**Machine Learning Programming Assignment 1**

**Comp540 Spring 2015 due 23 January 2015 at 8 pm**

**Team member : Haoyue Zhang Hz32**

**Miao Wang MW56**

**Problem 1: Linear regression with one variable**

Frist Plotting the data then training data , then visualized data

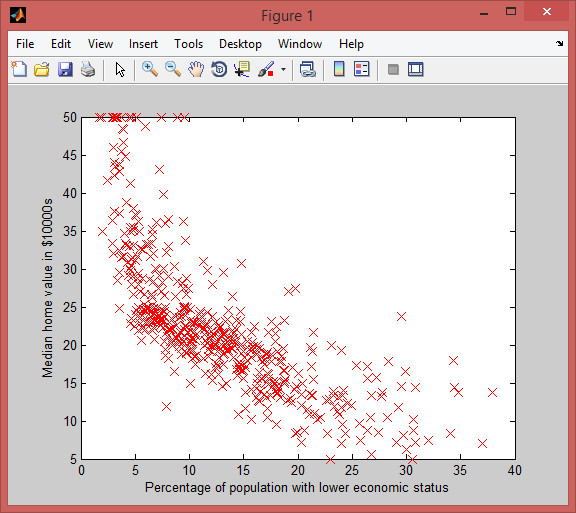
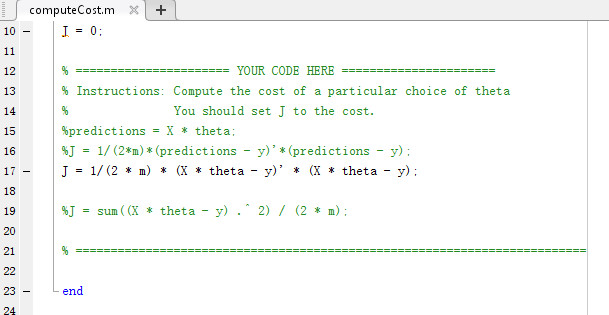


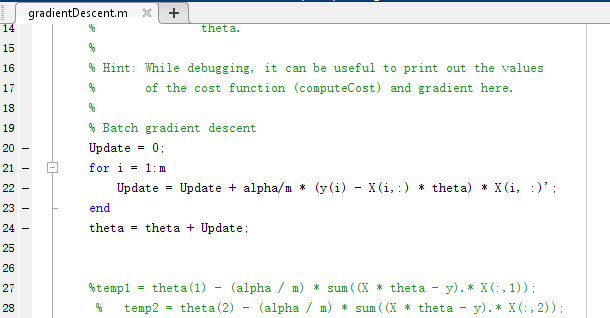
Figure 1: Scatter plot of training data

You then need to implement two functions” computeCost” and “graientDescent”, were calculated in accordance with the parameters of the cost function and gradient direction is updated, combined with the cost function Linear Regression formulas and parameters update Rule, we can achieve the following.

**Computing the cost function** *J* (*θ*)



**Implementing gradient descent**



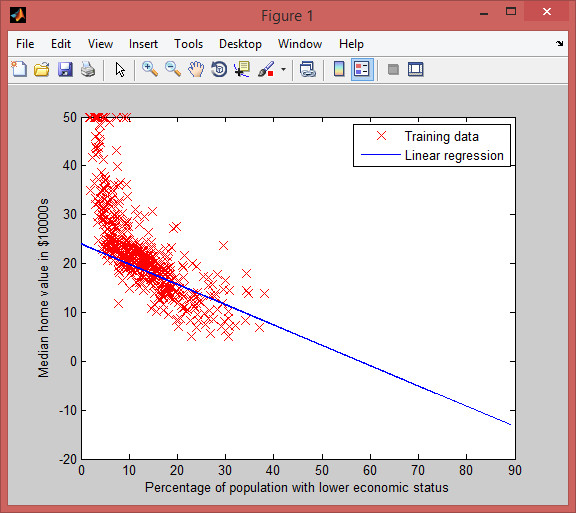
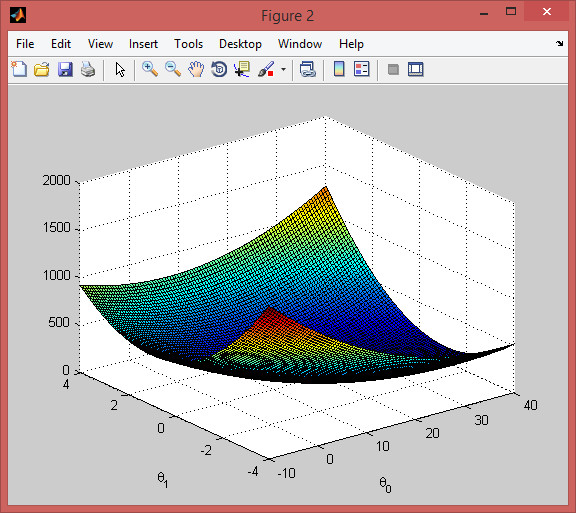
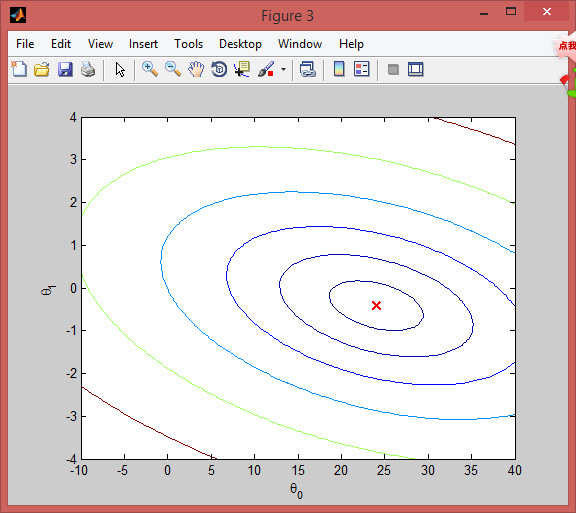


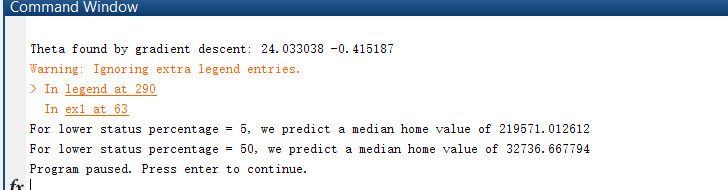
Figure 2: Fitting a linear model to the data in Figure

**Visualizing J (θ)** of the main part of the four methods by surf and contour methods were drawn theta1 cost function as circumstances change and the contours and theta2





**Making predictions on unseen data**

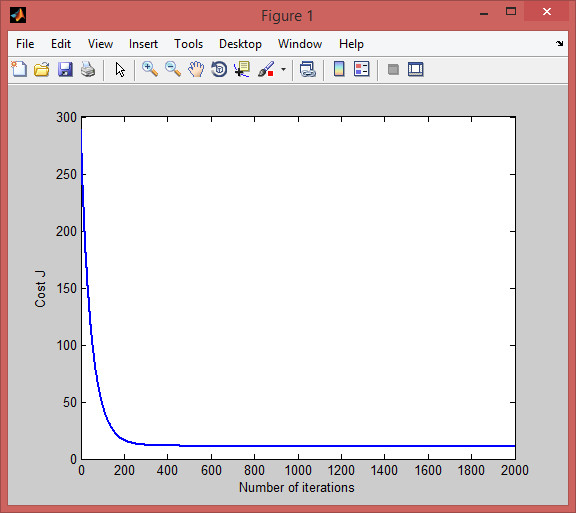
****

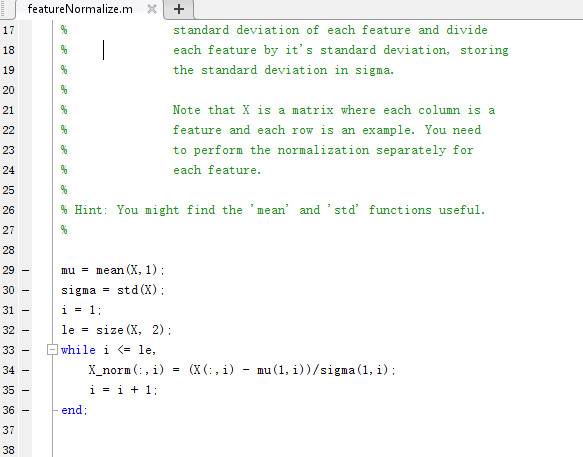
**Problem 2: Linear regression with multiple variables**

If each training sample to describe this problem is Linear Regression multivariate multiple feature. For example, according to the area and the bedroom we want to predict the number of house price of the house, so now each training sample is used to describe two feature.

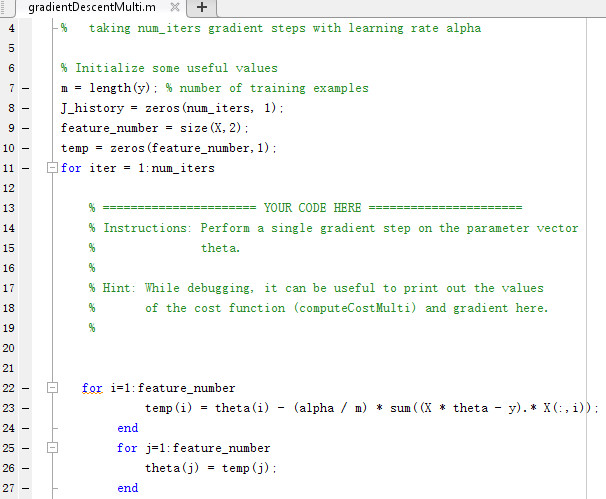
**Feature normalization**

By observing the characteristic feature may know, the value of the house area is about 1,000 times the number of values in the bedroom, when faced with different numerical range feature a very significant difference, the need to make feature normalization, which can speed up the learning algorithm convergence. To Feature Normalization, need first to calculate the mean value for each column feature \ mu and standard deviation \ sigma, then normalization / feature value x 'original feature value x satisfies x' scale after = (x - \ mu) / \ sigma . The original feature that is subtracted from the mean and standard deviation divided. Thus we can achieve such feature normalization function

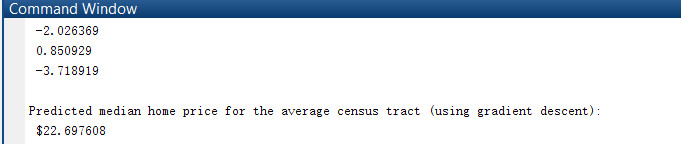




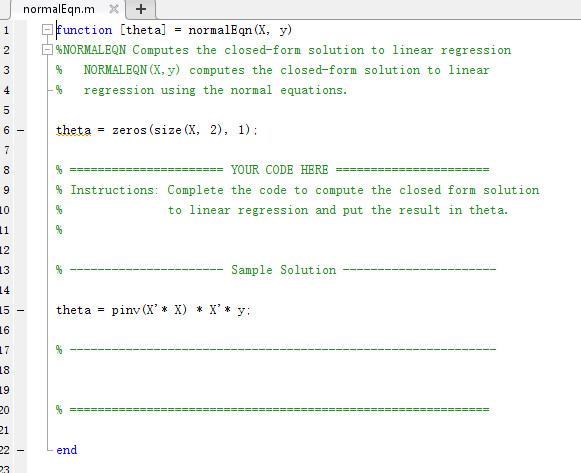
**Gradient descent multi**

****

**Making predictions on unseen data (5 points)**

****

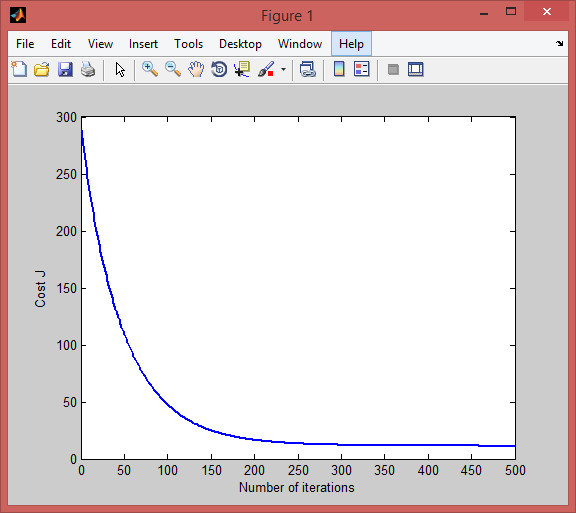
**Normal equations**

****

**Exploring convergence of gradient descent**

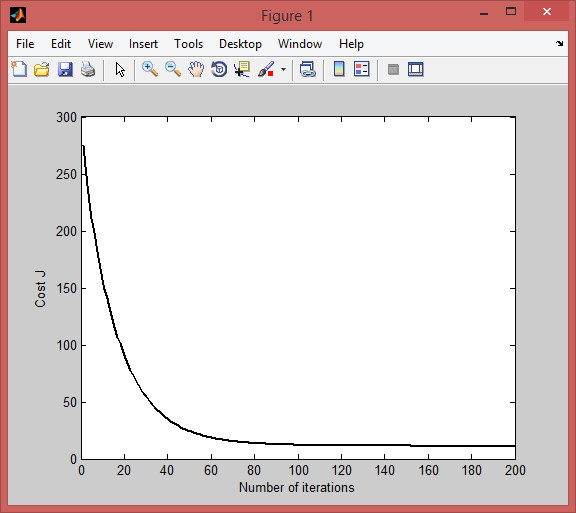
Learning rate set here \ alpha = 0.01, iteration 2000, can be seen at about 200 times the cost function J is almost convergence, not change. We can also adjust the learning rate \ alpha, select the appropriate learning rate is important, the election is too small convergence is very slow, the election may not be too much convergence (each iteration parameters changed so much, cannot find the extreme points) as the i.e. suggest to select \ alpha when follow log scale, such as constantly dividing 3,0.3, 0.1, 0.03, 0.01

Rate 0.01



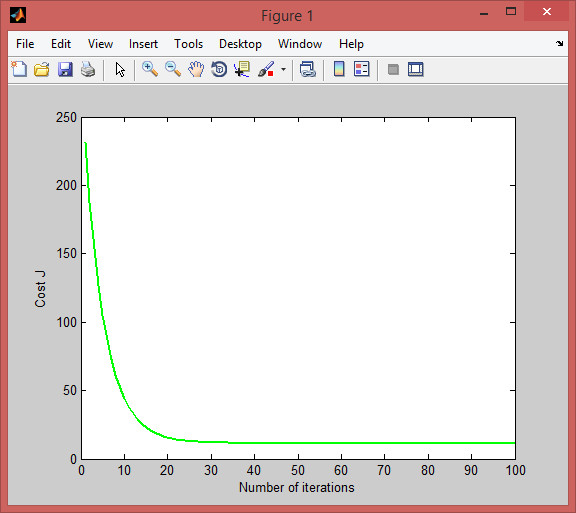
Cost J 300 iteration

Rate 0.03



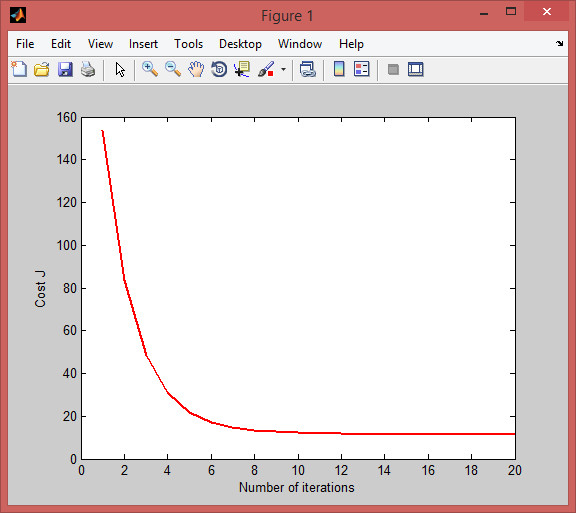
Cost J 200 iteration

Rate 0.1



Cost J 100 iteration

Rate 0.3



Cost J 20 iteration

Because of



Which is a little different with the learning rate =0.01, May be I will select 0.3.



**Part 2: Implementing regularized linear regression**

**Problem 1: Regularized linear regression**

**Visualizing the dataset**

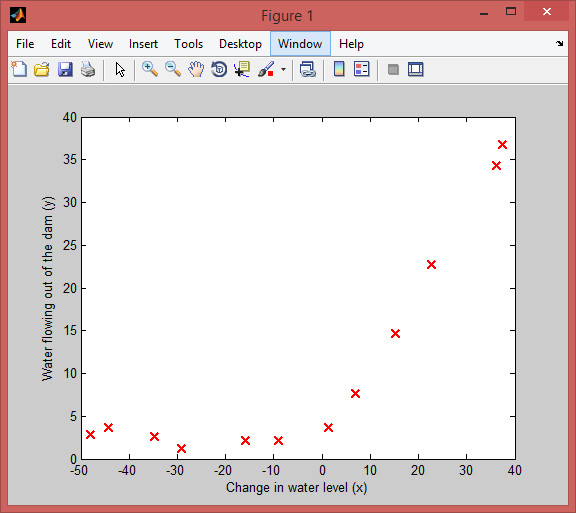
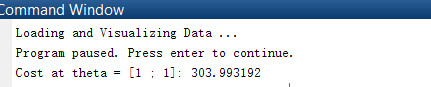
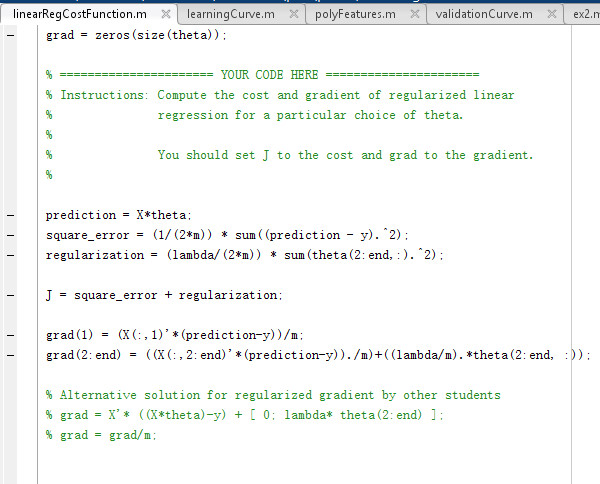
****

Figure 5: The training data

**Gradient of the Regularized linear regression cost function**

****

****



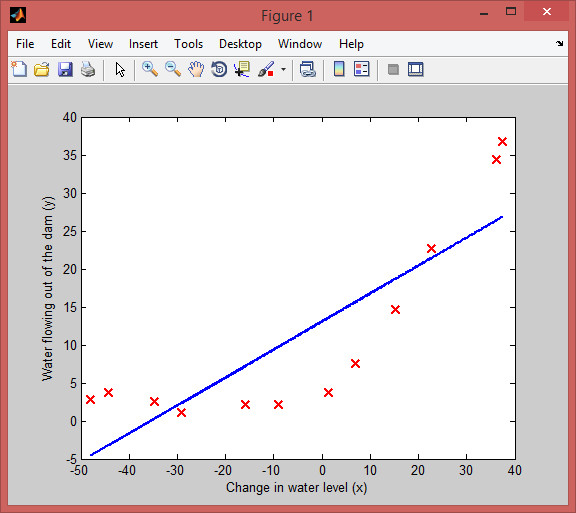
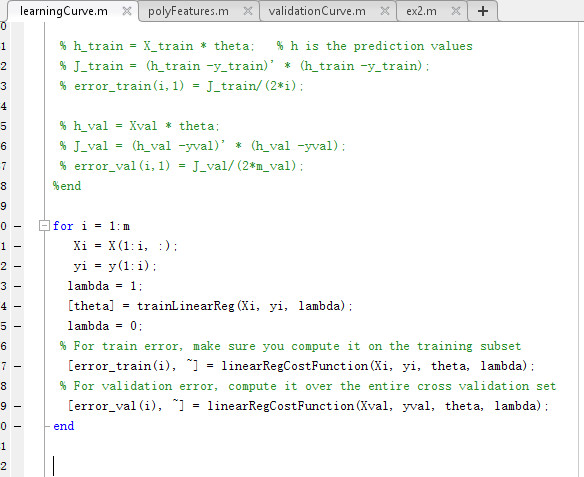


Figure 6: The best fit line for the training data

**Problem 2: Bias and Variance**

learningCurve Lambda = 0



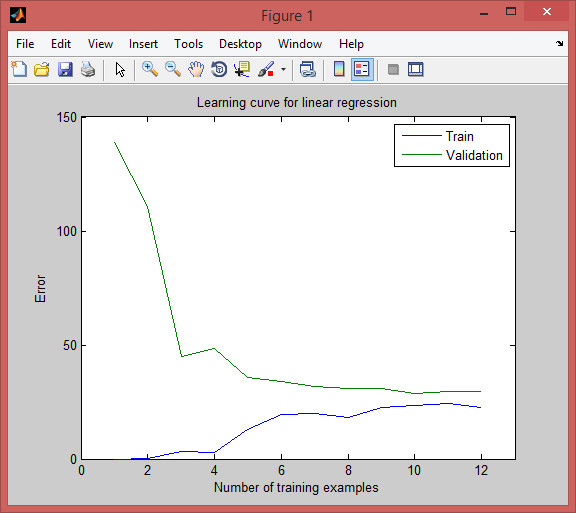


Figure 7: Learning curves

**Polynomial regression (5 points)**

in polyFeatures Lamba =0

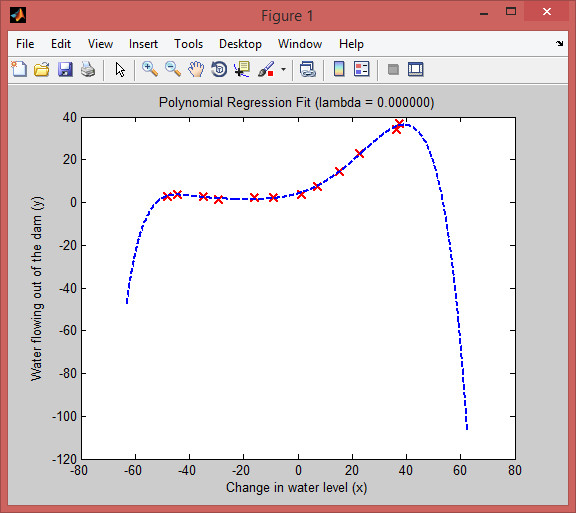
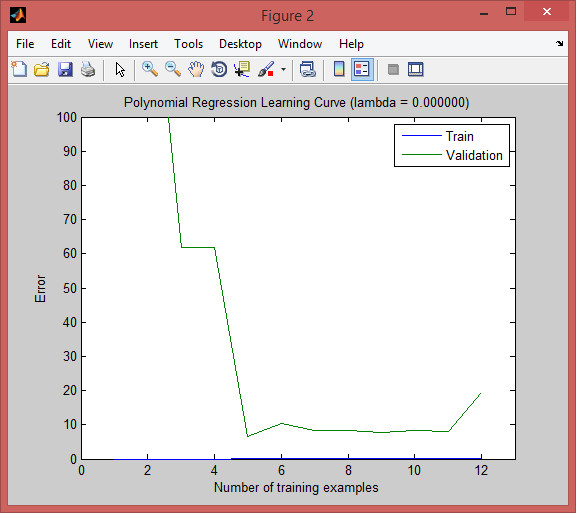
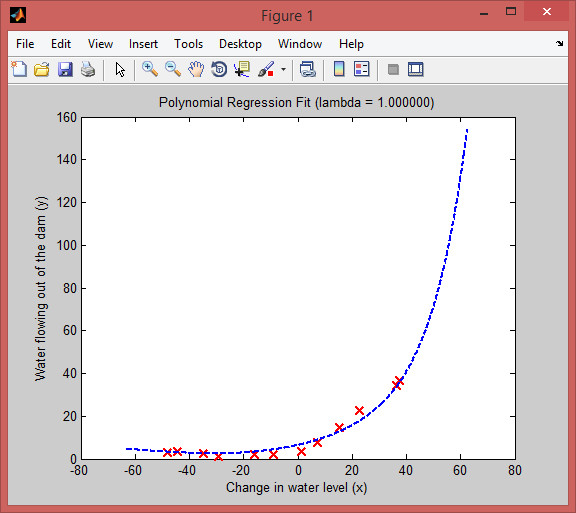
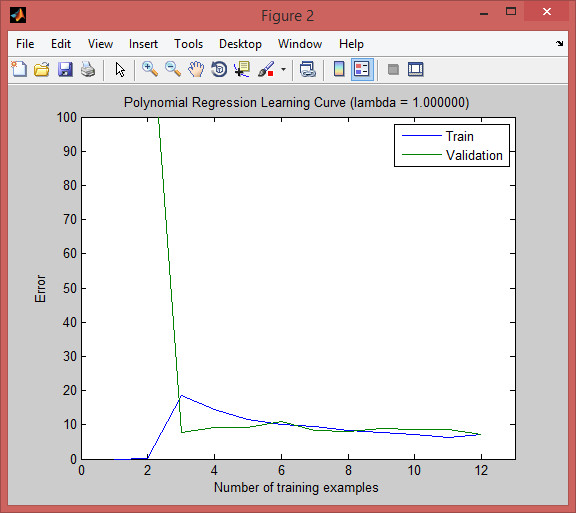


Figure 8: Polynomial fit for lambda = 0

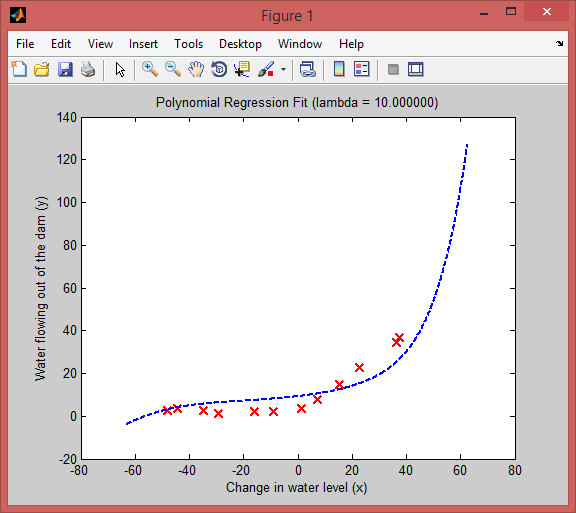


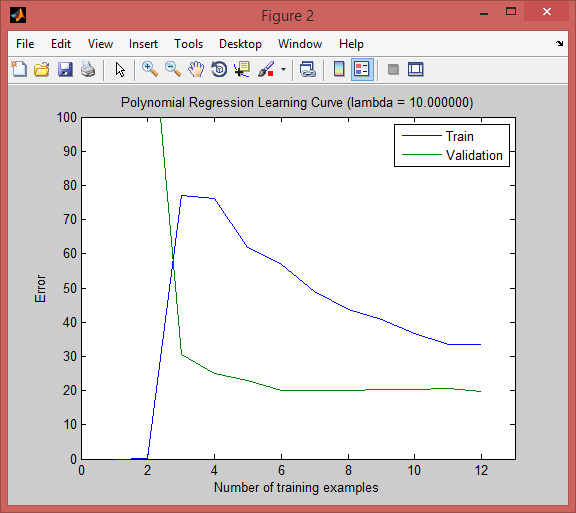
Lam =1



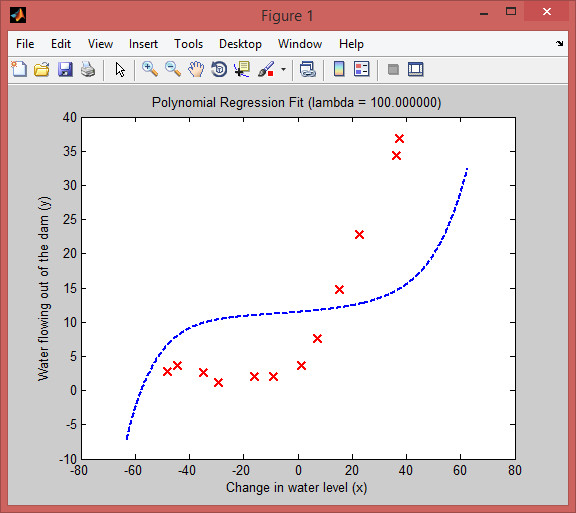


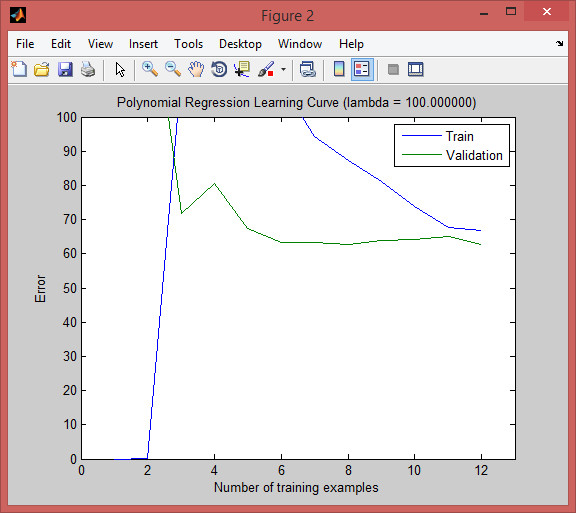
Lam=10





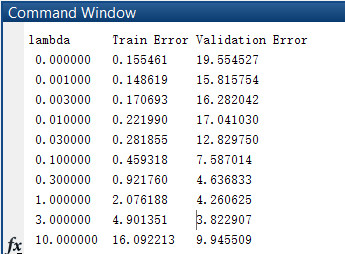
Lam=100

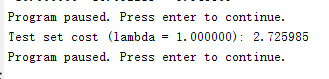




**Adjusting the regularization parameter (5 points)**

After compared with the lambda parameter take 1, 10,100, so found that when lambda = 1 generate a polynomial to the data and also a learning curve.





**Selecting A using a validation set**

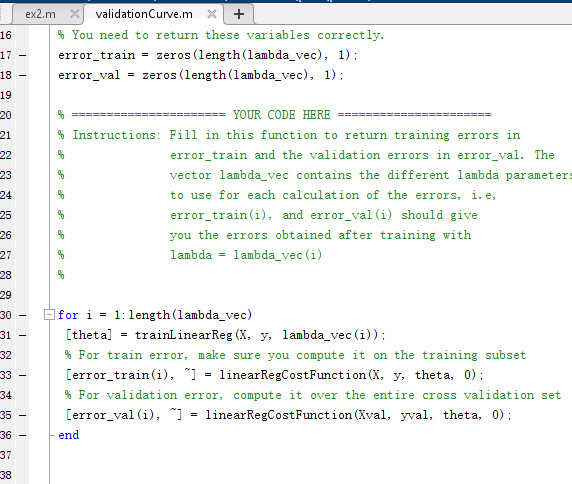
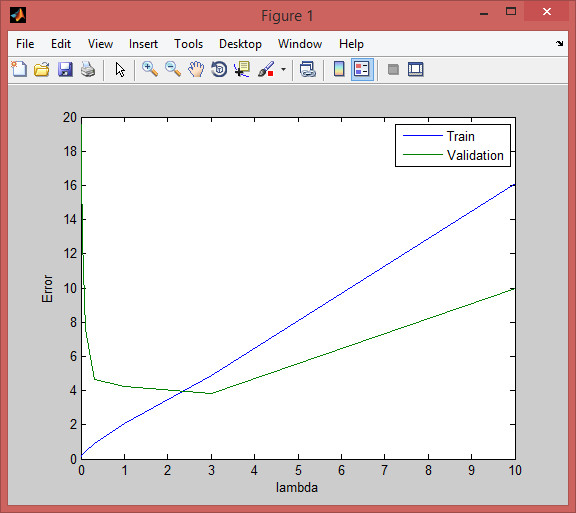
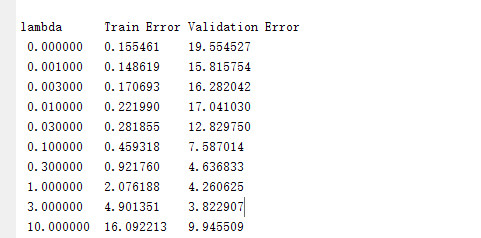
****

Figure 9: Learning curve for lambda = 1.0.

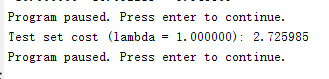
Selecting Lambda using avalidation set

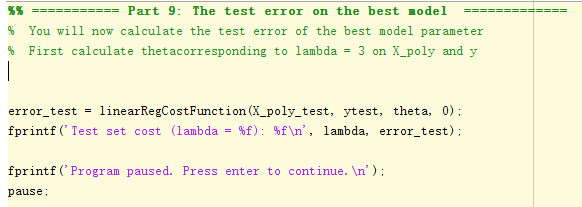




So best value is lam =1

**Computing test set error**

****



**Plotting learning curves with randomly selected examples**

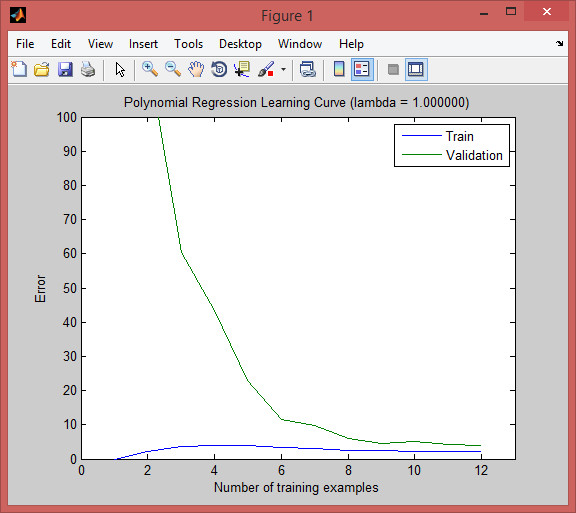


Figure 10: Averaged Learning curve for lambda = 1

**Problem 3: Building regularized models for the Boston housing data set (2156 points)**