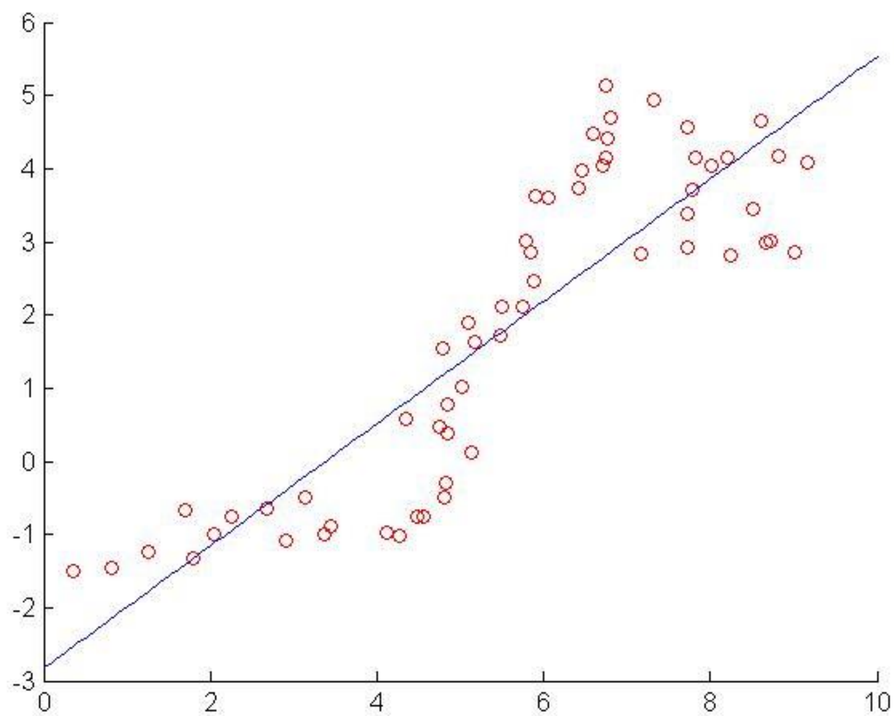

Figure 1

MSE value for testing and training data are calculated as follow.

MSE_te =2.2423

MSE_tr =1.1277

PART C: Fitting y = f(x) using a polynomial function f(x) of increasing order.

In order to fit y with a higher order polynomial function, the training data xTrain need to be transformed and rescaled with the given function: "fpoly" and "rescale". Then, linear regression learner is constructed with "linearRegress" function with the transformed training feature data (XtrP) and training result data (Ytr). As we are considering at a higher order, xs also needed to be transformed and rescaled in order to provide a valid projection from the learner to the estimated value. Then the MSE is constructed with learner(i.e. ls_P), transformed training or testing data (i.e. XtrP or XteP) and training or testing Y value (i.e: Ytr or Yte).

```
degree = 3;
XtrP = fpoly(Xtr,degree,false); %create poly features up to given degree; no
"1" feature
[XtrP, M,S] = rescale(XtrP); % Rescale the features
lr_P = linearRegress( XtrP,Ytr); %Create and Train model
```

```
XteP = rescale( fpoly(Xte,degree,false), M,S); %Apply the same polynomial
expansion & scaling to Xtest
ys_P_3 = predict( lr_P,rescale( fpoly(xs,degree,false), M,S));
MSE_te_3 = mse(lr_P,XteP,Yte)    %MSE on Test Data
MSE_tr_3 = mse(lr_P,XtrP,Ytr)    %MSE on Train Data
figure (3)
hold on;
scatter(Xtr,Ytr,'r')
plot(xs,ys_P_3)
hold off;
```

Different orders of polynomial are considered. Set d to be the degree, then d = [1 3 5 7 10 18]. Resultant plot of the learner can be found in the following figures.

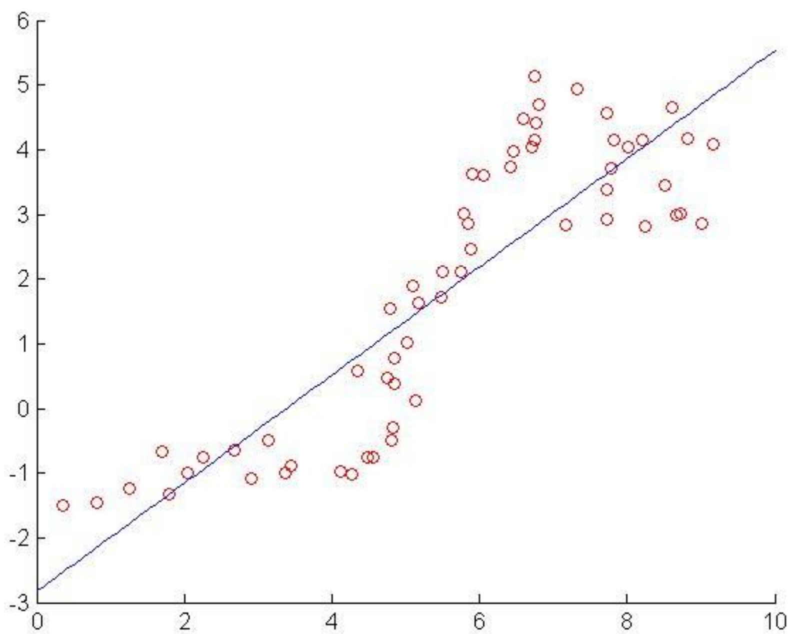

Figure 2 (Degree = 1)

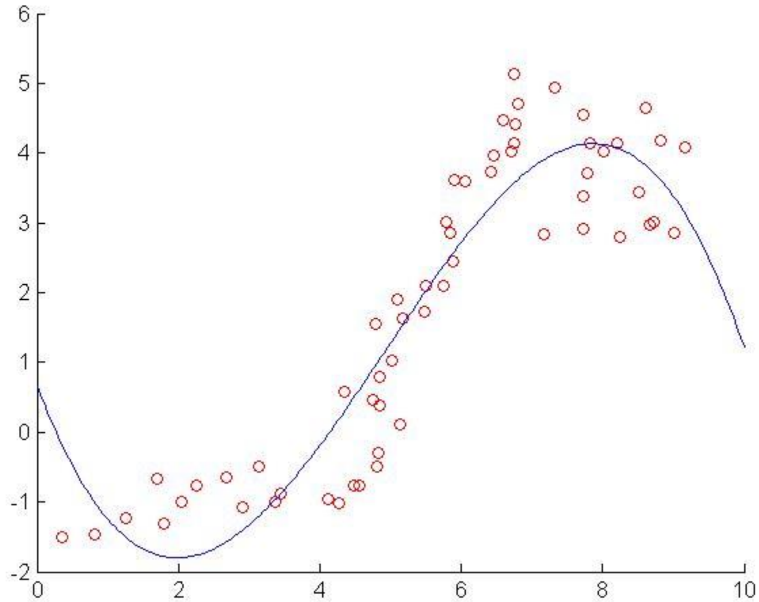

Figure 3 (Degree = 3)

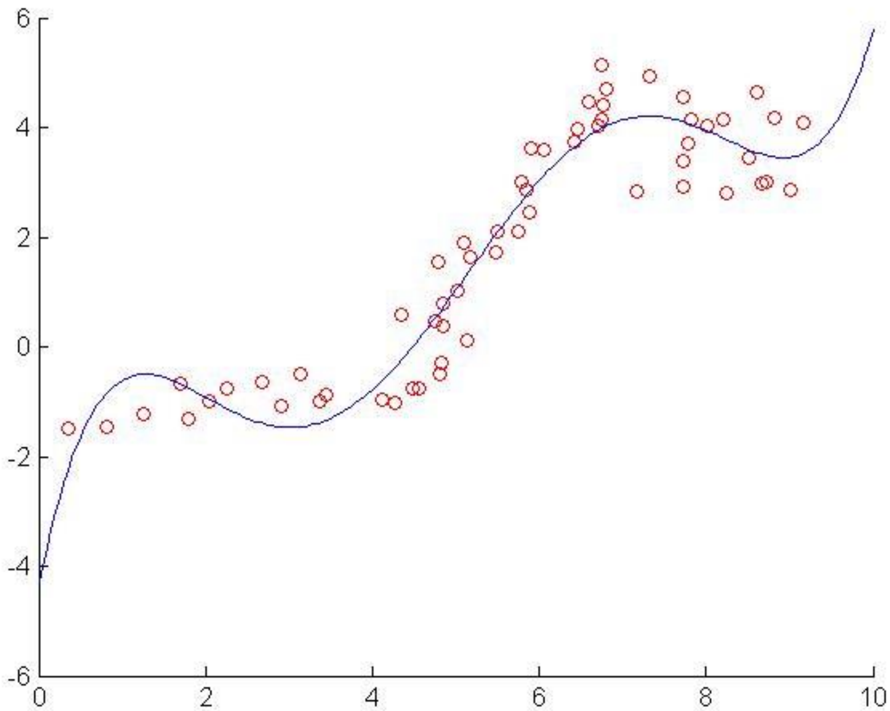

Figure 4 (Degree = 5)

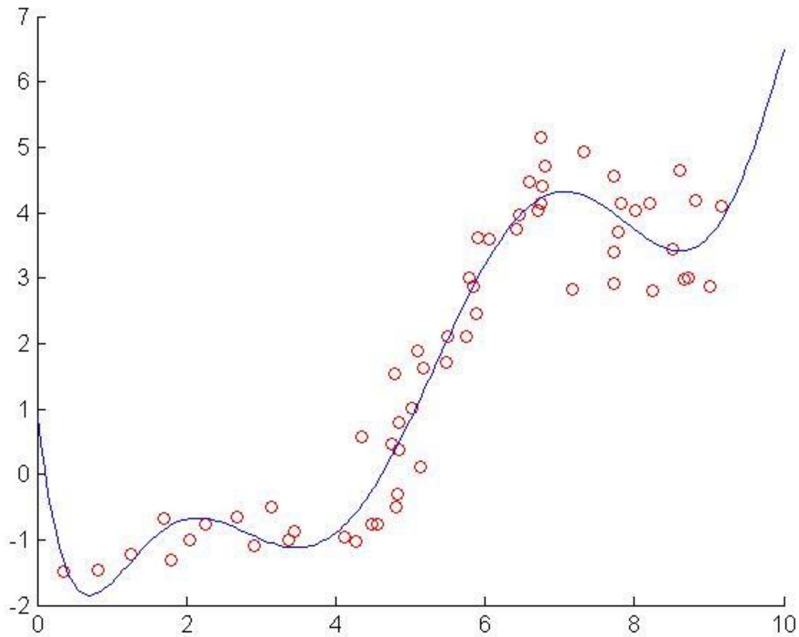

Figure 5 (Degree = 7)

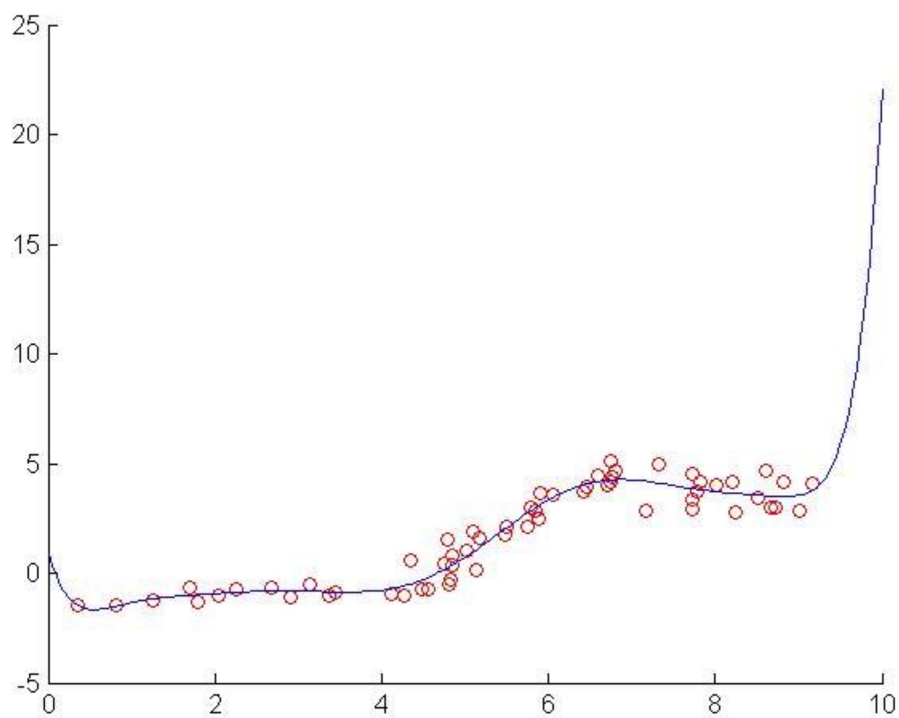

Figure 6 (Degree = 10)

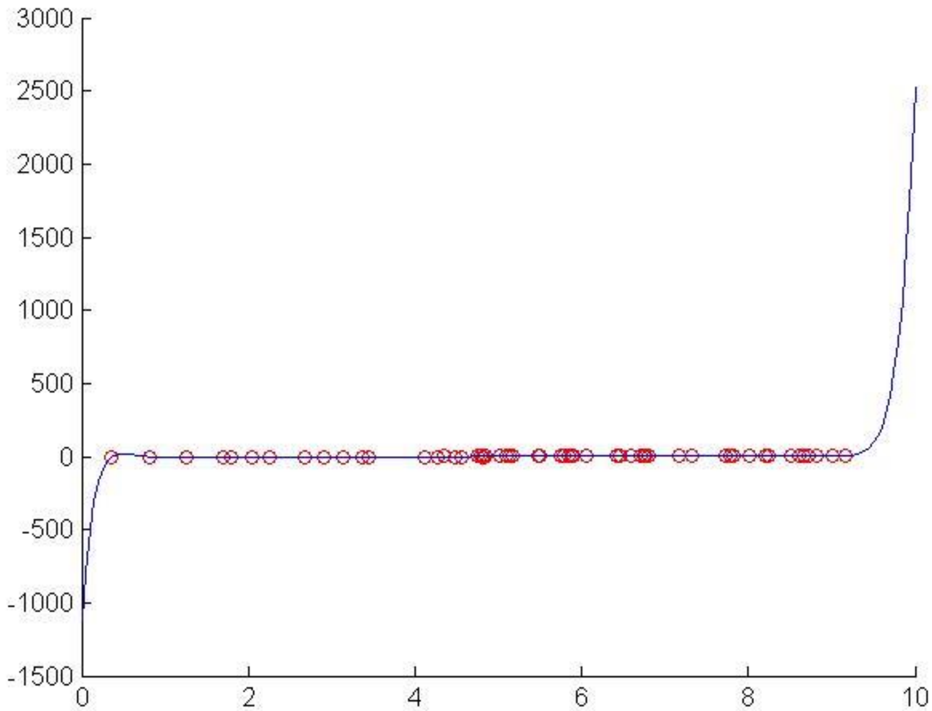

Figure 7 (Degree = 18)

MSE Testing Data Array:

2.24234920301013 0.861611481545005 1.03441902056335 0.650224607965615 0.609060074949968 482.646324168260

The first element refers to MSE for prediction function with degree 1. The second one refer to degree 3. Same order follows.

MSE Training Data Array

1.12771195560939 0.633965206311964 0.404248946445901 0.315634673989417 0.298947979673854 0.280486093901787

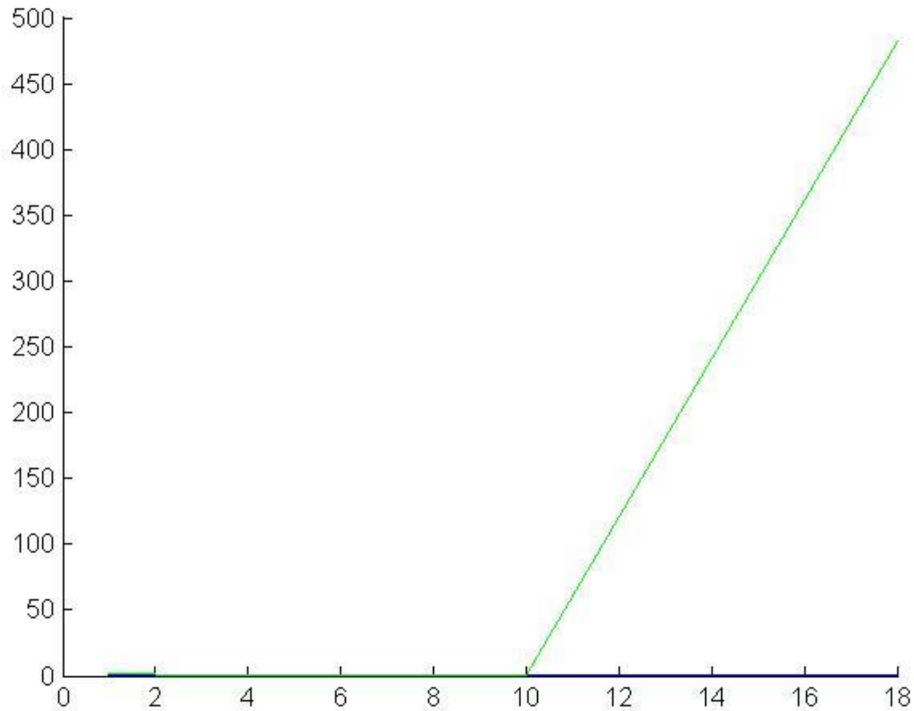

Figure 8 (MSE Plot)

PROBLEM 2

Cross-validation can be used in deciding the degree of polynomial fit. "crossValidate" split data into n-fold and group into testing and training data. Then the training feature (i.e. Xti) is transformed and rescaled for higher order approximation. Same applies on testing data. Learner is constructed by transformed training features and training value (i.e. Xti) and Yti). Lastly, MSE can be found with the "mse" function.

```
nFolds = 5;
for iFold = 1:nFolds,
    % Xti = xTrain for cross-validation
    % Xvi = xTest for cross-validation
    % Yti = yTrain for cross-validation
    % Yvi = yTest for cross-validation

    % STEP 1: Split data for n-fold cross validation with Xtr in problem 1
    [Xti, Xvi, Yti, Yvi] = crossValidate(Xtr,Ytr,nFolds,iFold);  %Take ith
data block as validation

    % Consider degree = 1
    degree = 1;
    XtiP_1 = fpoly(Xti,degree,false); %create poly features up to given
degree; no "1" feature
    [XtiP_1, M,S] = rescale(XtiP_1); % Rescale the features
```

```
    XviP_1 = fpoly(Xvi,degree,false); %create poly features up to given
degree; no "1" feature
    [XviP_1, M,S] = rescale(XviP_1); % Rescale the features

    % LinearRegress intake train data: Xti; Yti
    learner_1 = linearRegress(XtiP_1,Yti); % ToDo: Train on Xti, Yti, the
data for this fold
    J_1(iFold) = mse(learner_1,XviP_1,Yvi)% ToDo: Now compute the MSE on Xvi,
Yvi, and save it

end;
% The overall estimated validation performance is the average of the
% performance on each fold
J_mean_1 = mean(J_1)
```

A semilogy plot is constructed with an MSE array

MSE array is named as "J_mean_array". It carries the following value corresponding to different values of order.

1.39642483804461	1.18099516156796	1.28113141294213
	1.18924833124360	1.36187722764026
	376286.376115307	

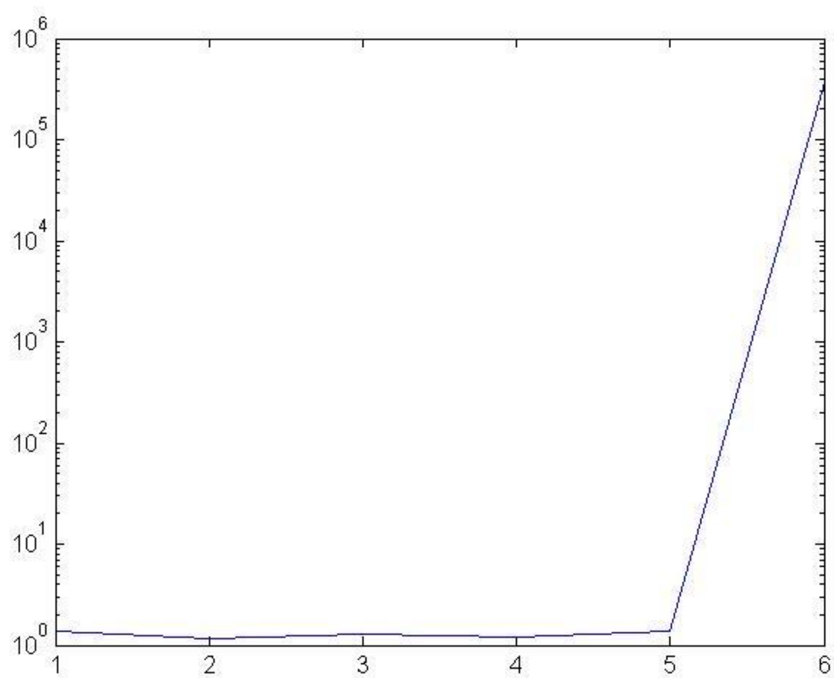

Figure 9 (Cross Validation MSE)

According to the MSE array (i.e. J_mean_array), degree = 3 gives the smallest cross validation MSE. Compare to both testing and training MSE array of problem 1, Cross validation gives a smaller MSE reading than the MSE observed from actual data.