# Kernel Methods for Pattern Analysis
# **Assignment Solution** 1


Rachit Garg
CS14B050

Keerthana S
CS13B041

Sphoorti K
CS13B042


February 18, 2017

# 1. Polynomial Curve Fitting

## 1.1 Regression Model

The function chosen by sir for us was $e^{\sin(2\pi x)} + \ln x$, and a gaussian mean noise was added to the distribution. We did the analysis for two cases, when the deviation was 1, and when the deviation was 0.15. The results and plots are as follows.

### 1.1.1 Standard deviation was 1

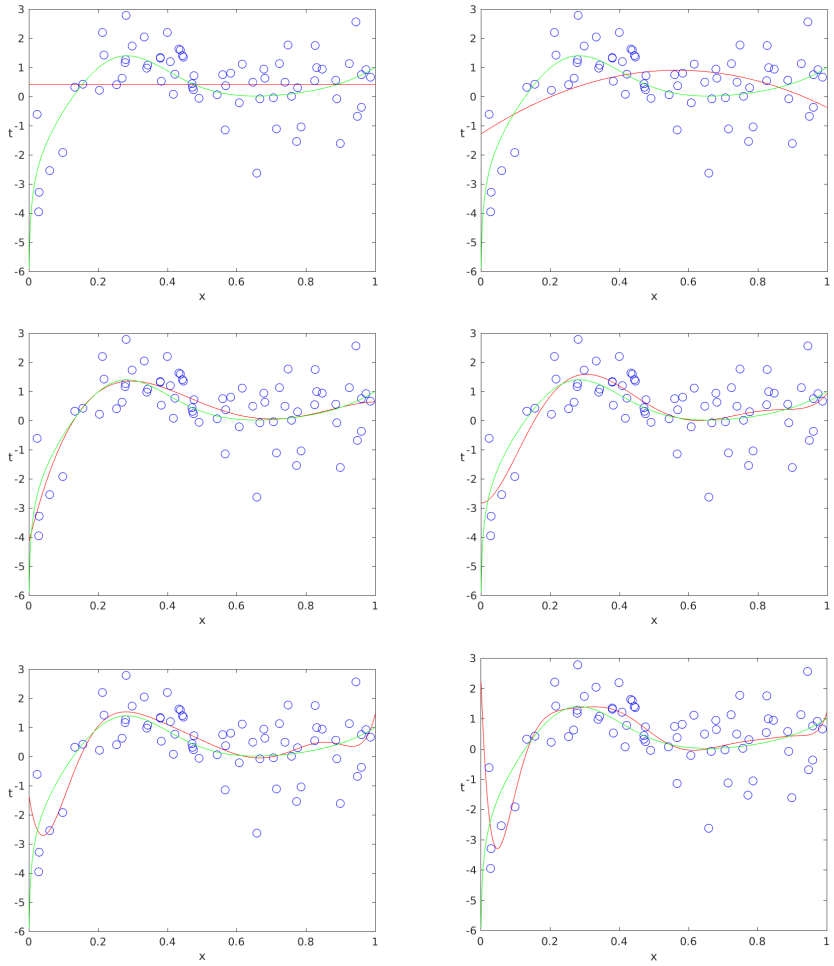



Figure 1: Plots of polynomials having various orders M with increasing complexity, shown as red curves, fitted to the noise added plots of the green curve for deviation equal to 1 and 70 train data points

Now we compare the results between different datasets. We observe for a different

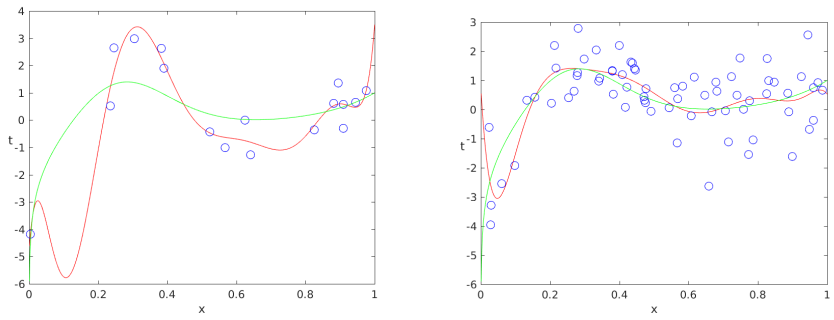



Figure 2: Plots of the solutions obtained by minimizing the sum-of-squares error function using the M = 9 polynomial for N = 15 data points (left plot) and N = 70 data points (right plot). We see that increasing the size of the data set reduces the over-fitting problem.

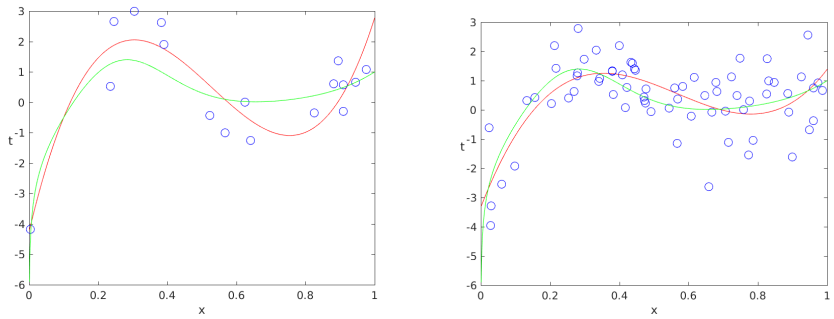



Figure 3: Plots of the solutions obtained by minimizing the sum-of-squares error function using the M = 3 polynomial for N = 15 data points (left plot) and N = 70 data points (right plot). We see that increasing the size of the data set reduces the over-fitting problem.

degree and note that as complexity increases, if we don't give enough data points, chance of overfitting increases much more.

1. Here we observed the usual decrease in train set error with increase in model complexity.

2. We also observed that the test set errors were more than the train set errors and the cross validated errors.

3. The best complexity model we found was for degree as 8. i.e for degree 8 the cross validated error was minimum. We observed the plots and came to the conclusion that the model for degree 6 is better, but we get the results for 8 as better because there is a lot of noise in the input data. Hence we decided to proceed with deviation as 0.15 as the second analysis.

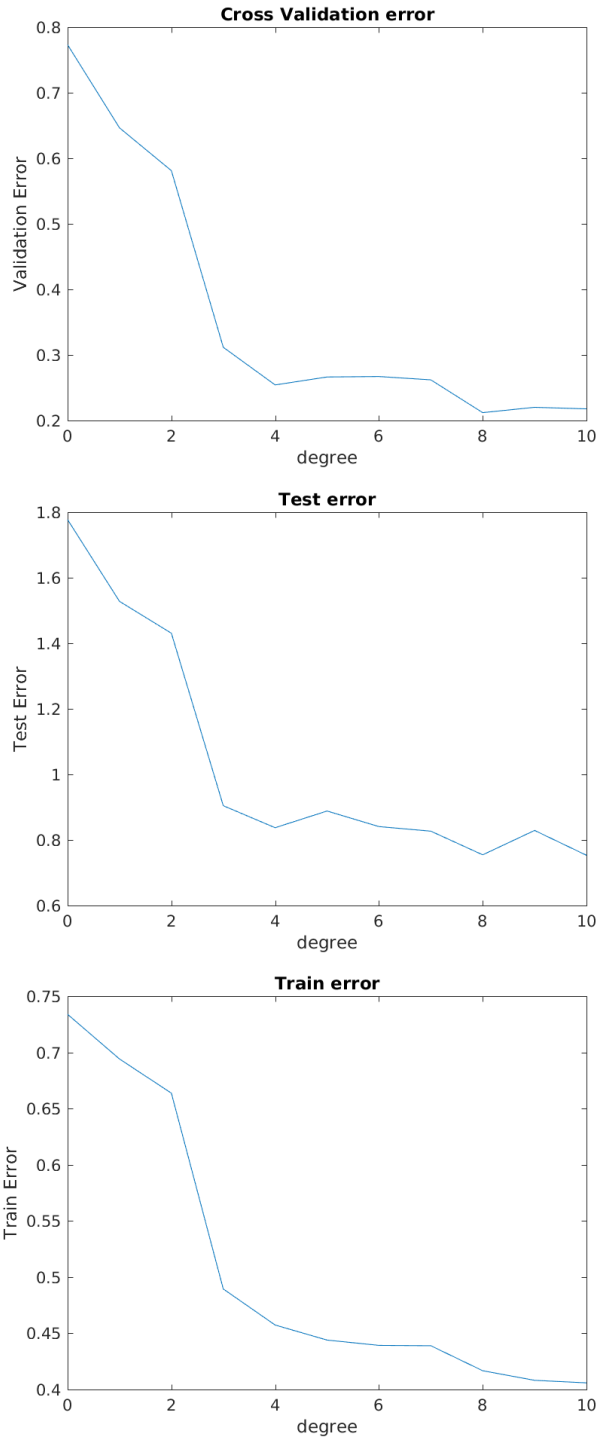



Figure 4: Here we observe the test, the cross validated, and the train errors with model complexity for train set as 100 data points

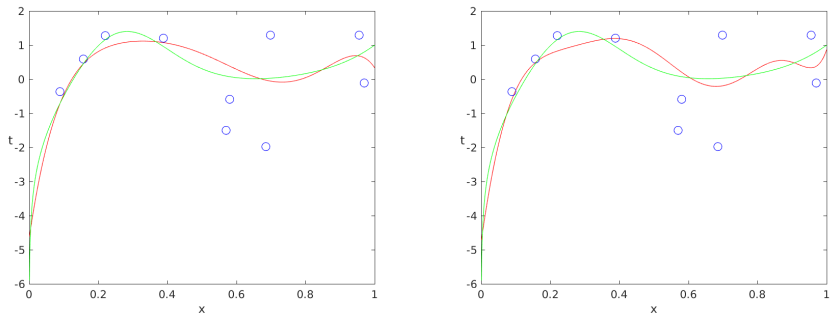



Figure 5: Cross validation data plots with our chosen model where validation set is 20 points for degree 6(left) and 8(right)

### 1.1.2 When Standard deviation is 0.15

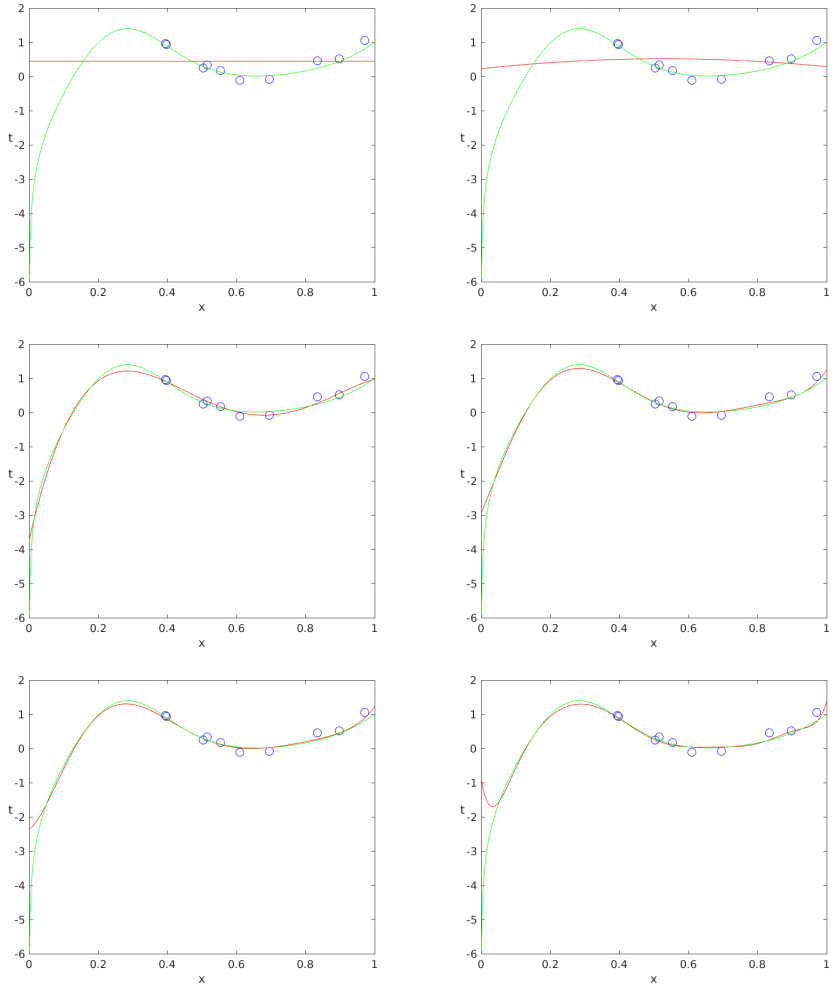



Figure 6: Plots of polynomials having various orders M with increasing complexity, shown as red curves, fitted to the reduced noise of deviation of 0.15 on cross validation set

1. Here we observed the usual decrease in train set error with increase in model complexity.
2. We also observed that the test set errors were more than the train set errors and the cross validated errors.
3. The best complexity model we found was for degree as 4. And the fitted red curves seemed to be really close to the curve to be predicted.

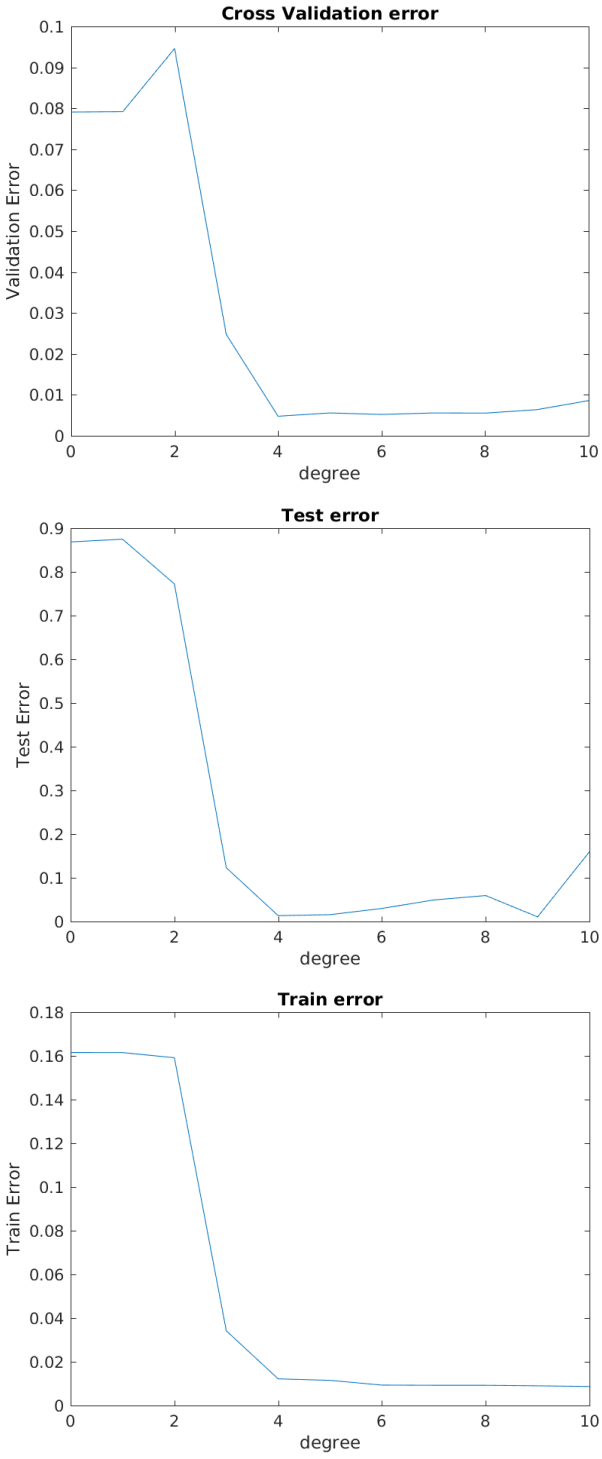



Figure 7: Here we observe the test, the cross validated, and the train errors with model complexity for train set as 100 data points for deviation 0.15

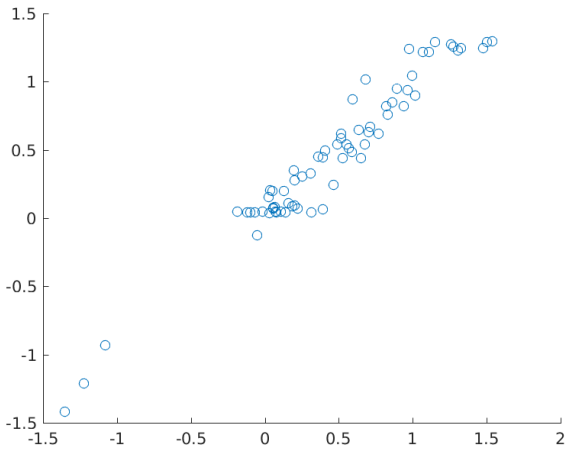



Figure 8: Scatter Plot for train data

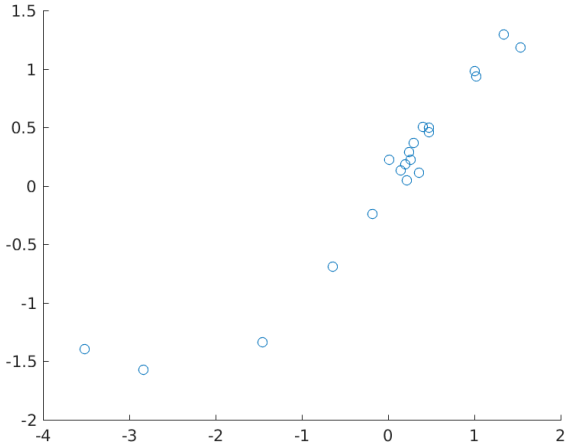



Figure 9: Scatter Plot for test data

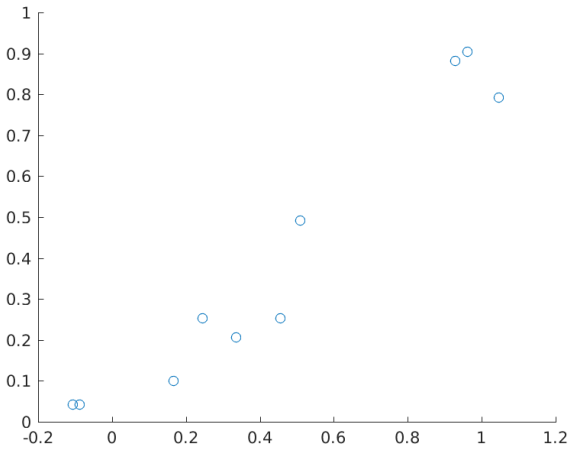



Figure 10: Scatter Plot for cross validation data

## 1.2 Regularization

We observed for the dataset where standard deviation was 1 as a large noise meant we could observe the effects of regularizations. We varied lambda from 1 to $10^{-20}$.
We observed that after we reduce lambda beyond a certain point, reducing it more causes no effect on our model. And this value at which lambda saturates is less for more complex models. Which is expected as with higher degree polynomials, the change can be more drastic and hence must be penalized more.

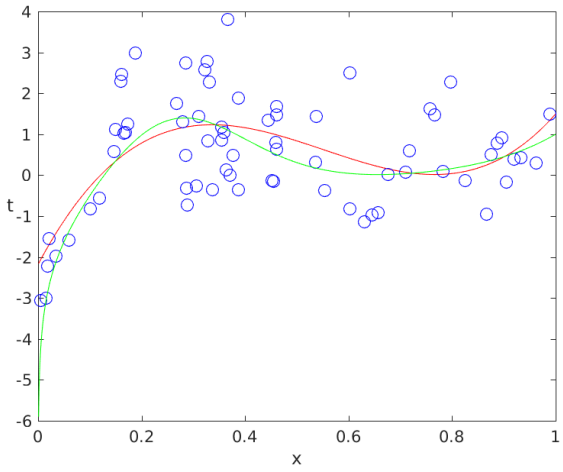



Figure 11: Plot for lambda = $10^{-3}$ and degree = 3

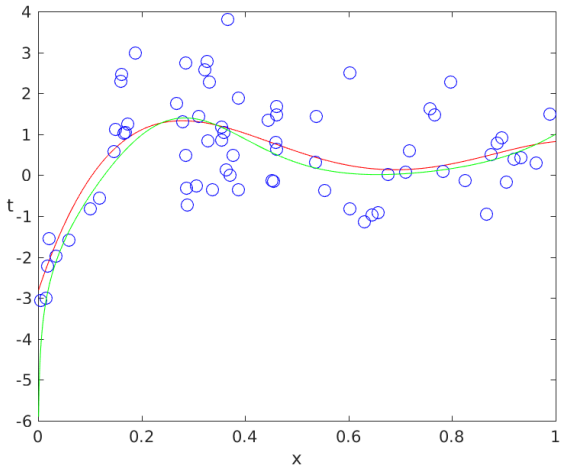



Figure 12: Plot for lambda = $10^{-4}$ and degree = 4

1. We observe that the best model is achieved with degree = 10, and lambda = $10^{-4}$.

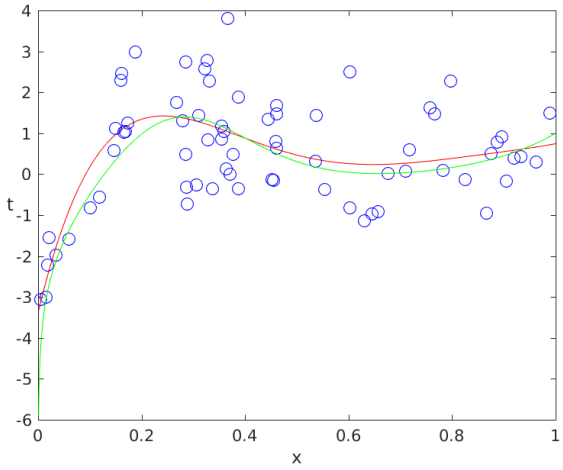



Figure 13: Plot for lambda = $10^{-6}$ and degree = 5

2. Generally for this dataset, lambda = $10^{-4}$ turns out to be a good approximation.
3. We plot degree with lambda to show our observations.

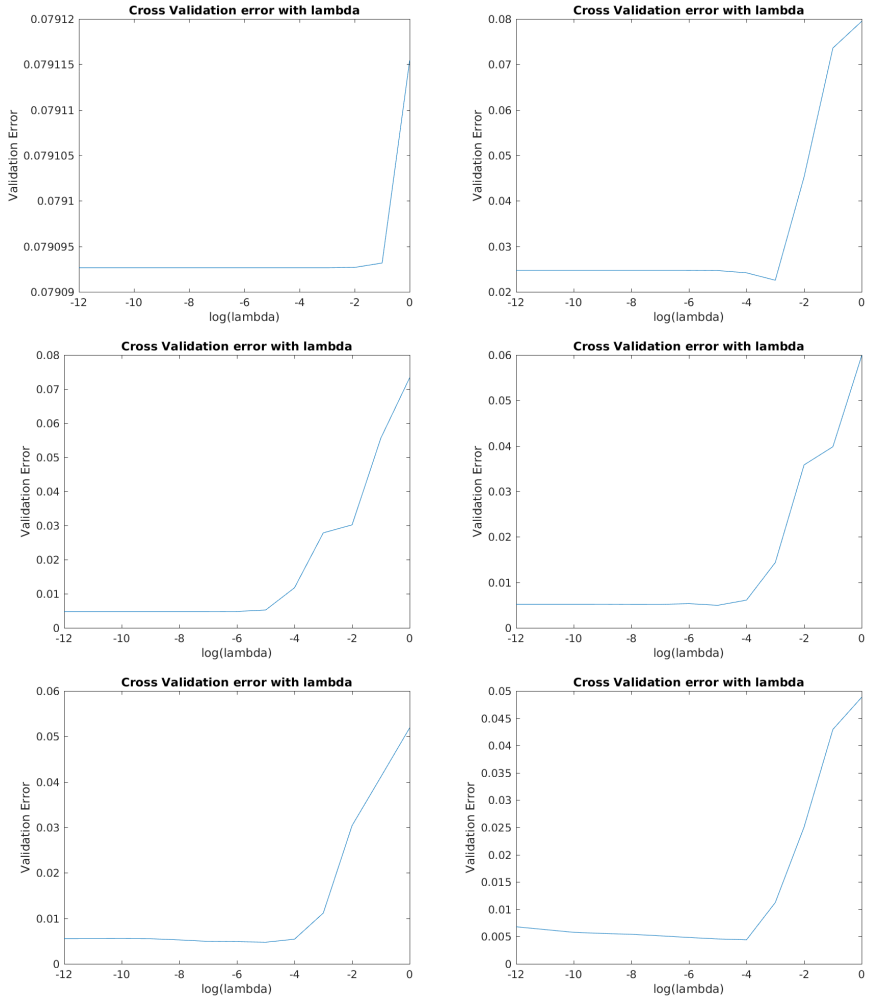



Figure 14: Cross Validation plots of polynomials having various orders M with different lambda's plotted, we observe that for higher and higher degree the model is performing better and there seems to be a minimum around the $\lambda = 10^{-4}$ mark

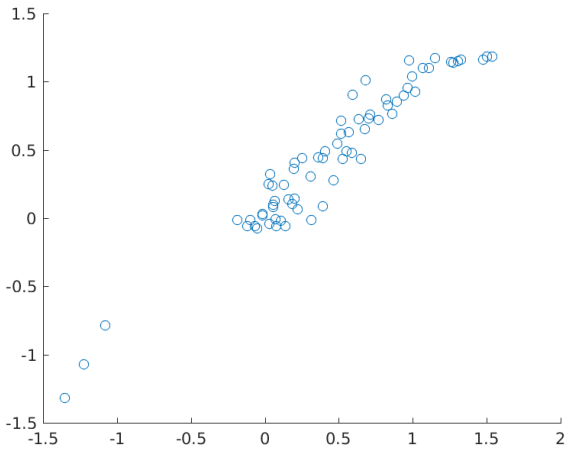



Figure 15: Scatter Plot for train data

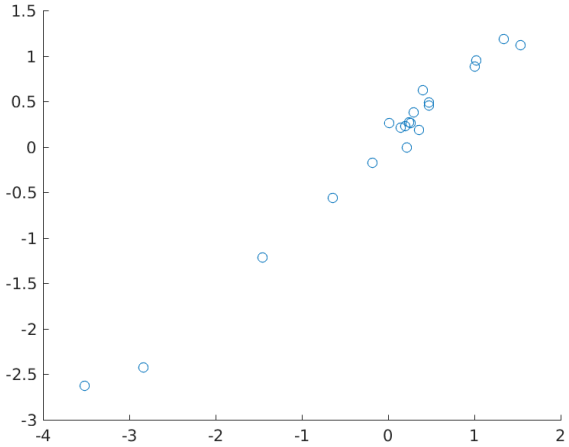



Figure 16: Scatter Plot for test data

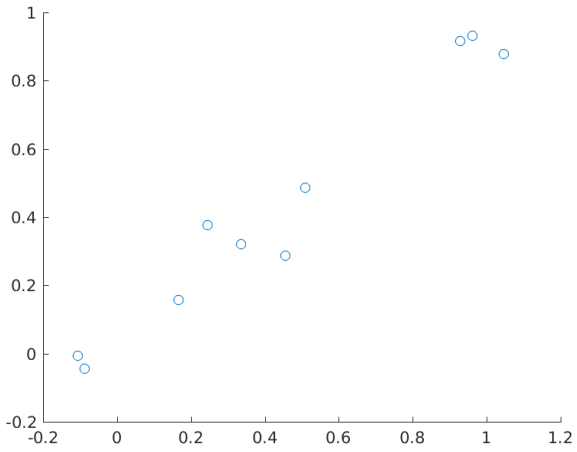



Figure 17: Scatter Plot for cross validation data

# 2. Polynomial Basis Function

## 2.1 Regression Model

We first show the plotted functions for different datasets.

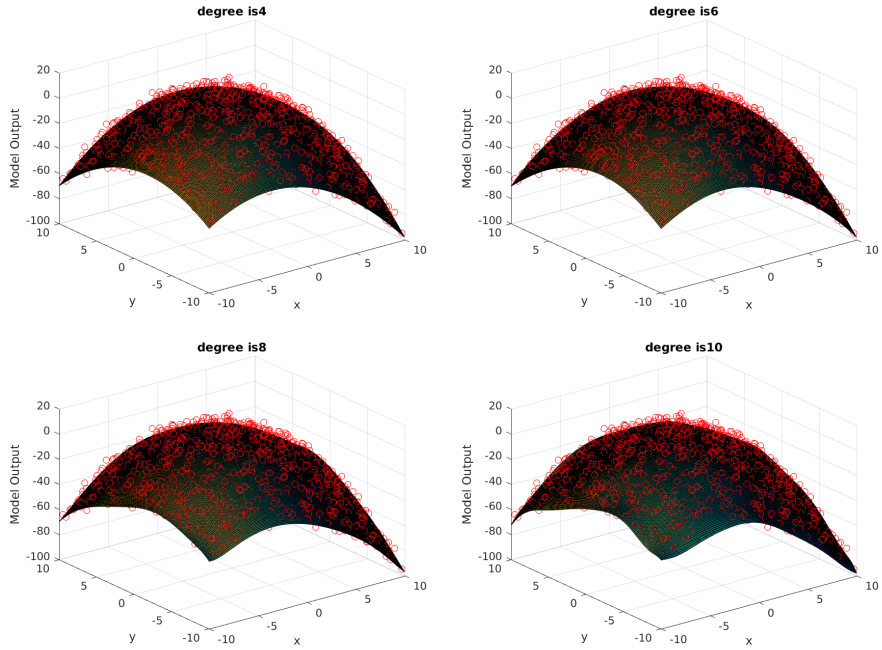



Figure 18: Output plots for training data points equal to 1000

1. We observe a rather insignificant change in the model shape with the increase in degree.
2. We plot the data for various train sets, and note the fact that outputs of the model in the case of small train set are overfitted.
3. We also note down the error plots of train error, test error, and validation error for all the training cases.

1. We observe that a degree of 2 works very well for higher input train sets. For smaller train sets we witness overfitting, where train error is driving to zero but cross validation and test errors are increasing really fast.
2. We conclude that a degree of 2 is suitable in complexity of the model as well as gives a less error. The min mean square error on the validation set is of the value 0.0233 which means that the model is a good approximation.

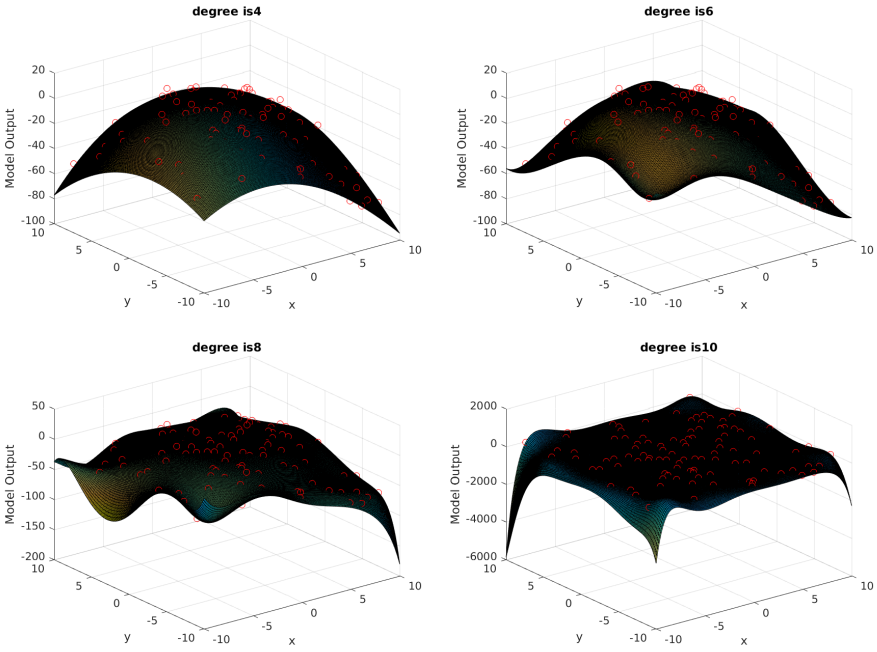



Figure 19: Output plots for training data points equal to 100

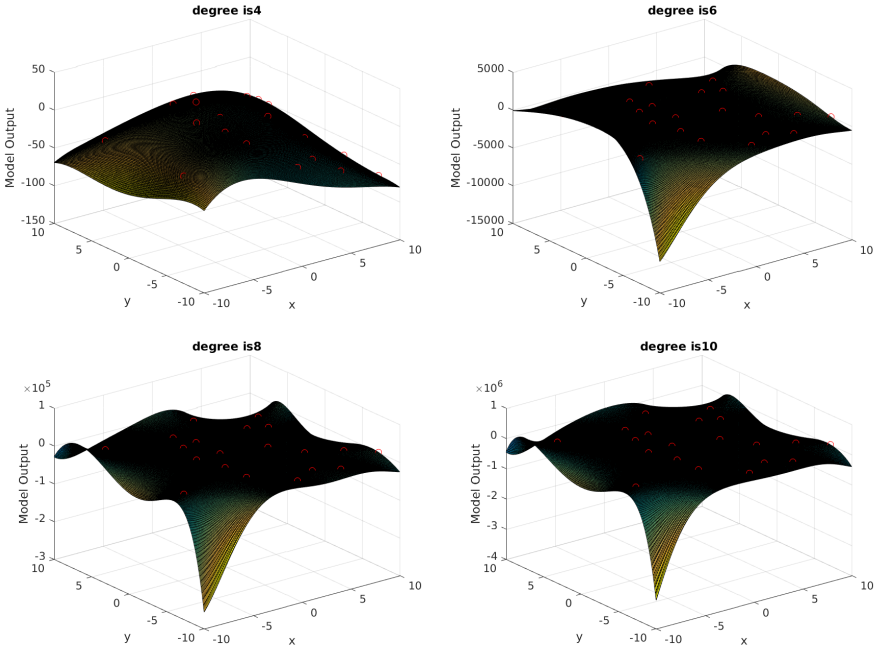



Figure 20: Output plots for training data points equal to 20

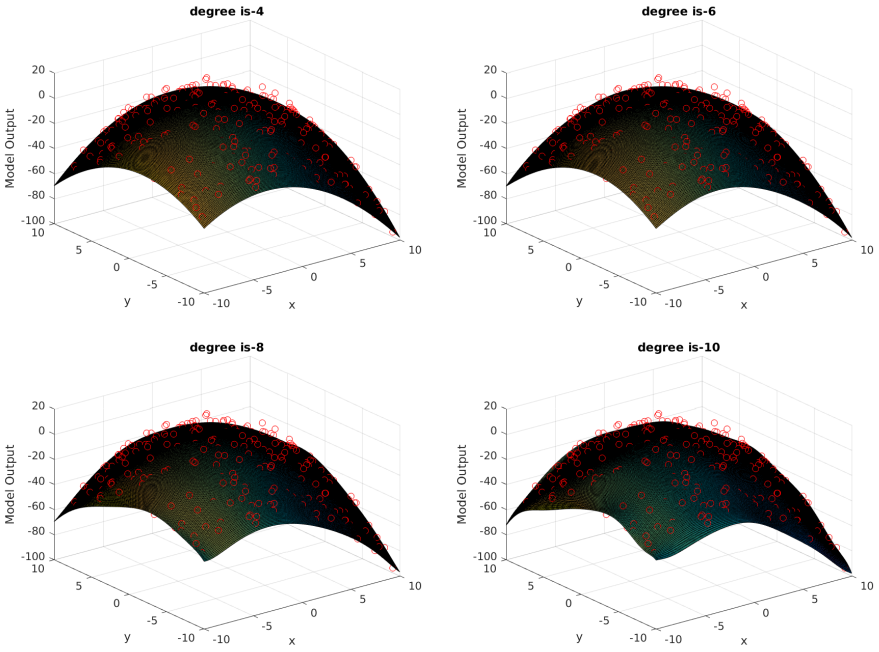



Figure 21: Output plots for cross validated data

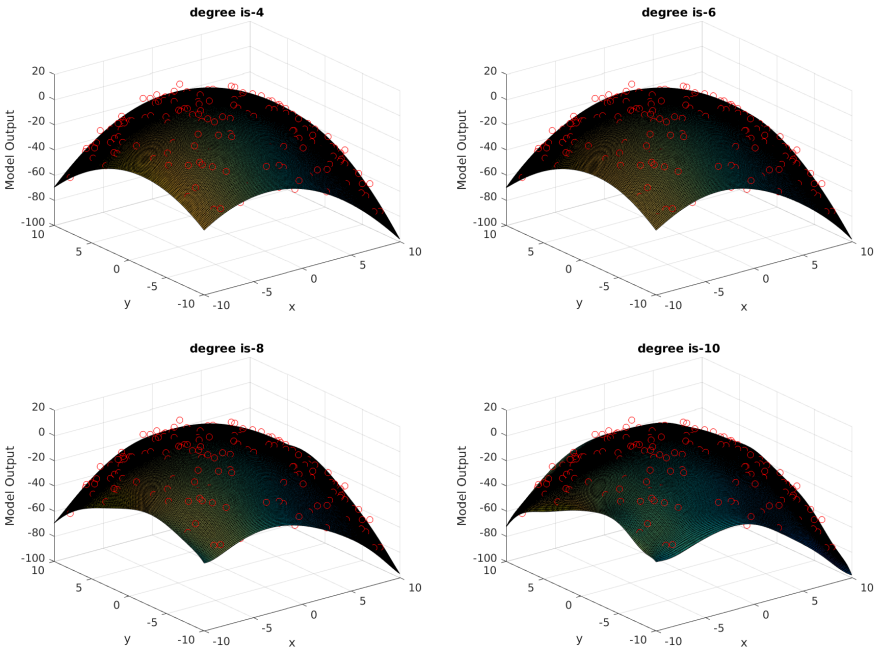



Figure 22: Output plots for test data

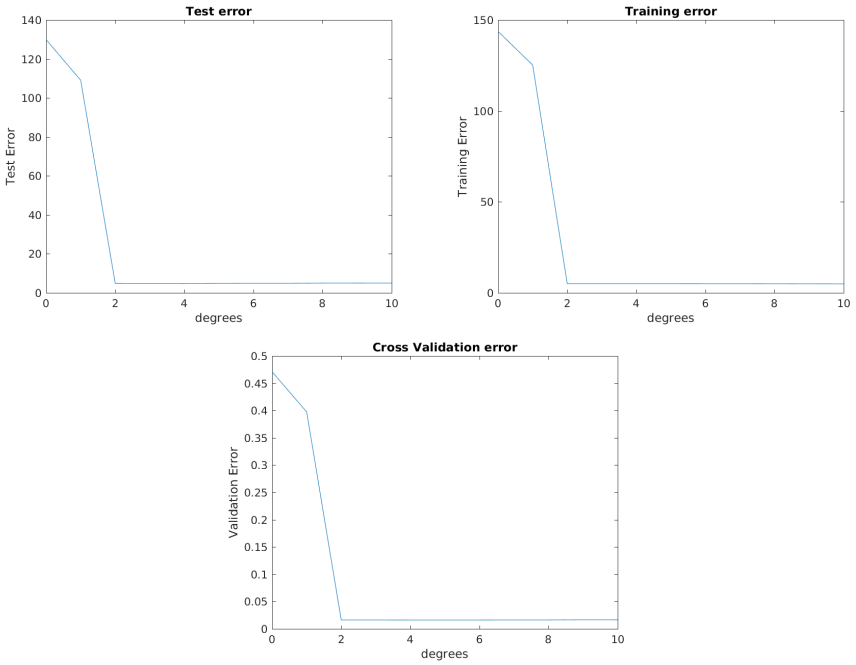



Figure 23: Variation of errors with degree trained on 2000 data points

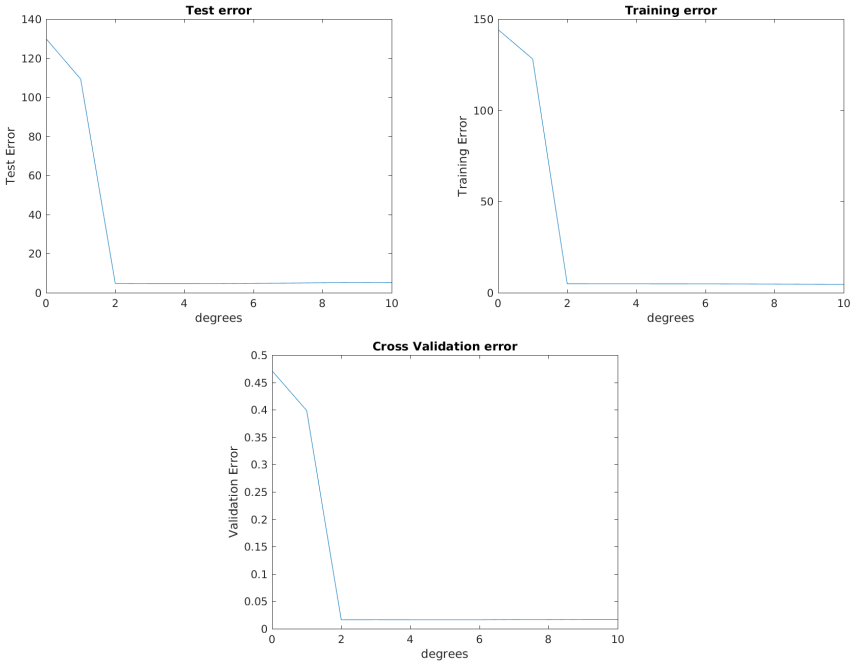



Figure 24: Variation of errors with degree trained on 1000 data points

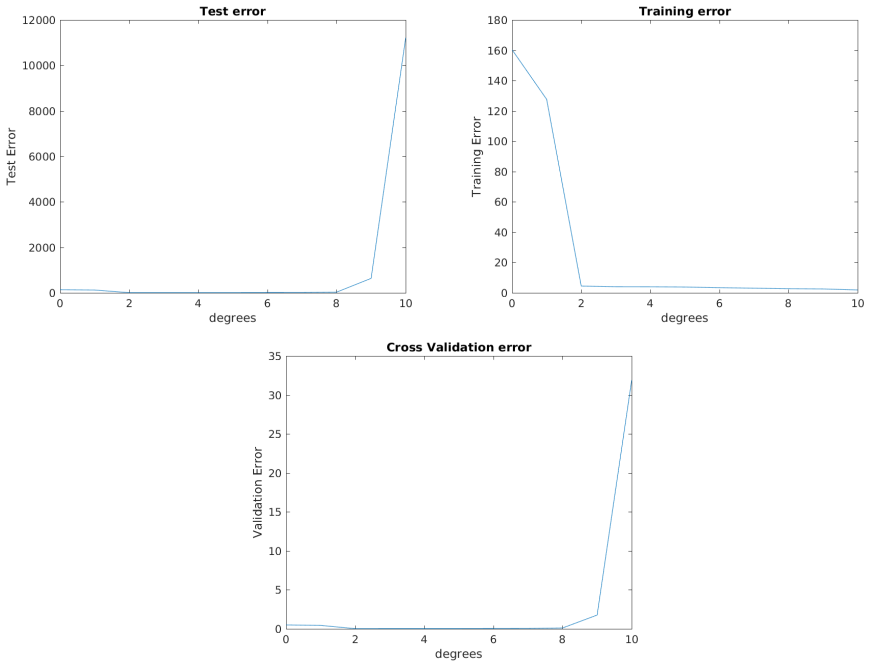



Figure 25: Variation of errors with degree trained on 100 data points

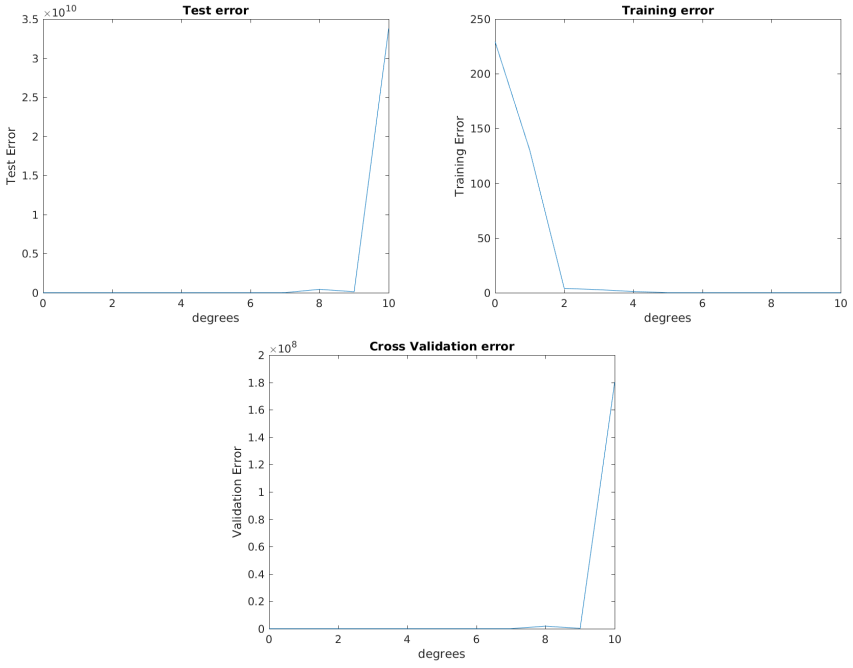



Figure 26: Variation of errors with degree trained on 20 data points

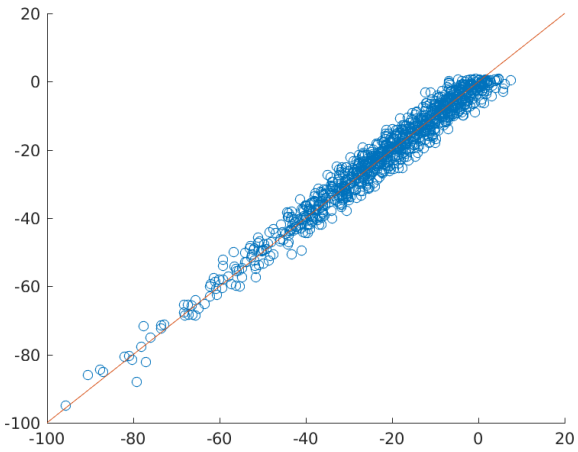



Figure 27: Scatter Plot for train data

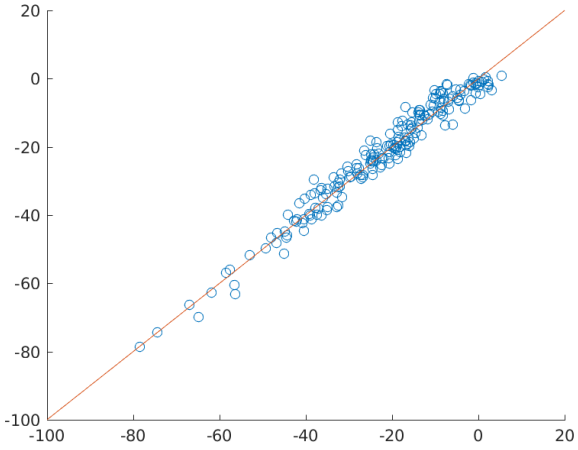



Figure 28: Scatter Plot for test data

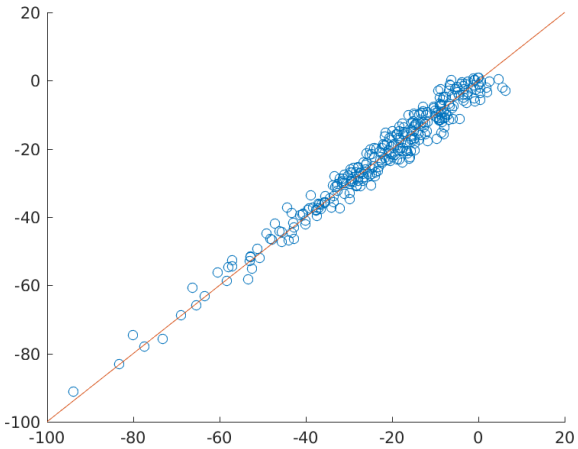



Figure 29: Scatter Plot for cross validation data

## 2.2 Regularization

1. We observe that with regularization we achieve the minimum mean square error of 0.0161. Which is better than than the error we observed in the non regularized cases.

2. The minimum degree observed to be was degree 6. It seems that around $\lambda = 10^{-1}$, best results are observed. We varied lambda along closer ranges and also observed. Similar errors were observed. The various plots are as follows.

1. Fine Tuning was done, but similar results were obtained. Error is minutely changing around our fine tune for us to observe any significant changes.

1. Final regularized plots of the model output vs the real output was as follows.

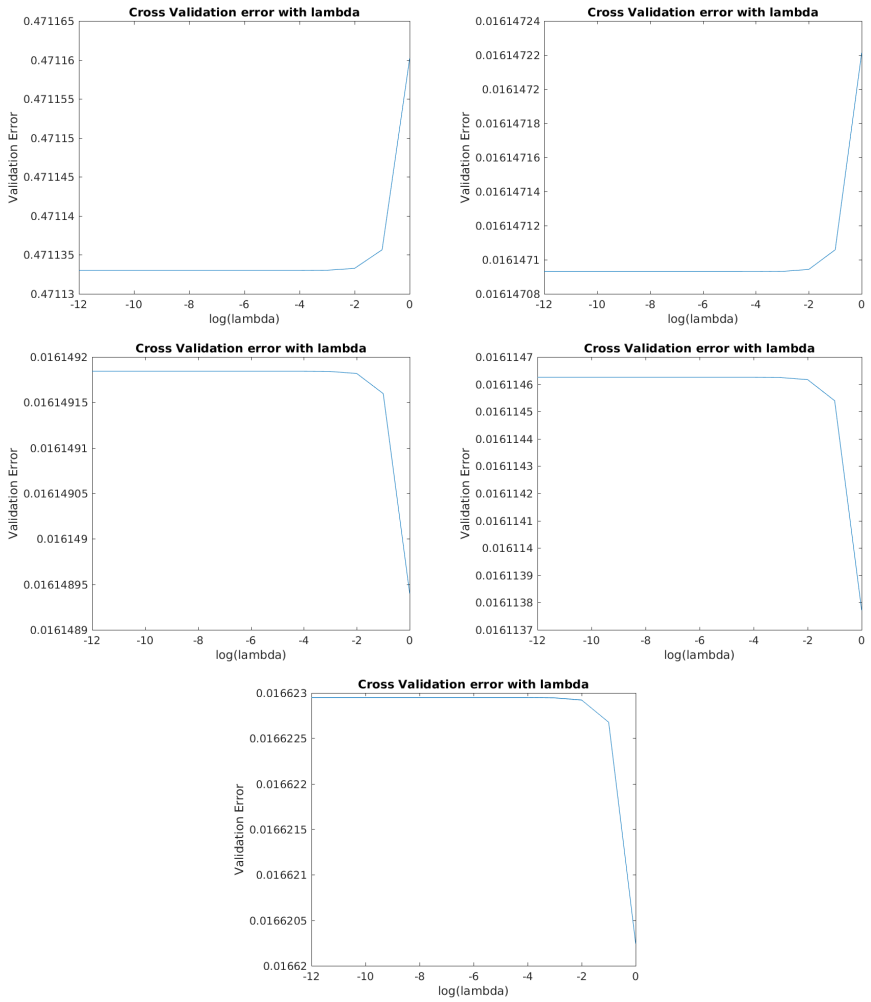



Figure 30: Cross Validation plots of data having various orders M with different lambda's plotted, we observe that for higher and higher degree the model is performing better and there seems to be a minimum around the $\lambda = 10^{-1}$ mark

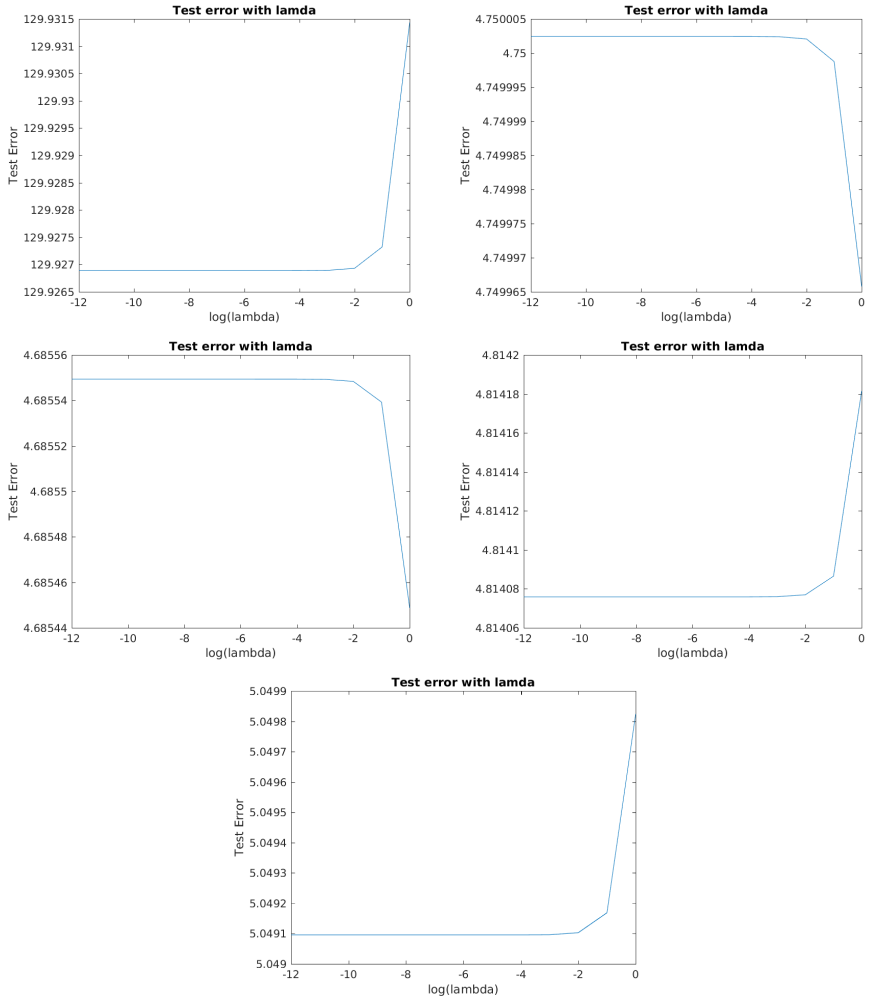



Figure 31: Test plots of data having various orders M with different lambda's plotted, we observe that for higher and higher degree the model is performing better and there seems to be a minimum around the $\lambda = 10^{-1}$ mark

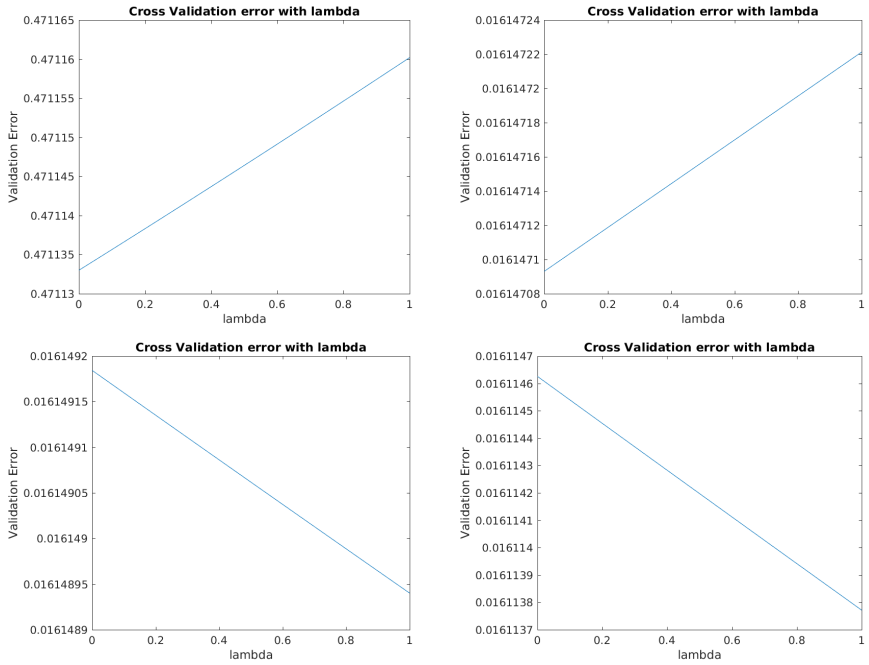



Figure 32: Cross Validation plots of data having various orders M with different lambda's plotted, we try to fine tune the parameters here, but error is very minute

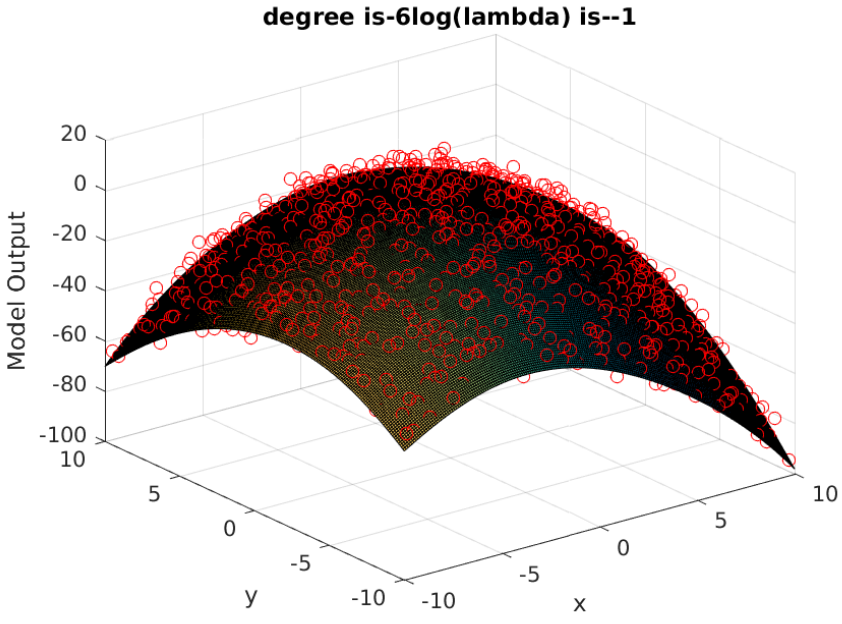



Figure 33: The final best regularized plot that we get

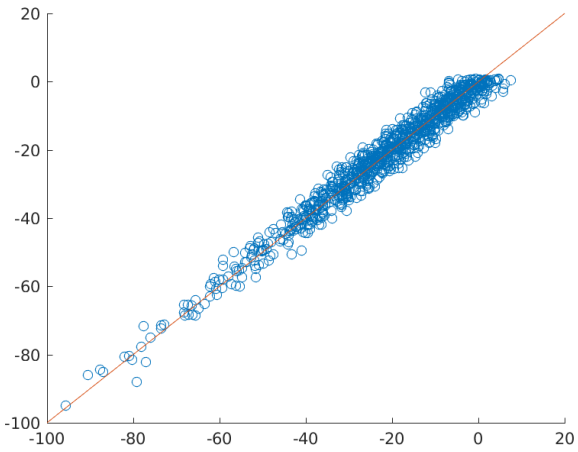



Figure 34: Scatter Plot for train data

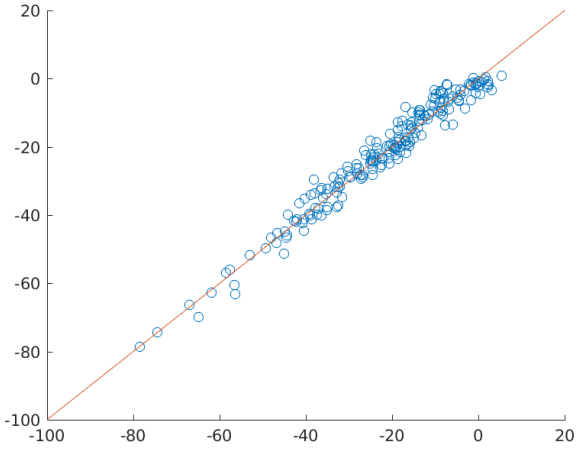



Figure 35: Scatter Plot for test data

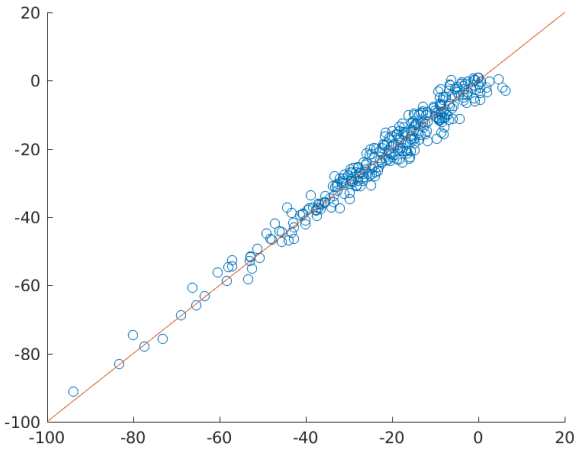



Figure 36: Scatter Plot for cross validation data

# 3. MLFFNN for univariate data

Function to be fitted: $e^{\sin(2\pi x)} + \ln x$
Number of points=100
Train:Test:validation =70:20:10
Number of hidden layers=1

## 3.1 Choosing best configuration for MLFFNN

Choosing the best configuration is done by validation.The plot of error on validation set for different number of hidden neurons is found to be:

We choose the complexity(no.of hidden neurons) which has least error on validation

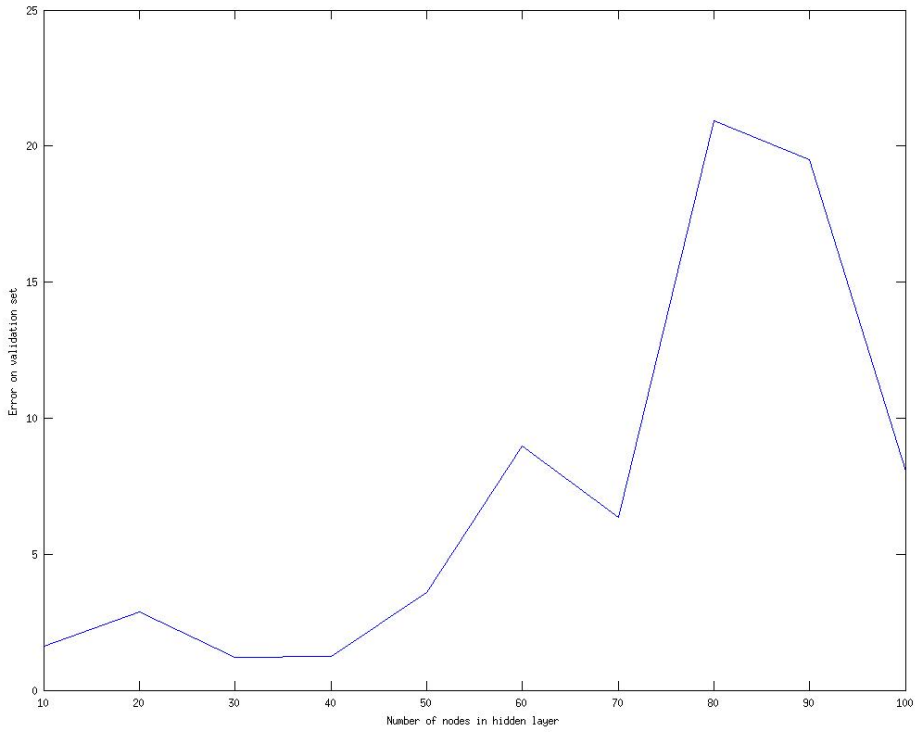



Figure 37: Plot showing validation error vs complexity

set.We see that error decreases initially with increase in no.of neurons ,this is because the complexity of network with very few number of networks is not enough to fit the data, then error increases because it overfits the data.We see that error on validation set is minimum when number of neurons is in the range of (30,40).

On the other hand we observe that error on training data decreases continuously with increase in number of neurons.(till n=100)

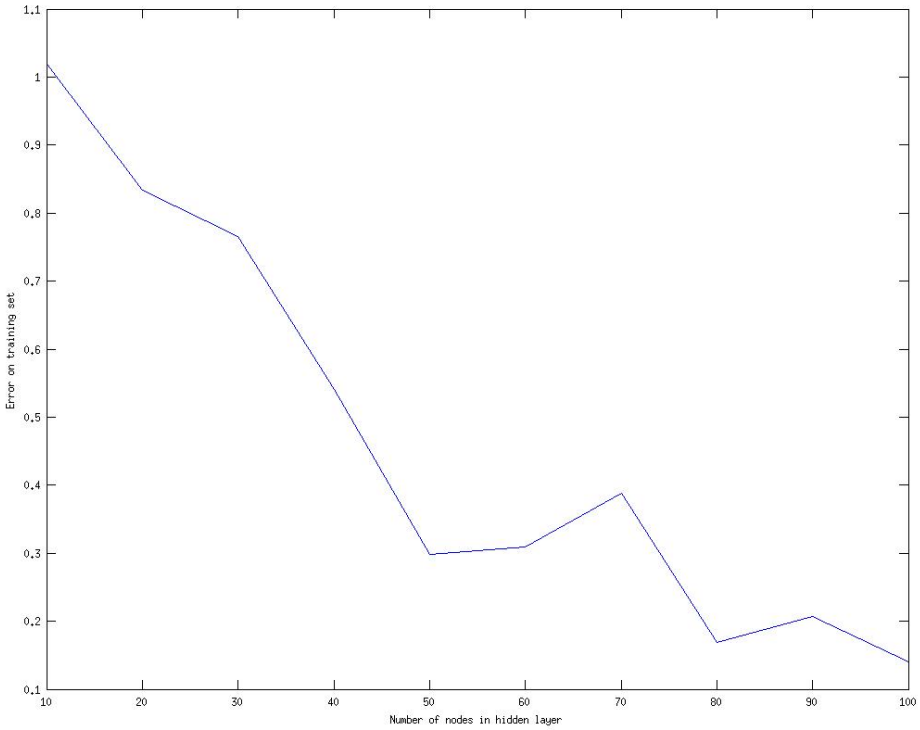



Figure 38: Plot showing training error vs complexity

Plotting validation error vs neurons between 30 to 40, we get the best configuration. From the above figure we choose best configuration to be n=33

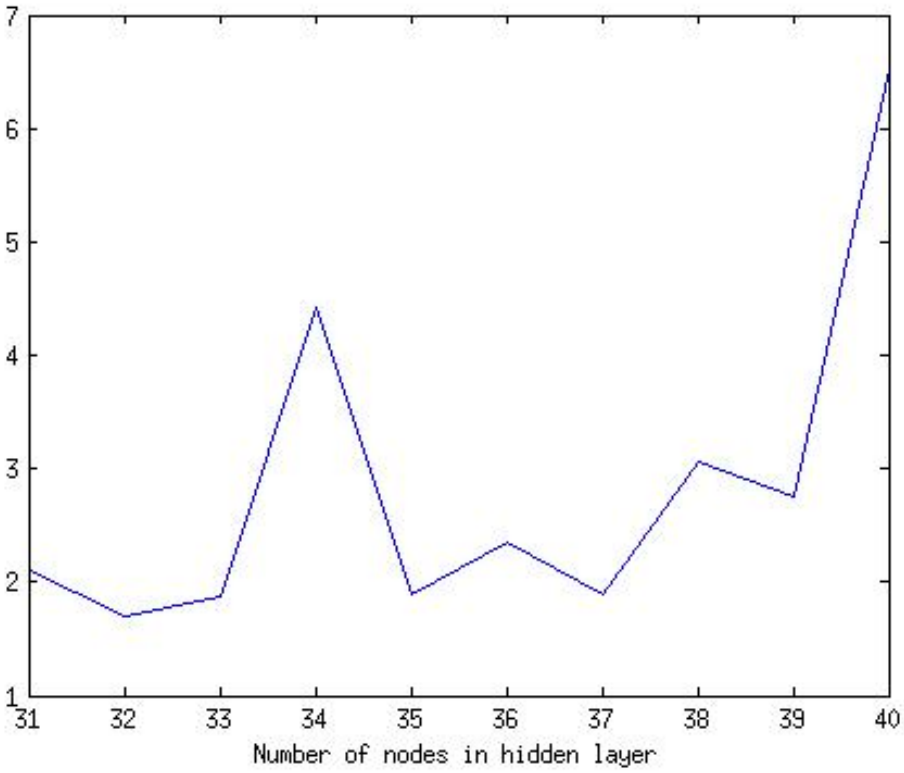



Figure 39: Plot showing best configuration

## 3.2 Scatter plot

For the best configuration i.e n=33,scatter plot looks like

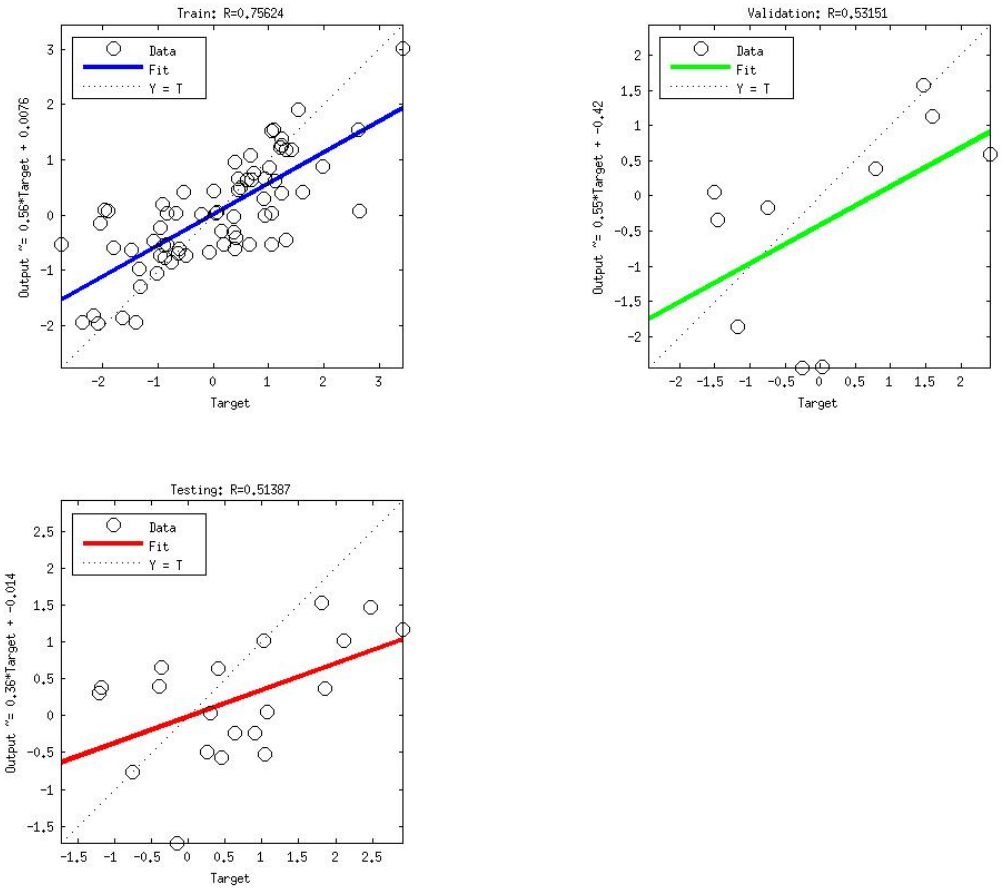



Figure 40: Scatter plot with target output on x axis and model output on y axis for n=33(number of hidden neurons)

From the above scatter plot we can see that even best configuration MLFFNN is not good enough to approximate the given univariate function(mean squared error value is far above 0).This can be overcome by using more no.of hidden layers or by having more number of training examples,which will be more of correct representation of the given function

# 4. MLFFNN for bivariate and multivariate data

## 4.1 MLFFNN for bivariate data

Dataset:7
Number of hidden layers:2

### 4.1.1 Choosing best configuration

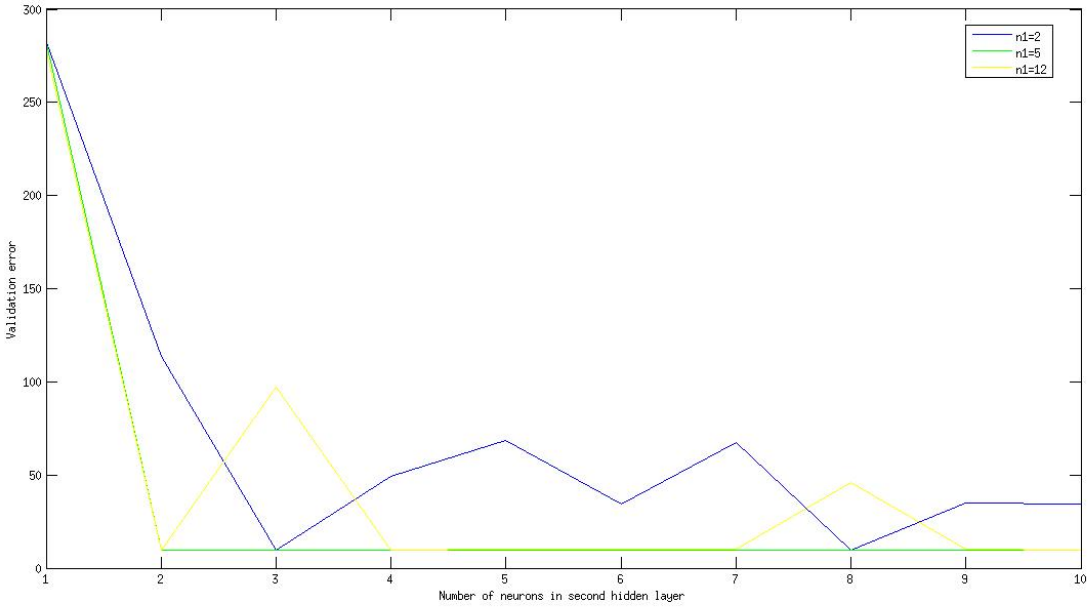



Figure 41: Validation error for different number of neurons in hidden layer 2 for fixed number of neurons in hidden layer 1,here n1=number of neurons in hidden layer 1, n2= number of neurons in hidden layer 2

From the plot we observe that across different no. of neurons in hidden layer 1, minimum error is nearly same, hence we choose the one with less complexity i.e n1=2.

For n1=2, we see that error is minimum for n2=3.Hence best configuration is n1=2,n2=3.

### 4.1.2 Scatter plot

Scatter plot for best configuration i.e n1=2, n2=3 for training data,validation data and test data

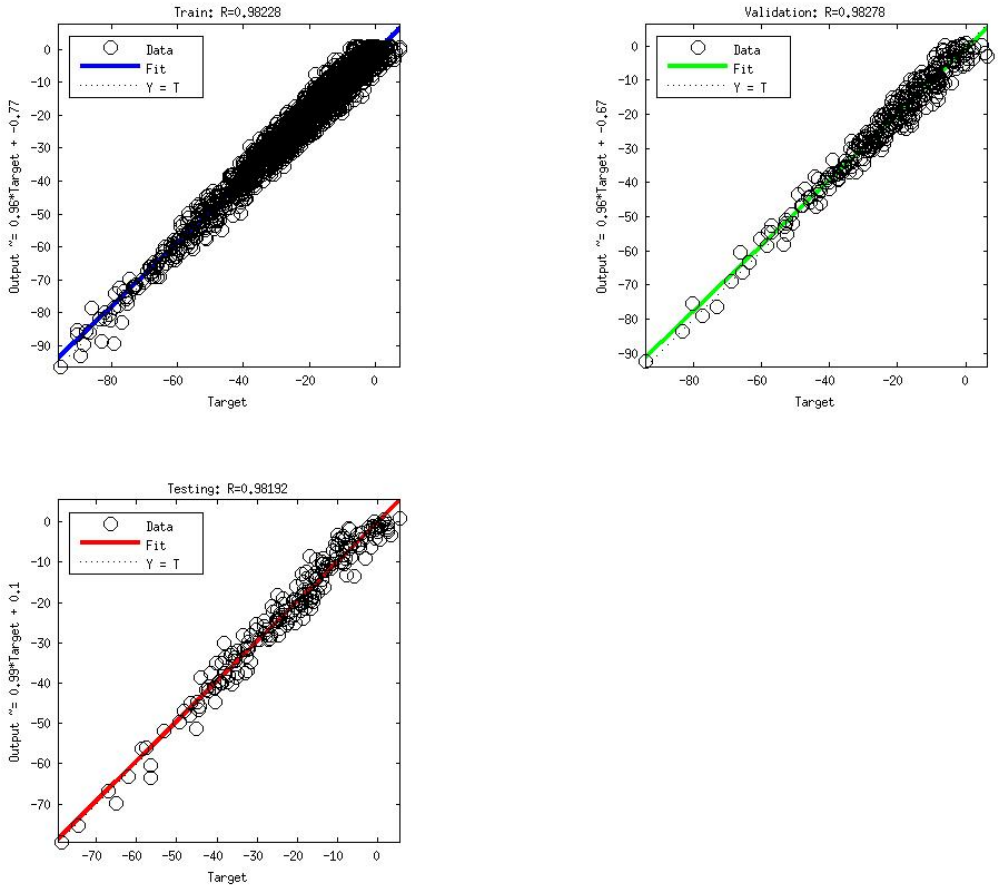
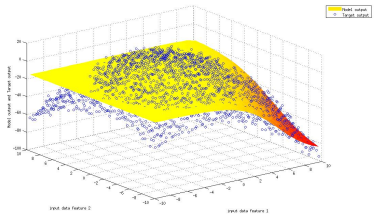


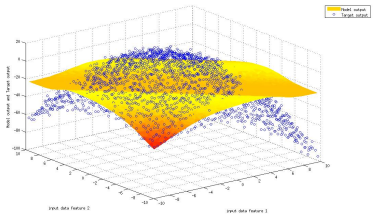

Figure 42: Scatter plot of model output vs target output for best configuration of MLFFNN

### 4.1.3 Surface plots for various complexities

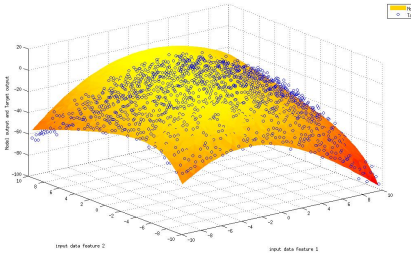
n1=no. of hidden neurons in first hidden layer
n2=no. of neurons in second hidden layer



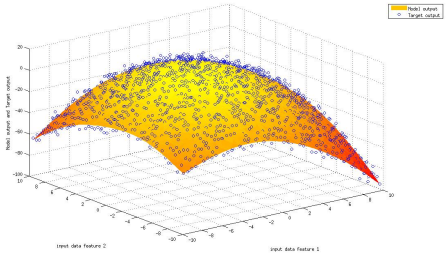
(a) Plot showing model output and target output for n1=1,n2=1



(b) Plot showing model output and target output for n1=1,n2=5



(a) Plot showing model output and target output for n1=2,n2=3



(b) Plot showing model output and target output for n1=2,n2=10

From the above plots we observe that given fixed no. of neurons in first hidden layer, model output and target outputs become nearly equal with increase in number of nodes in 2nd hidden layer,this is because the initial complexity is not enough to fit the given data,with increase in complexity training error decreases,which is not necessarily the case with validation data,because it overfits after some point.

We also observe that with increase in number of nodes in first hidden layer, the model output converges with original data.

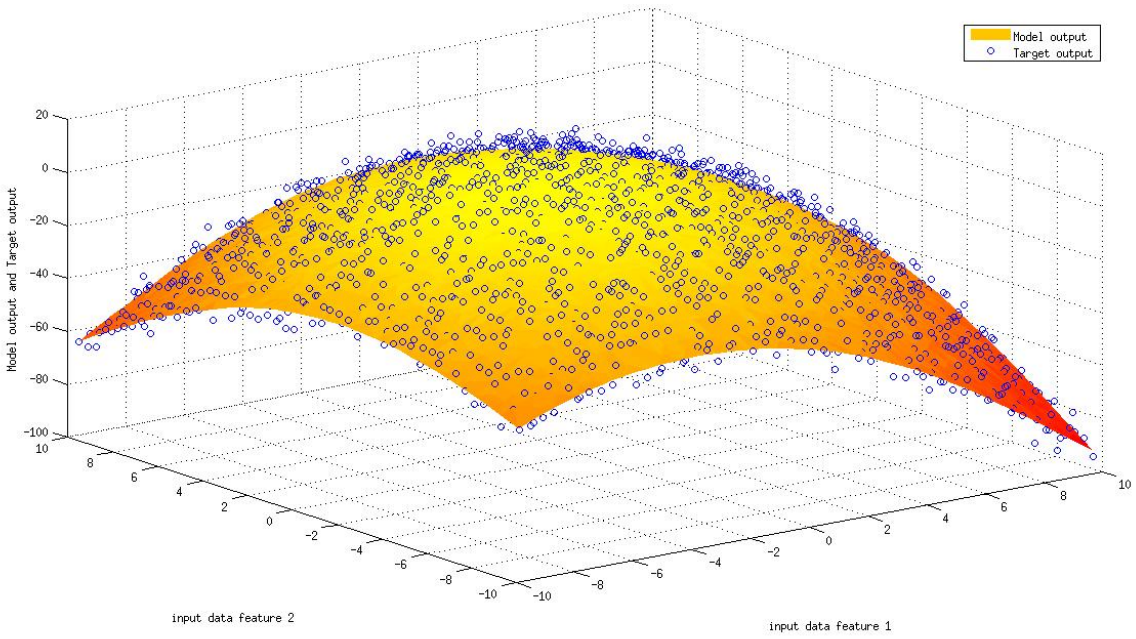



Figure 45: Plot showing model output and target output for n1=3,n2=2

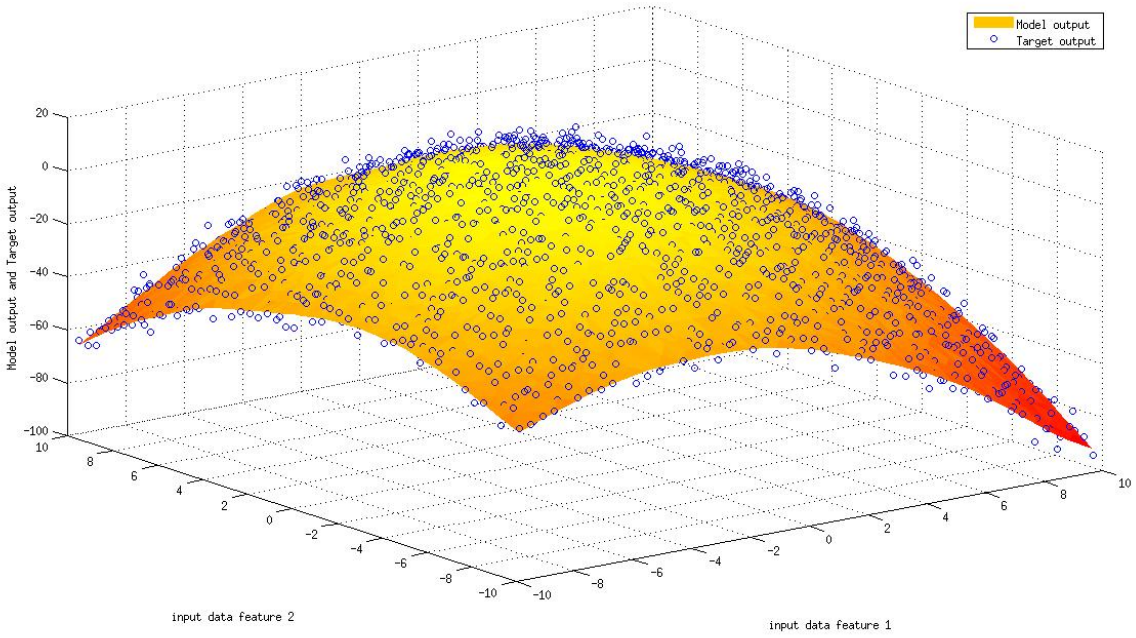
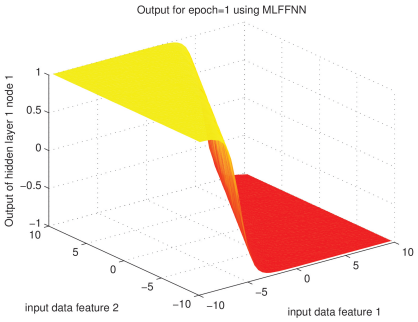
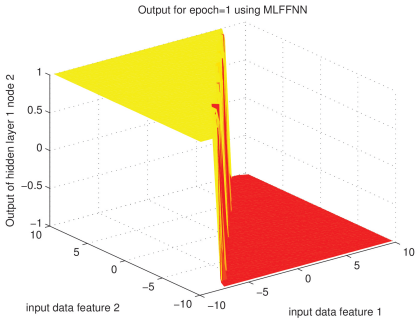


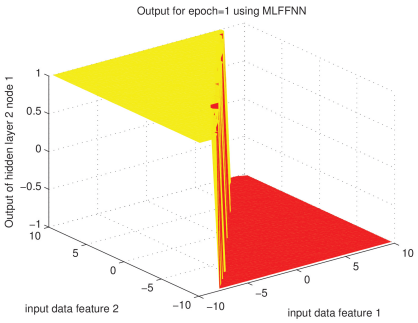

Figure 46: Plot showing model output and target output for n1=10,n2=5

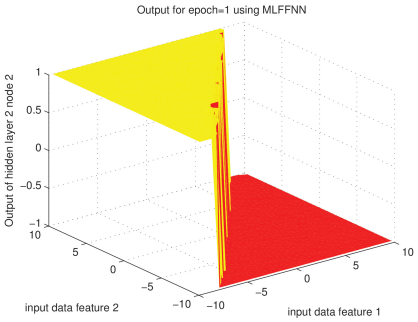
### 4.1.4 Outputs of output nodes and hidden nodes for varying no .of epochs

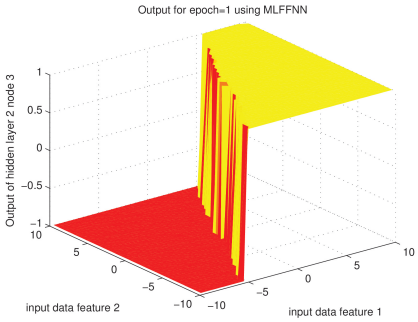
**No. of epochs=1**

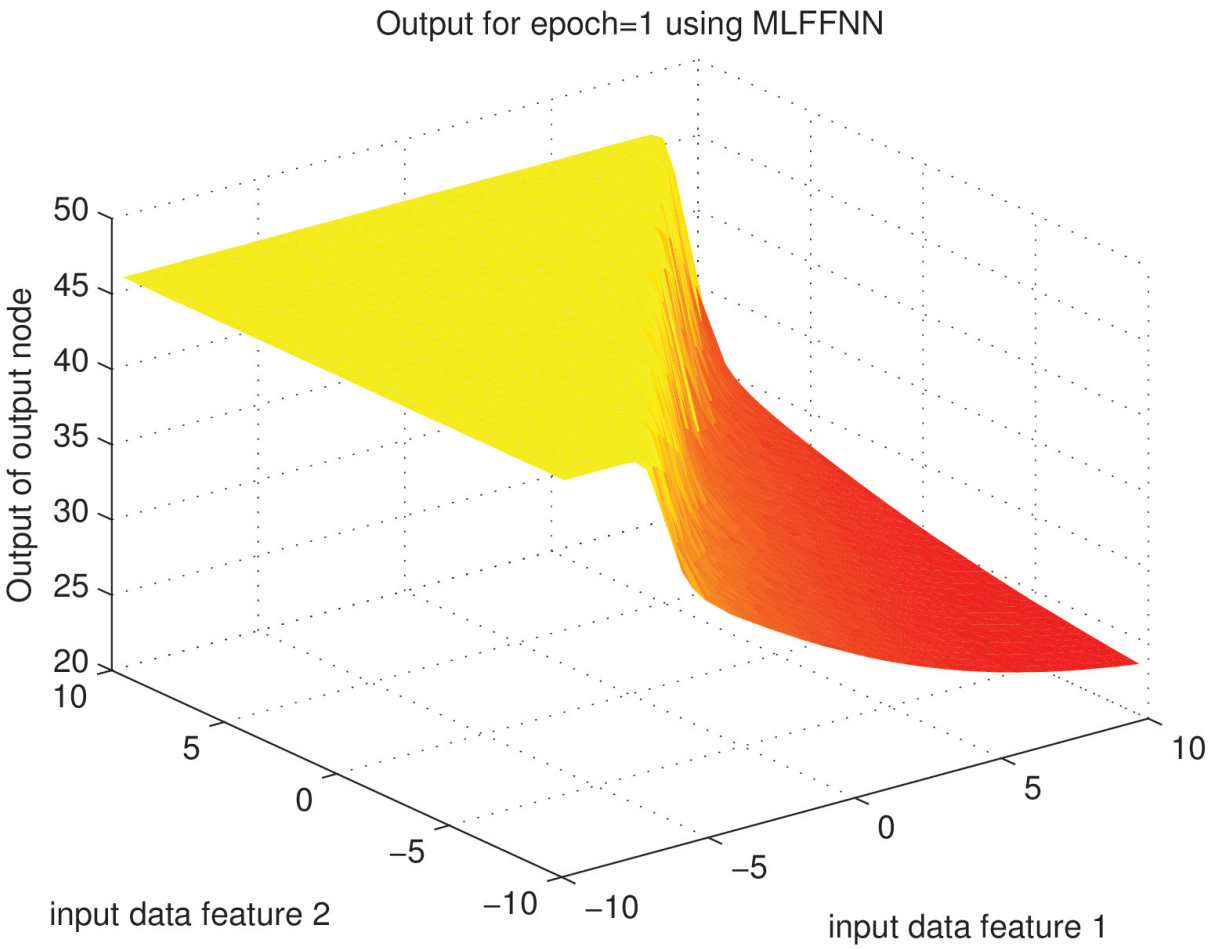


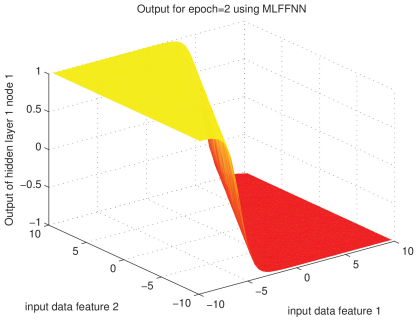



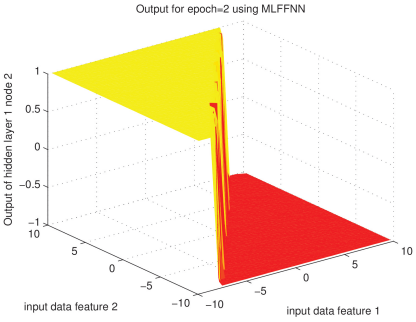



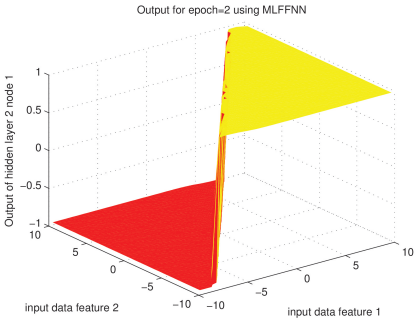



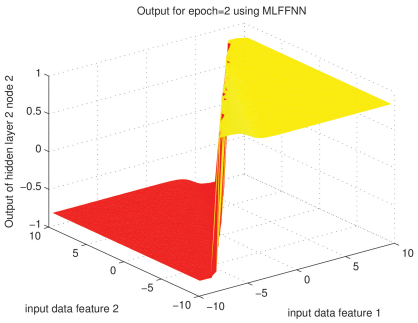

Output for epoch=1 using MLFFNN
Output of output node
50
45
40
35
30
25
20
10
5
0
−5
−10
input data feature 2
−10
−5
0
5
10
input data feature 1

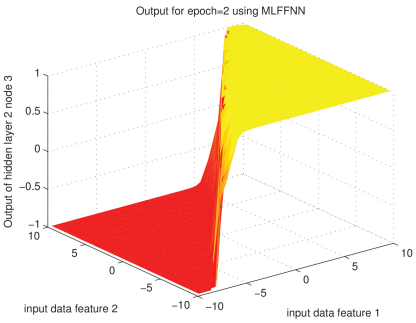
## No.of epochs=2

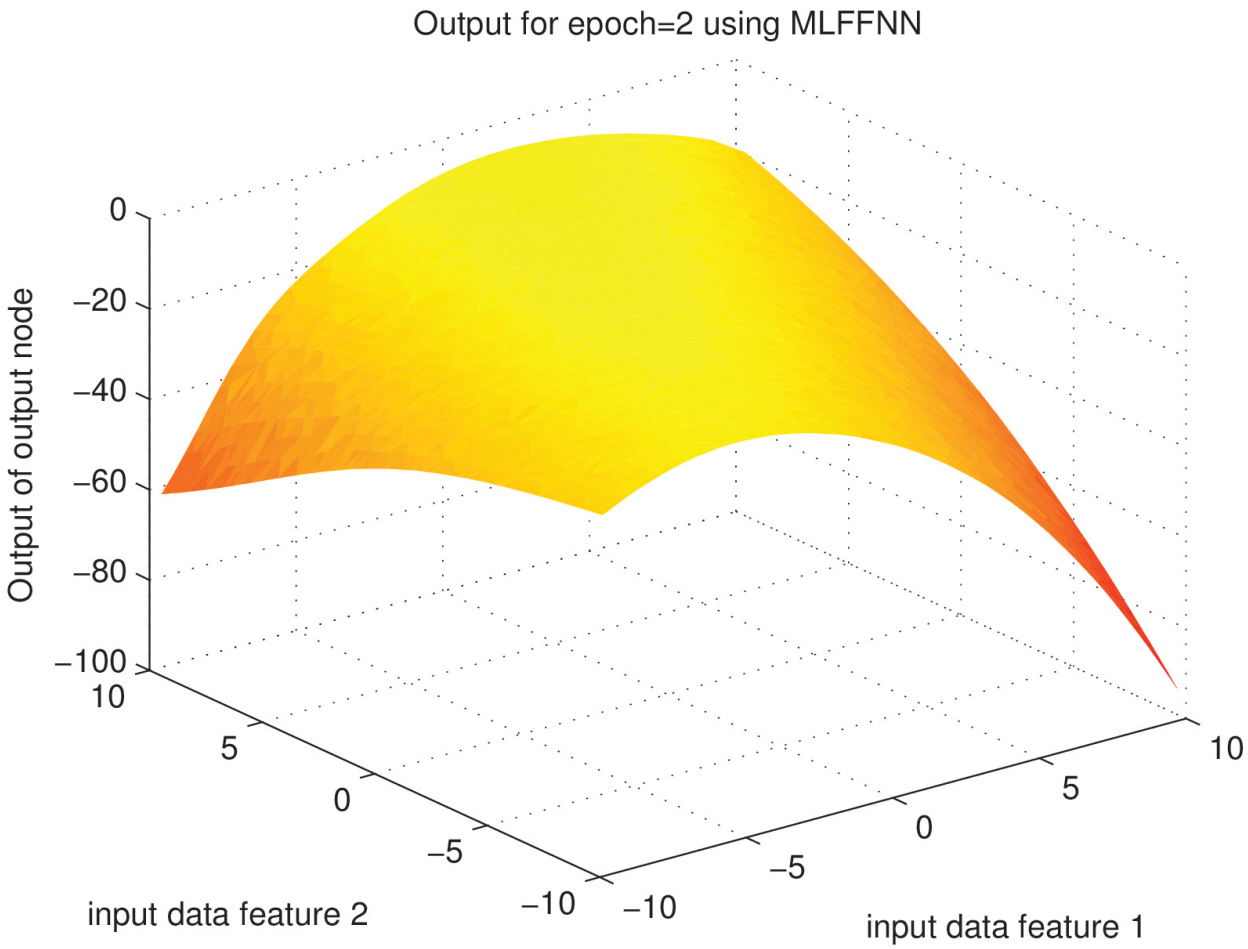


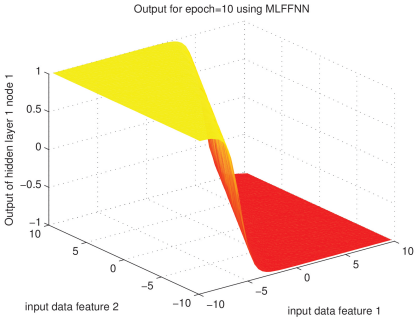



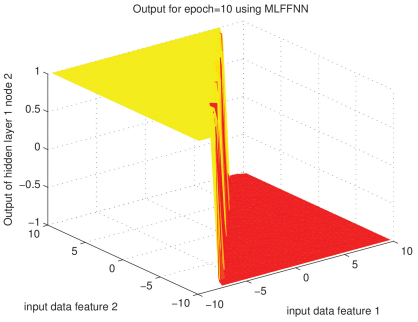



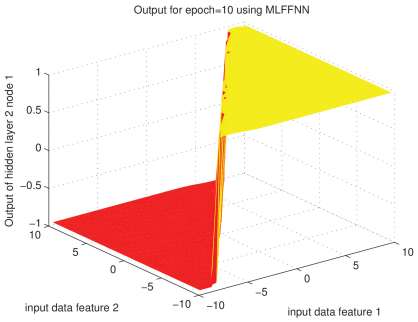



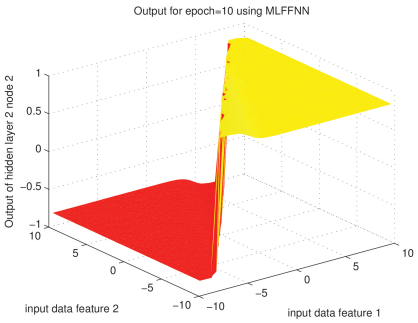

Output for epoch=2 using MLFFNN
Output of output node
0
−20
−40
−60
−80
−100
10
5
0
−5
−10
input data feature 2
−10
−5
0
5
10
input data feature 1

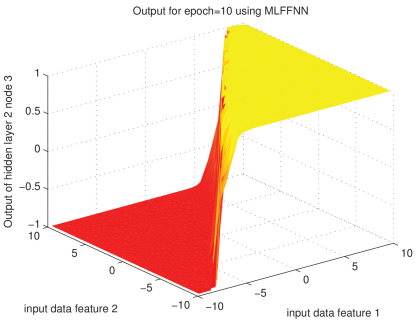
## No.of epochs=10

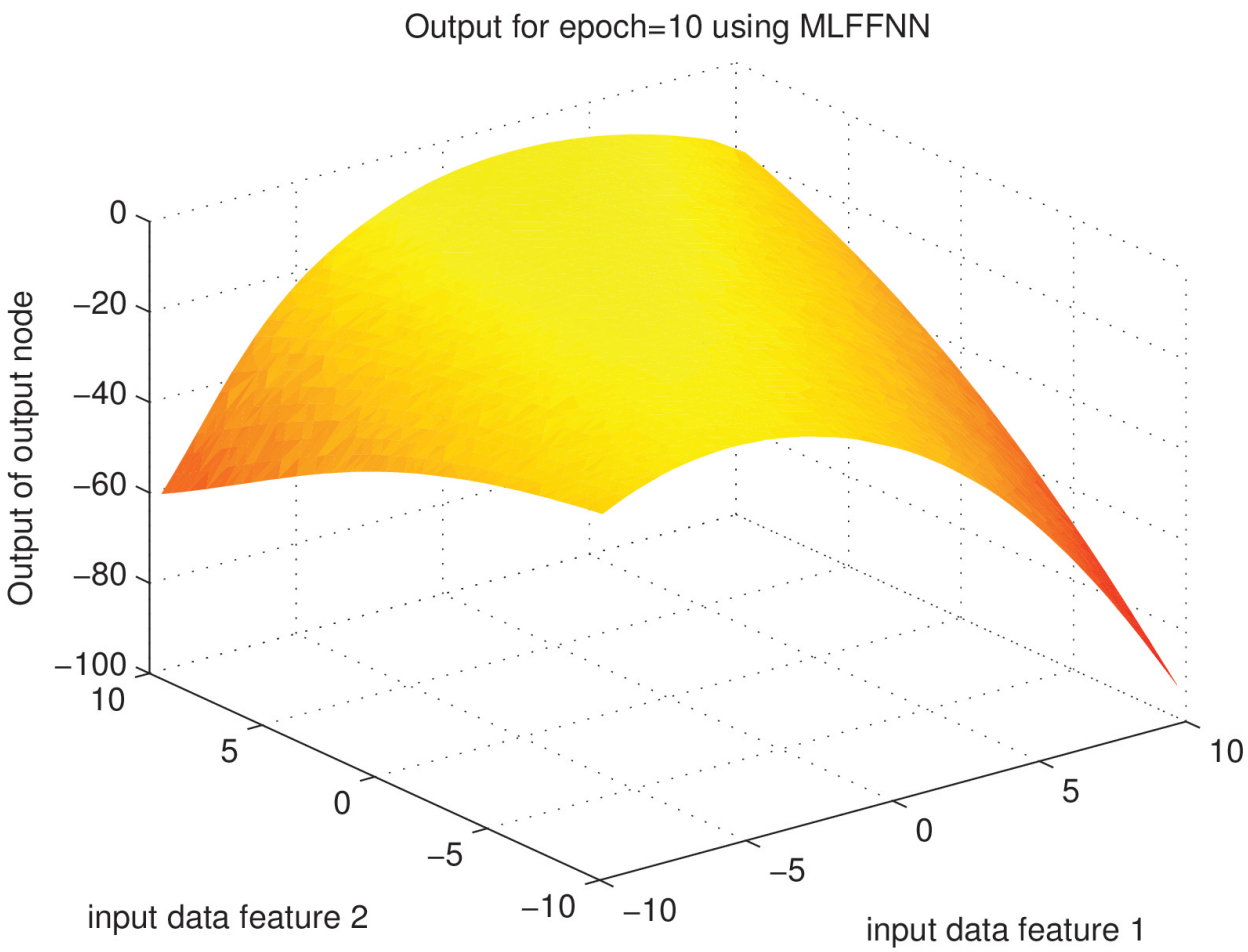


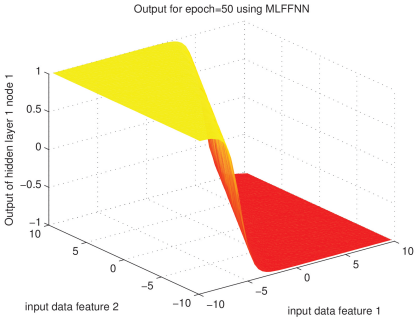



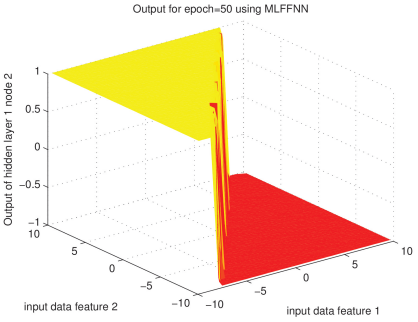



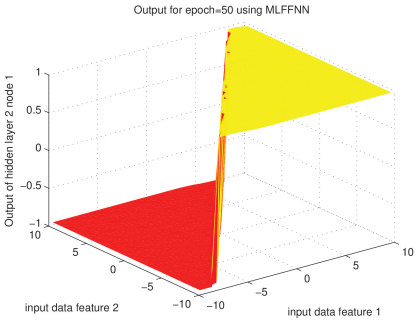



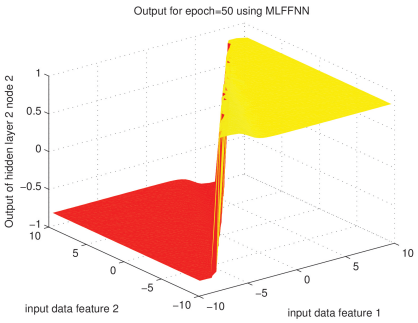

Output for epoch=10 using MLFFNN
Output of output node
0
−20
−40
−60
−80
−100
10
5
0
−5
−10
input data feature 2
−10
−5
0
5
10
input data feature 1

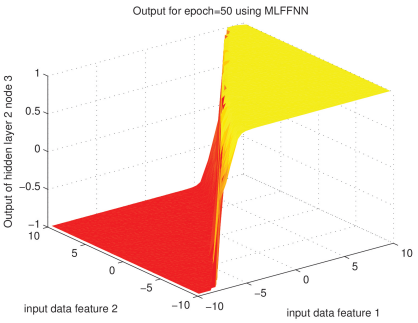
## No. of epochs=50

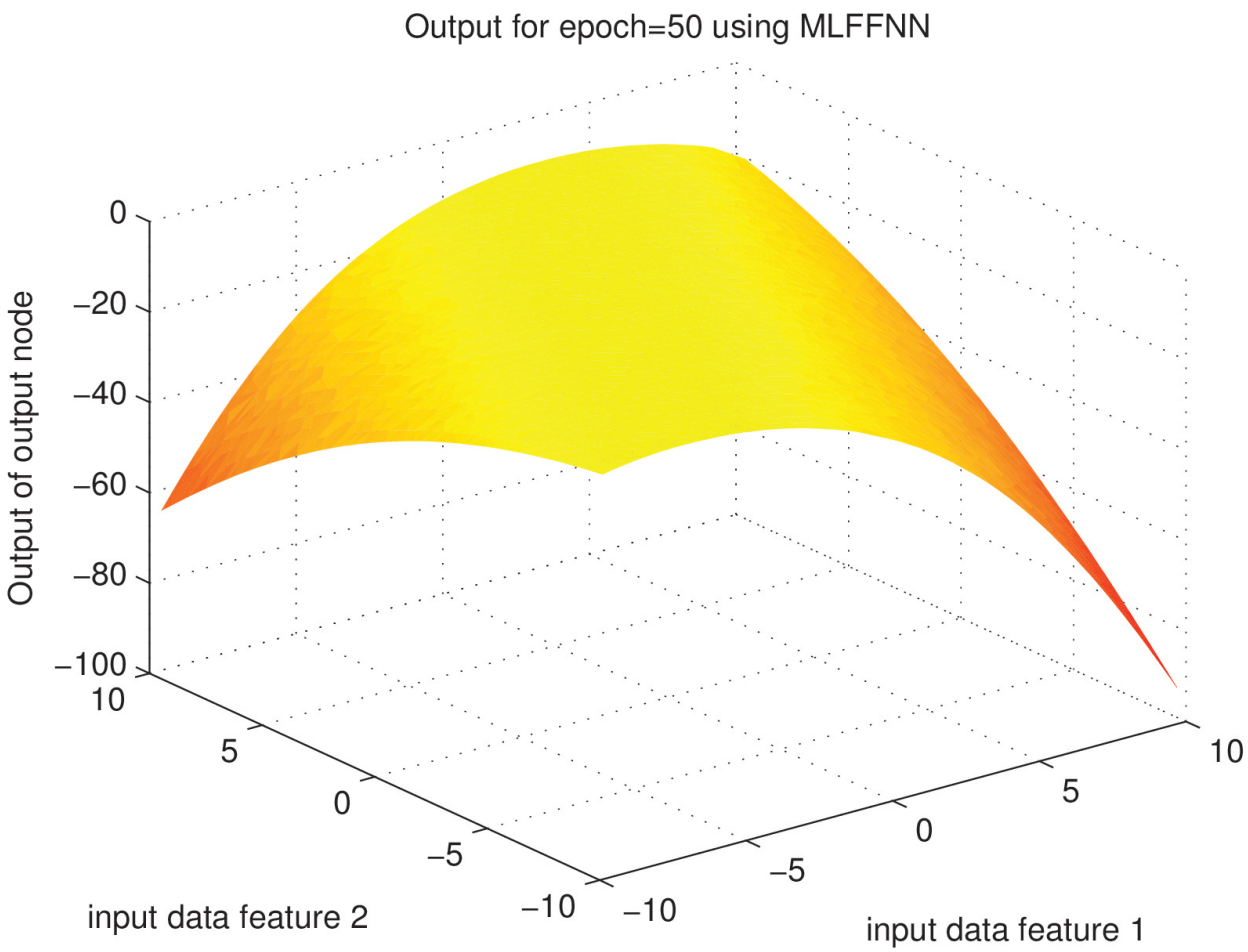


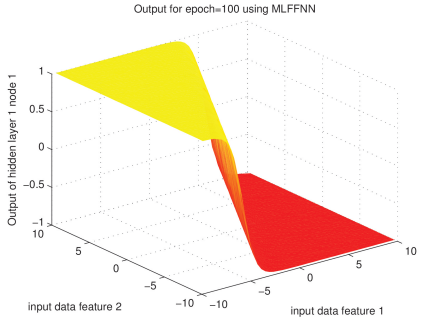



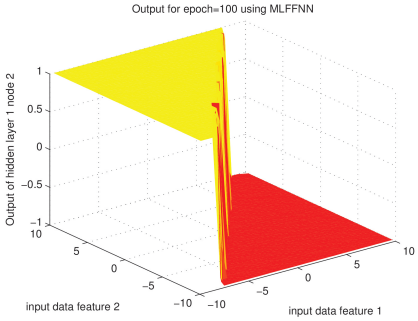



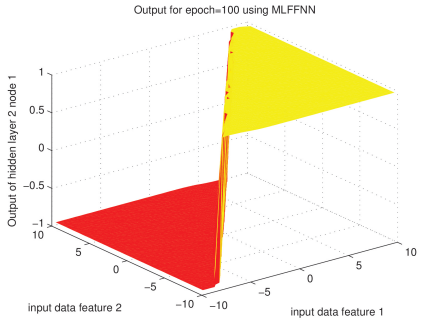



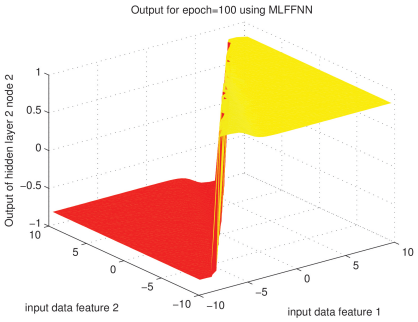

Output for epoch=50 using MLFFNN
Output of output node
0
−20
−40
−60
−80
−100
10
5
0
−5
−10
input data feature 2
−10
−5
0
5
10
input data feature 1

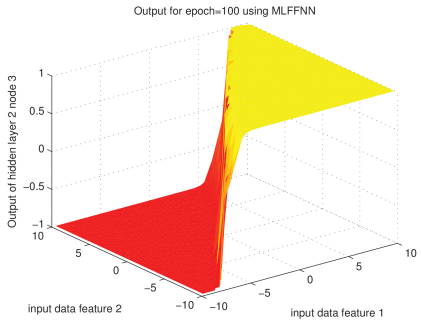
No.of epochs=100

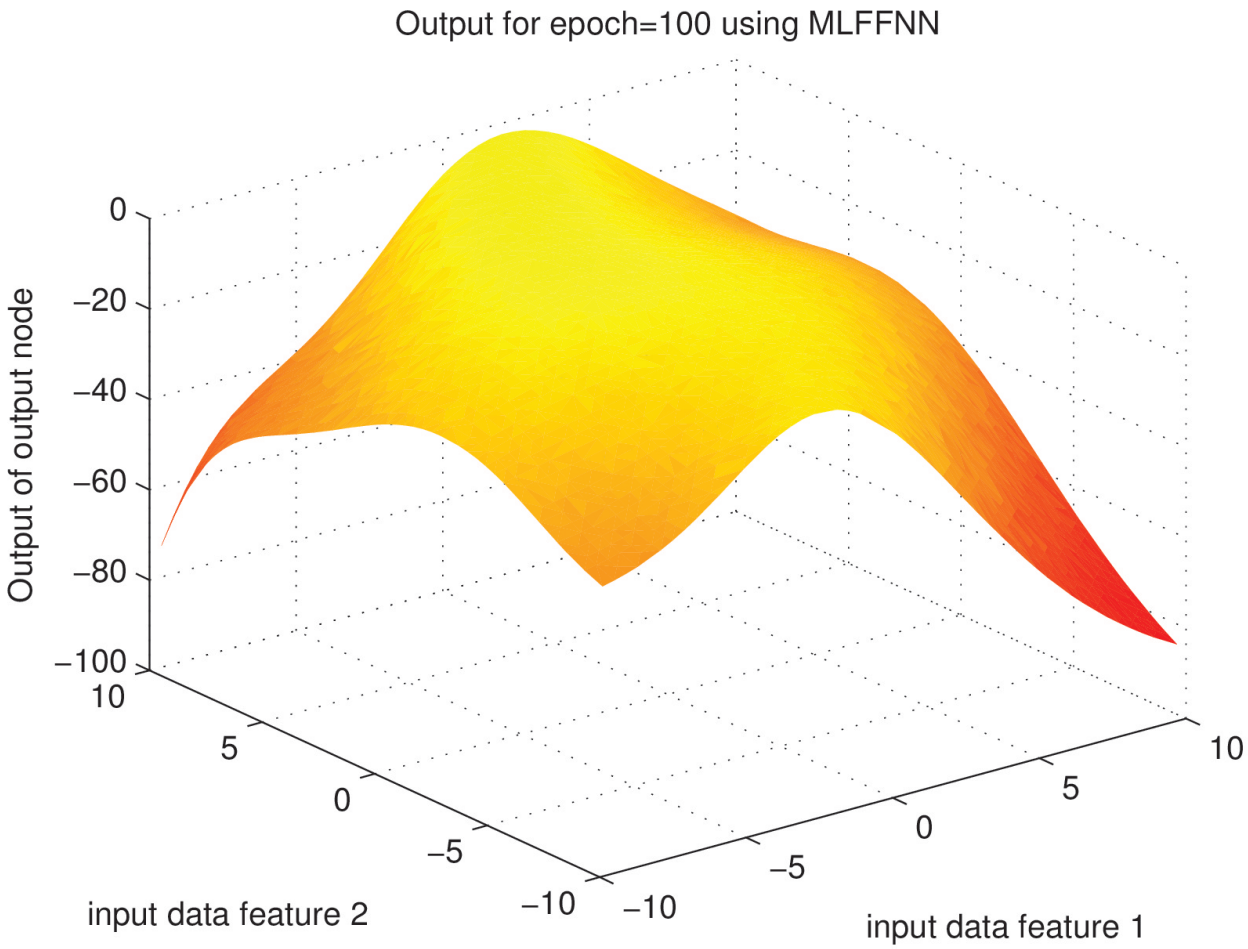


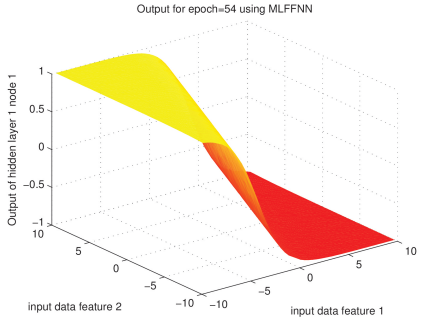



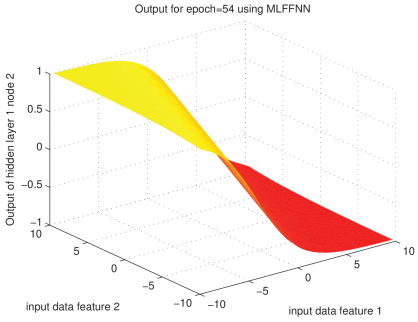



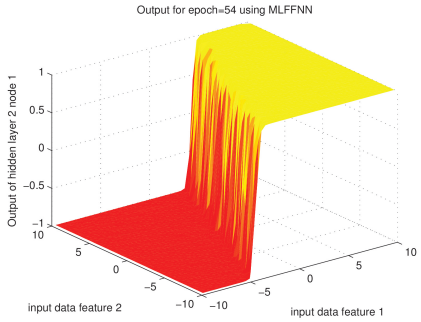



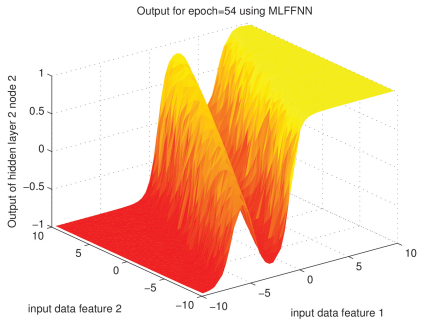

Output for epoch=100 using MLFFNN
Output of output node
0
−20
−40
−60
−80
−100
10
5
0
−5
−10
input data feature 2
−10
−5
0
5
10
input data feature 1

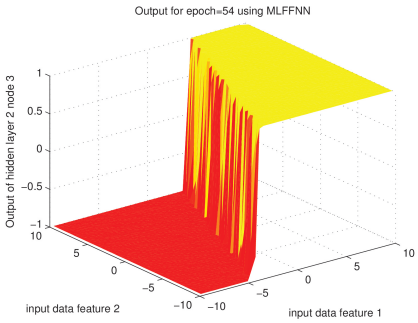
## Surface plots after completion/convergence



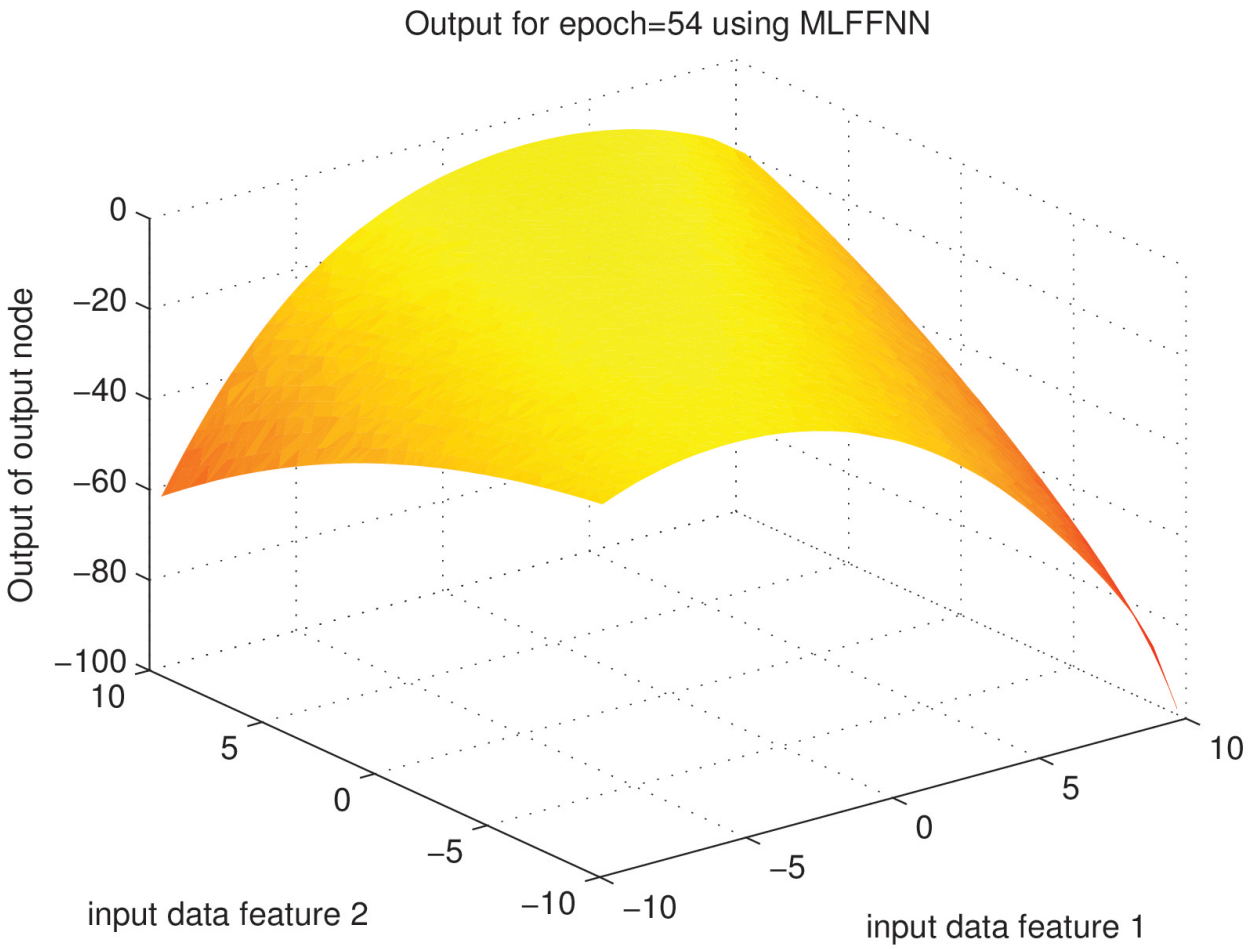




We observe that each of the surface plots is S shaped,i.e each of the hidden nodes are non-linearly dependent on inputs or on hidden nodes of previous layer


Output for epoch=54 using MLFFNN
Output of output node
0
−20
−40
−60
−80
−100
10
5
0
−5
−10
input data feature 2
−10
−5
0
5
10
input data feature 1

## 4.2 MLFFNN for multivariate dataset

### 4.2.1 Choosing best configuration

It is observed that for given number of nodes in first hidden layer,the minimum error is observed to be nearly same.Choosing n1=2
We see that minimum error is observed at n2=2.

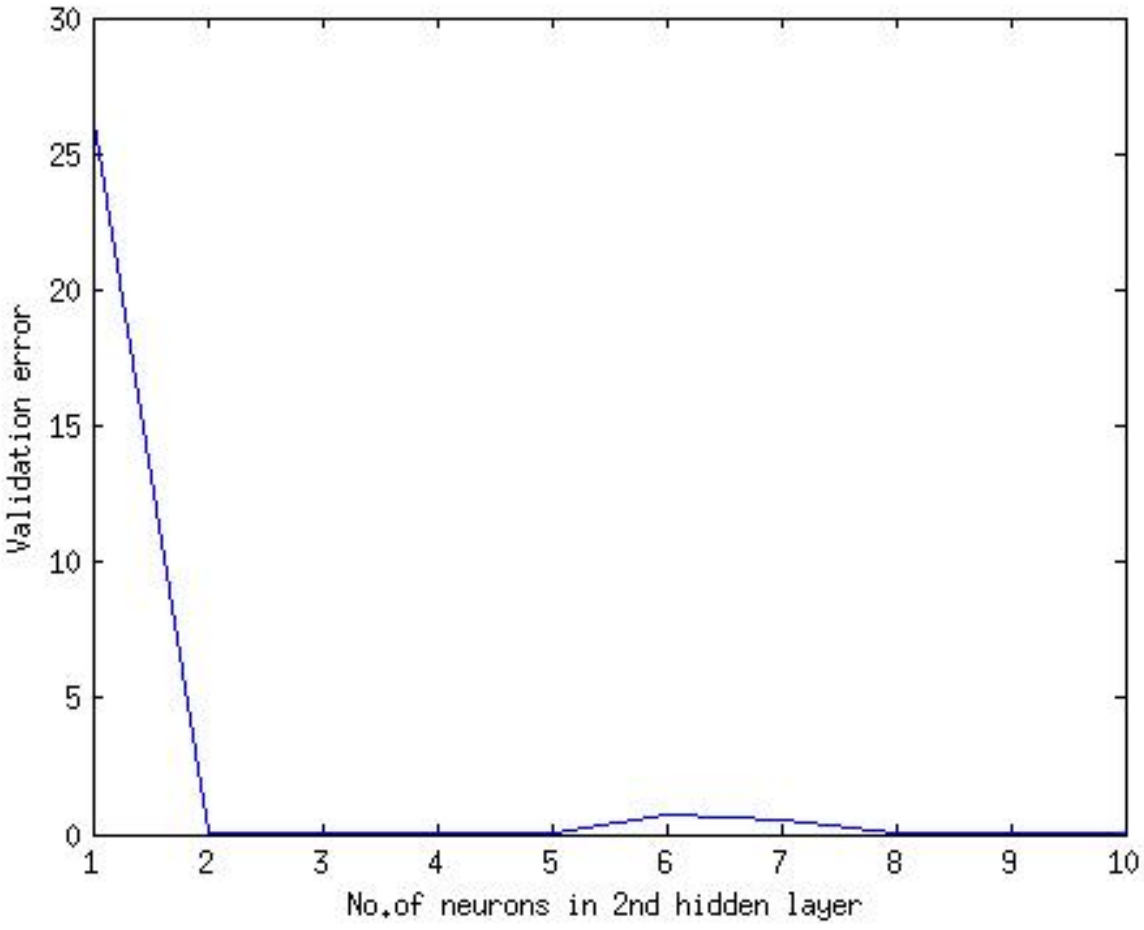



Figure 59: Validation error for different number of neurons in hidden layer 2 for fixed number of neurons in hidden layer 1,here n1=2, n2= number of neurons in hidden layer 2

Hence best configuration is n1=2, n2=2

### 4.2.2 Scatter plot

For the best configuration of neural network i.e n1=2, n2=2,the scatter plot is observed to

be

We observe that the fit almost coincides with y=x line. The validation error for above

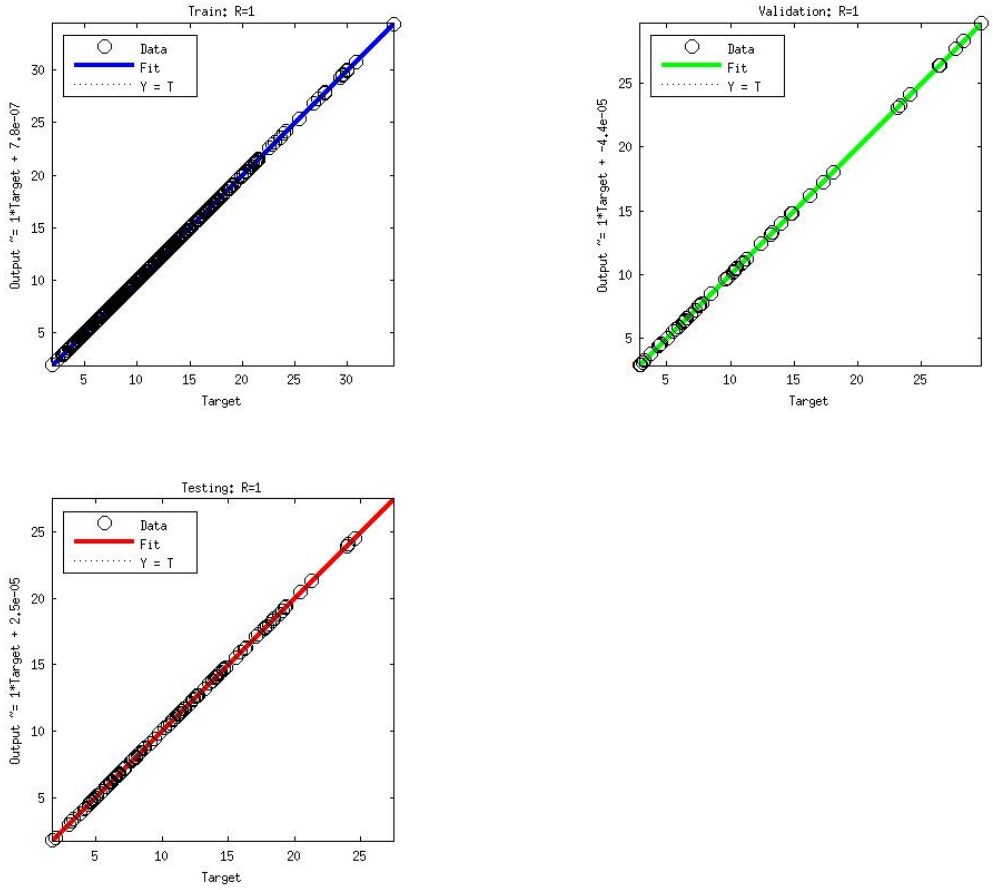



Figure 60: Scatter plot of model output vs target output for n1=2,n2=2

configuration is also very small($10^-7$,nearly 0),we conclude that MLFFNN with 2 hidden layers is sufficient to approximate housing model which is real world data.

## 5. Gaussian Basis Function

Gaussian basis functions for the regression model are chosen using K-means clustering. Quadratic regularization and Generalized RBF are

## 5.1 N = 20

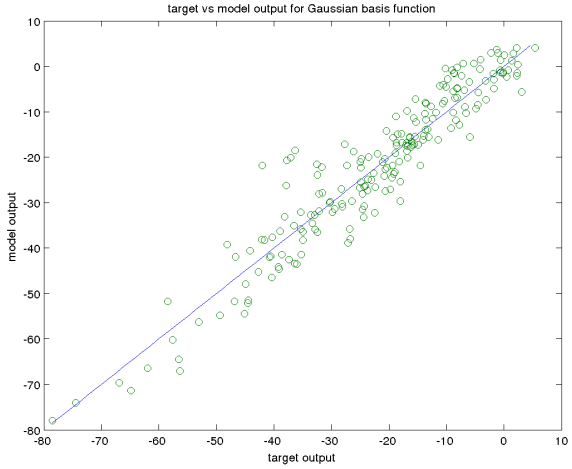



Figure 61: Scatter plot for test data

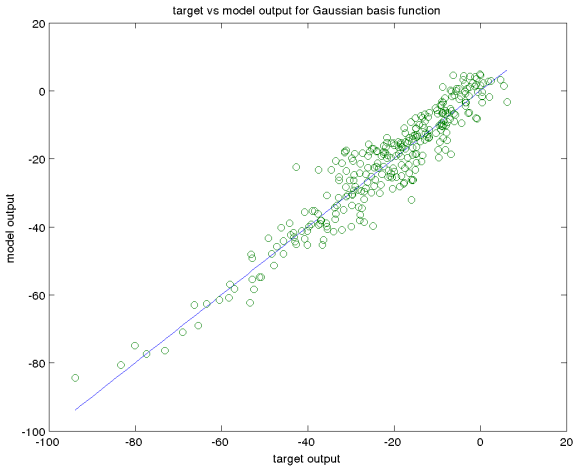



Figure 62: Scatter plot for Validation data

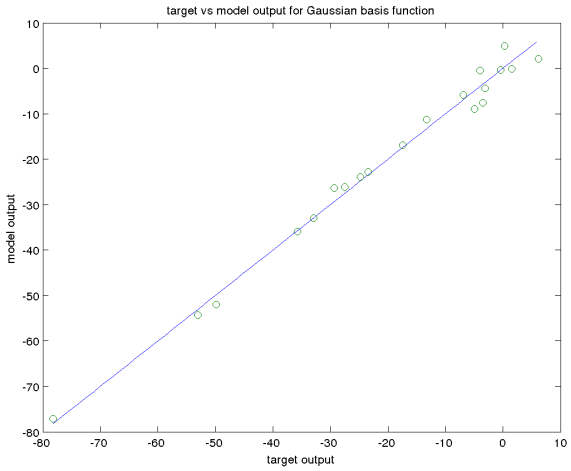



Figure 63: Scatter plot for Train data

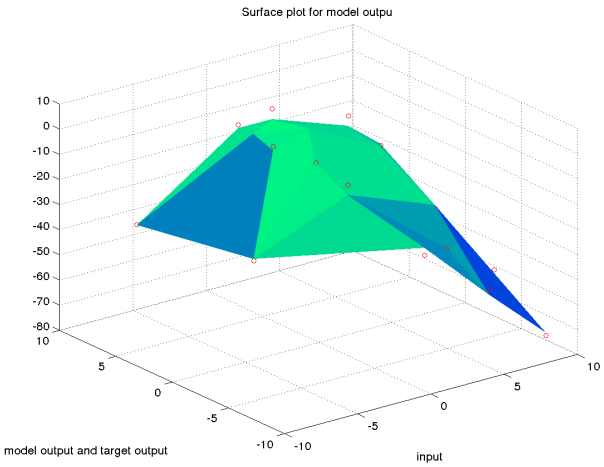



Figure 64: Model output and Target output, lamda = 0, complexity = 11

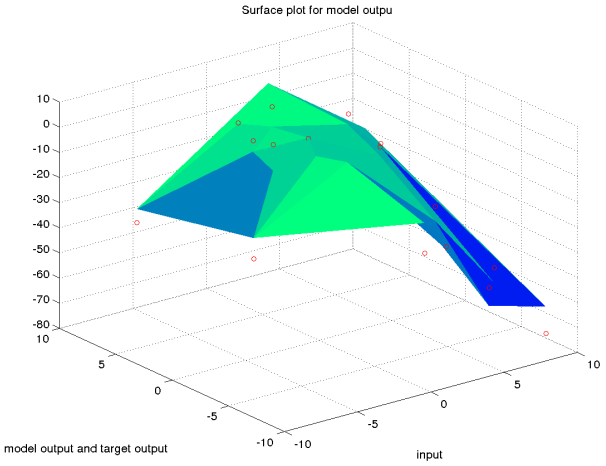



Figure 65: Model output and Target output, lamda = 0, complexity = 9

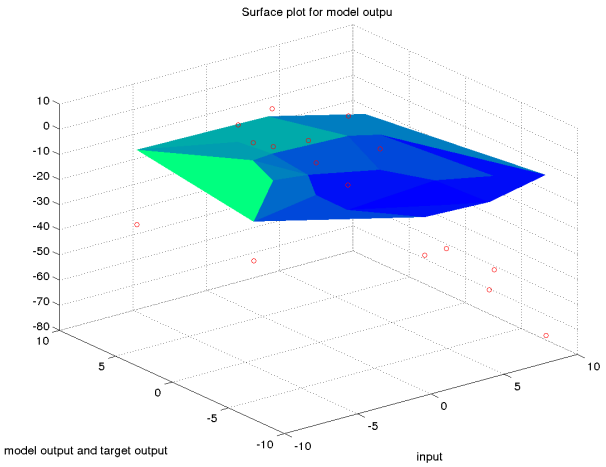



Figure 66: Model output and Target output, lamda = 0, complexity = 9

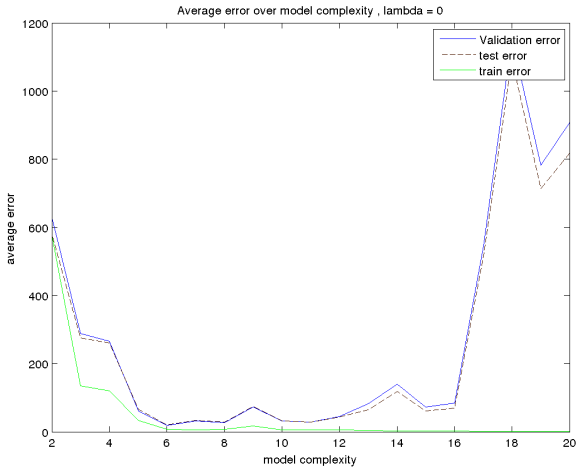



Figure 67: Error for different complexities of model

From the above graph we can conclude that best complexity would be around n = 11.(Running the model several times, had fluctuations in values while n = 11 had minimum in most runs)

With quadratic regularization:

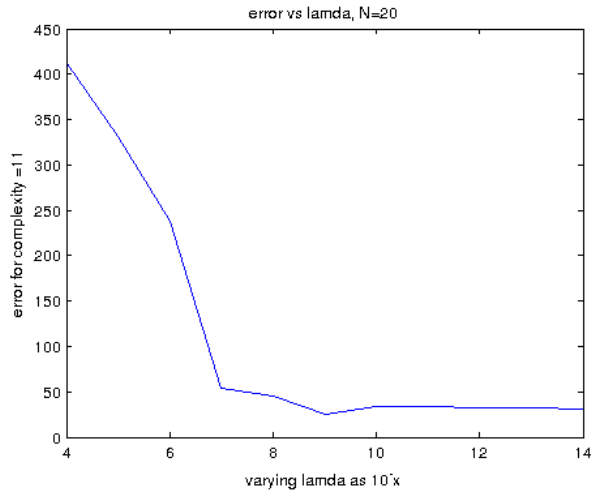



Figure 68: Error for different lamda values

We see that best solution is attained for lamda = $10^{-8}$.

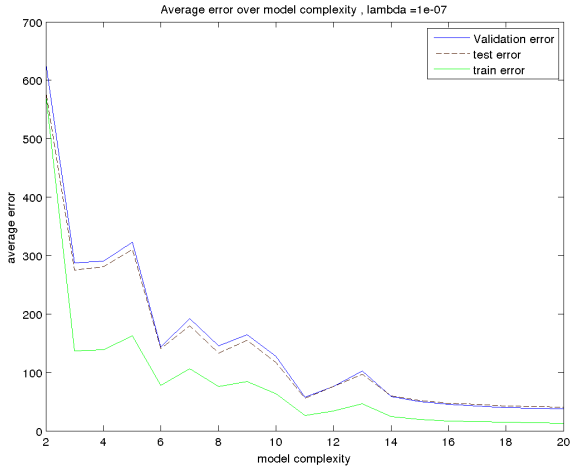



Figure 69: Error for different lamda values

Though there is not much of a decrease in error with regularization, we can see that the model has become quite general since the graphs for validation and test are very close.

## 5.2 N=100

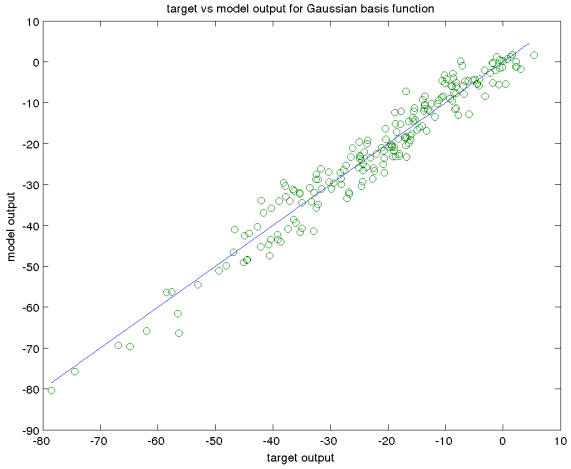



Figure 70: Scatter plot for test data

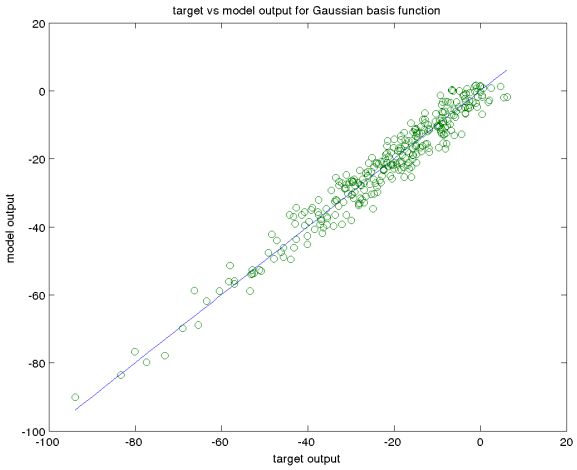



Figure 71: Scatter plot for Validation data

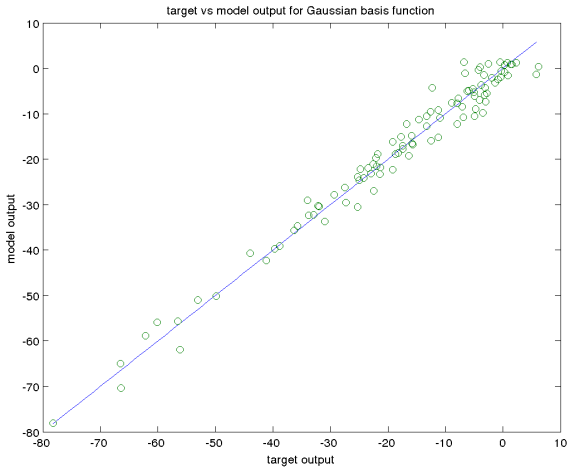



Figure 72: Scatter plot for Train data

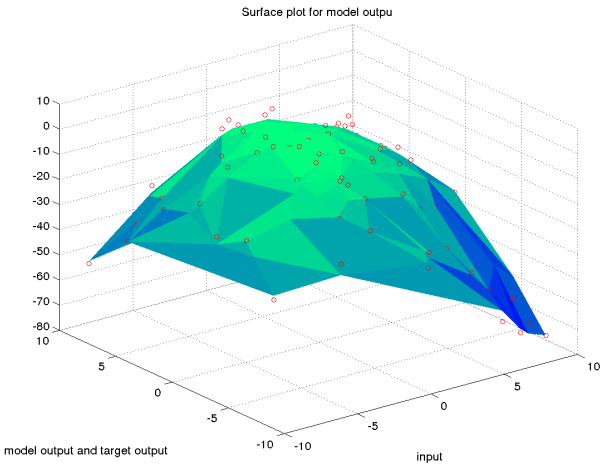



Figure 73: Model output and Target output, lamda = 0, complexity = 11

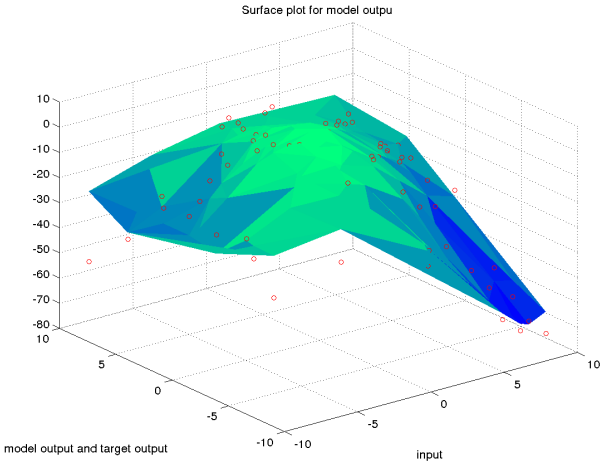



Figure 74: Model output and Target output, lamda = 0, complexity = 5

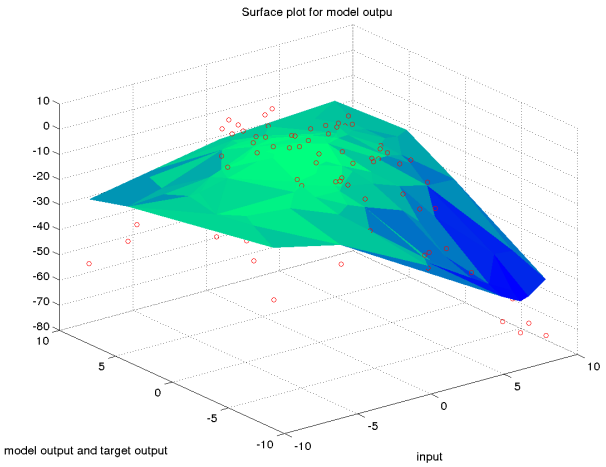



Figure 75: Model output and Target output, lamda = $10^{-6}$, complexity = 11

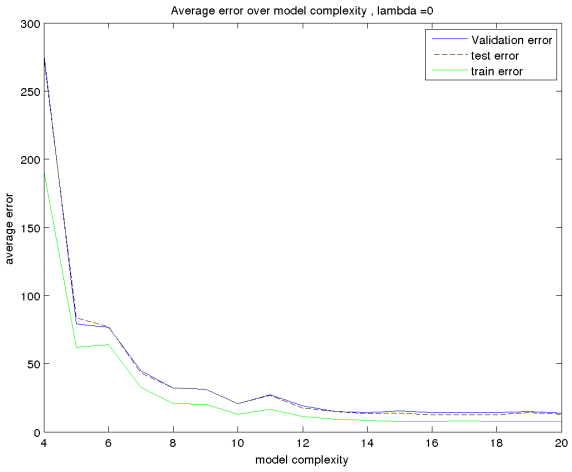



Figure 76: Train, Validation, and test error for different model complexity.

Running the model for different model complexities gave n = 11 to be best without regularization.

With quadratic regularization:

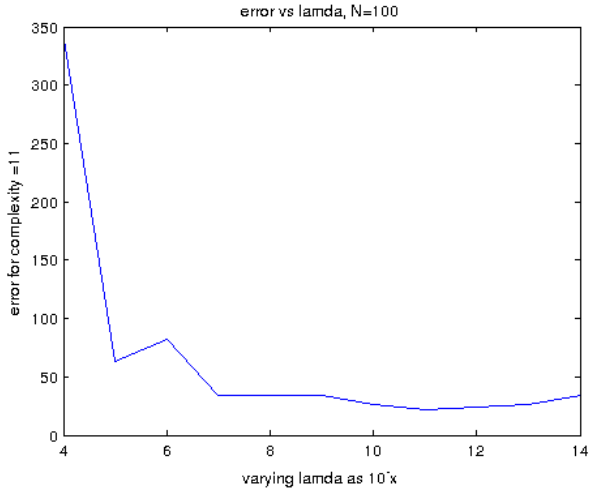



Figure 77: Error for different values of lamda , for n = 11.

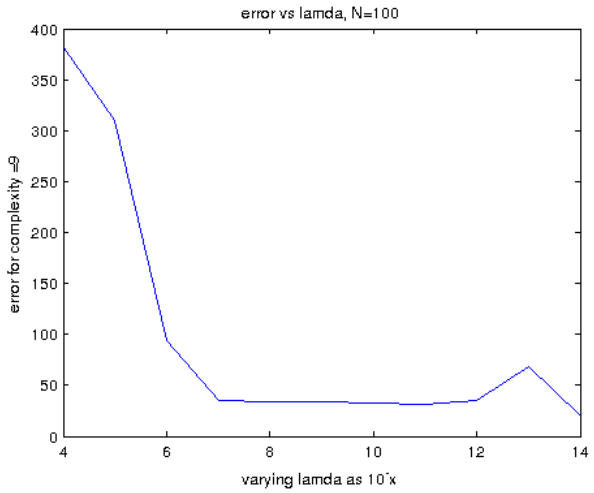



Figure 78: Error for different values of lamda

The above plots were plotted for n = 9, 11. We can see that for both values of complexity there is not much change in error for regularization. Best model is for lamda = $10^{-7}$.

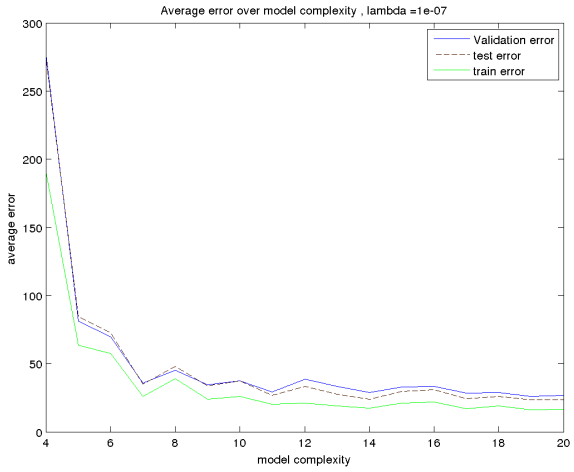



Figure 79: Error for different complexities for chosen lamda = $= 10^{-7}$.

## 5.3 N=1000

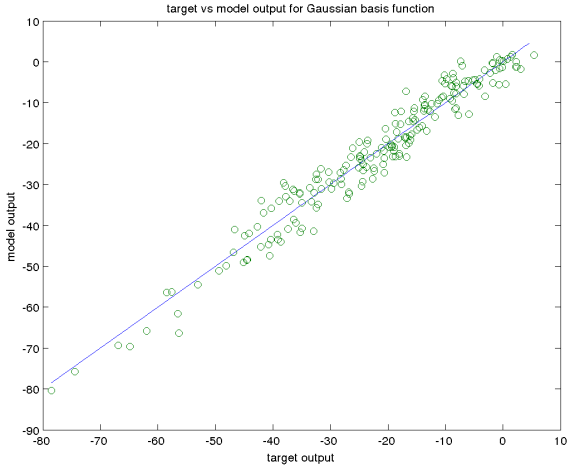



Figure 80: Scatter plot for test data

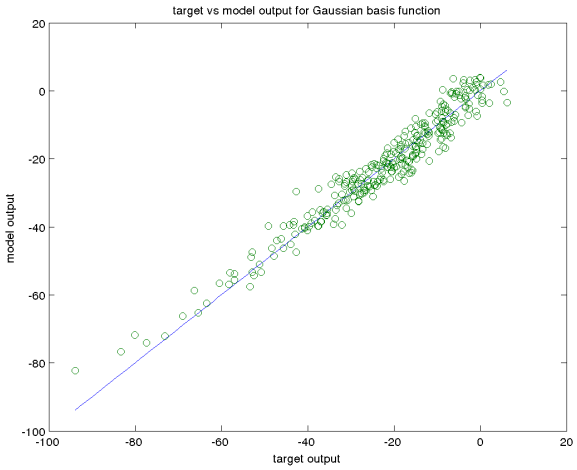



Figure 81: Scatter plot for Validation data

Regularization did not make much effect as we can see the errors of Train, Validation and Test are converging.

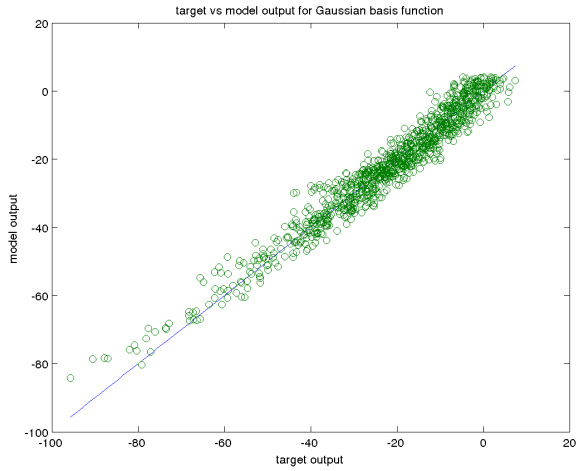



Figure 82: Scatter plot for Train data

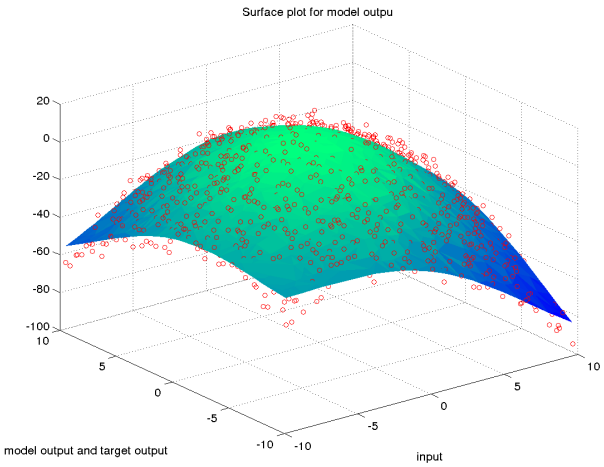



Figure 83: Model output and Target output, lamda = 0, complexity = 9

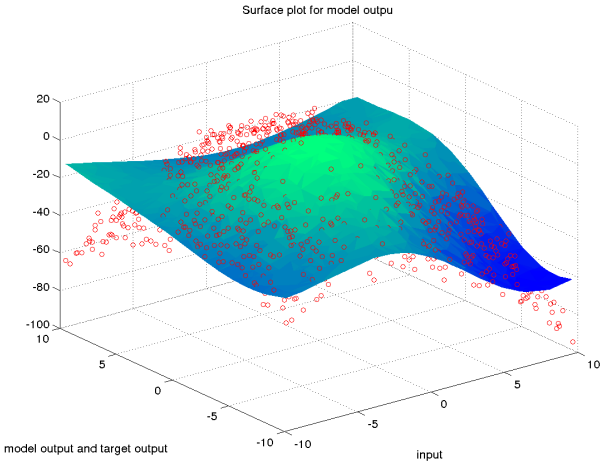



Figure 84: Model output and Target output, lamda = 0, complexity = 6

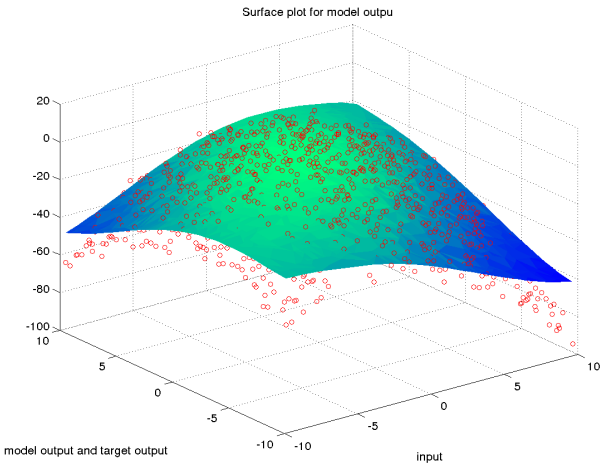



Figure 85: Model output and Target output, lamda = $10^{-6}$, complexity = 9

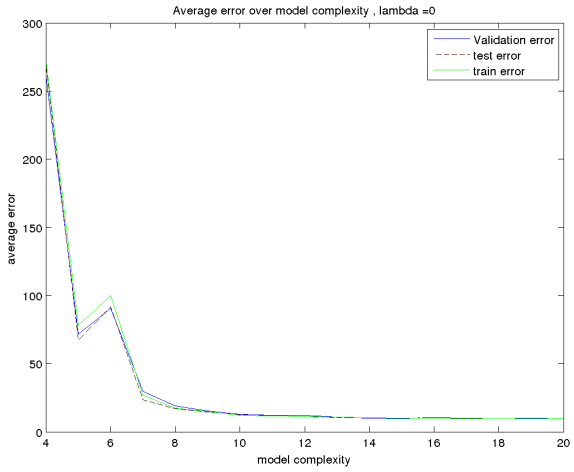



Figure 86: Error for different complexities of the model

## 5.4 Dataset 3

Scatter plots:

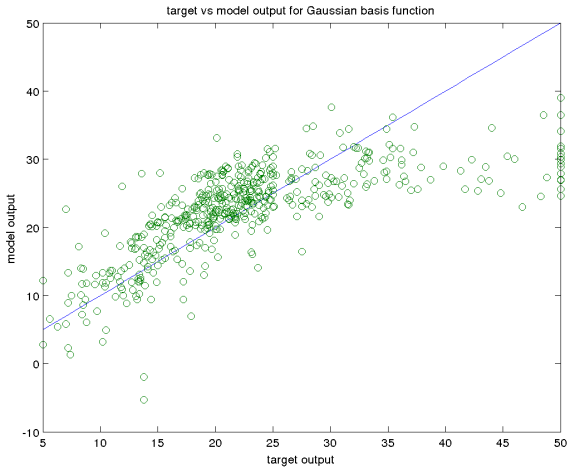



Figure 87: Model output and Target output

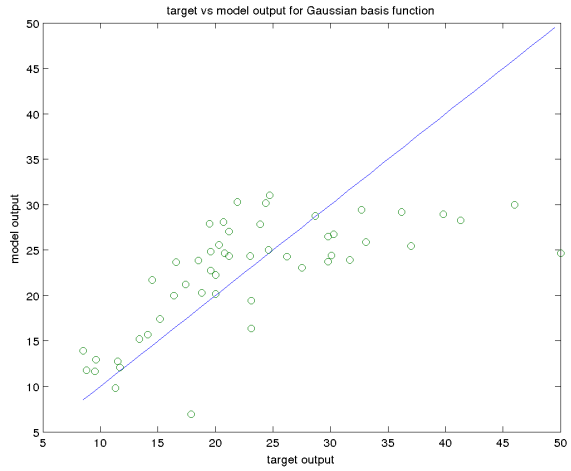



Figure 88: Model output and Target output

On applying quadratic Smoothing: From the above two plots the minimum error occurs for n = 16 complexity, and lamda = $10^{-79}$.

Plot for regularization with lamda = $10^{-79}$.

Regularization did not make much effect in the error values, the error graphs are similar for Train, Test and Validation data.

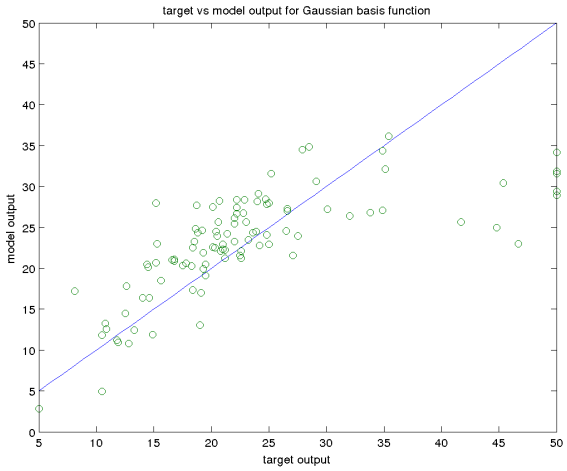



Figure 89: Model output and Target output

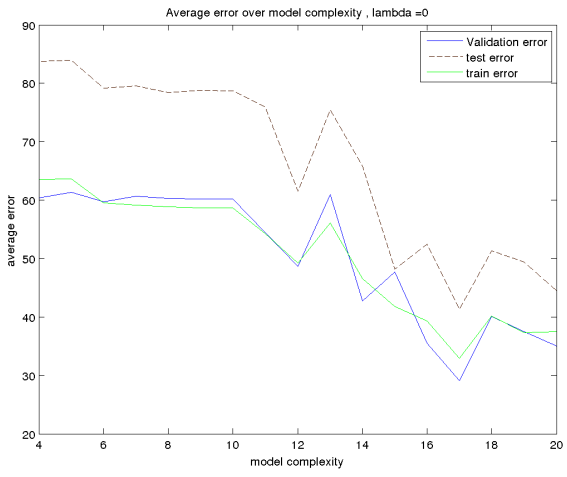



Figure 90: Error for different complexities of model, lamda = 0

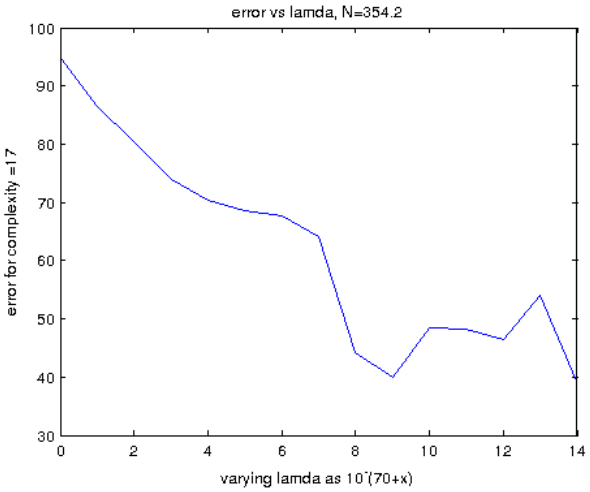



Figure 91: Complexity plot

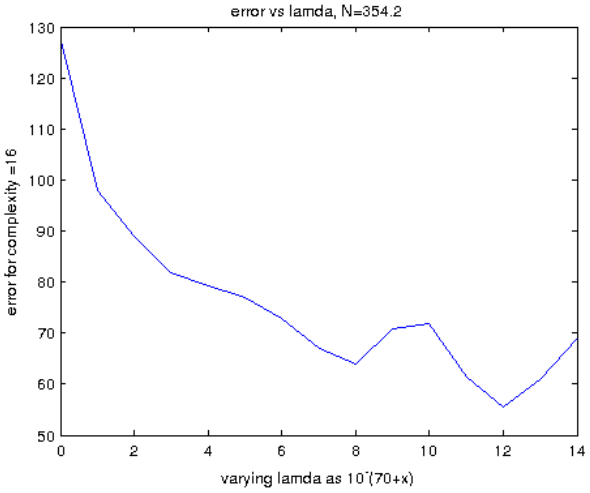



Figure 92: Complexity plot

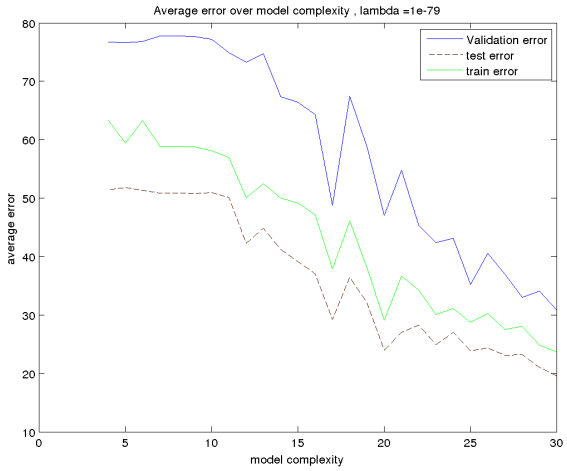



Figure 93: Complexity plot

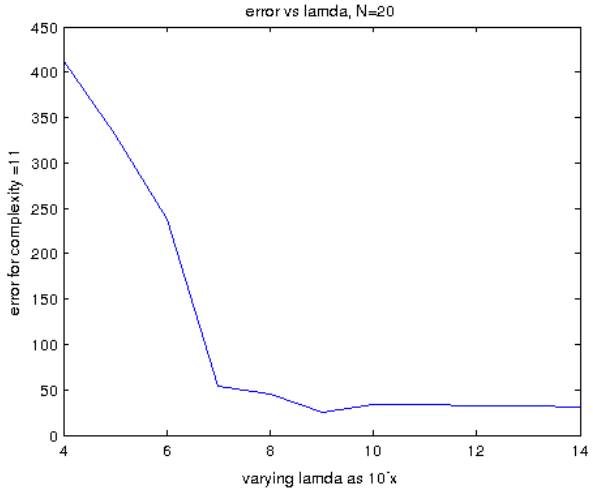



Figure 94: lamda for N = 20, n = 11

# 6. Gaussian Basis with Generalized RBF:

## 6.1 N=20

From lamda = 0, we get best complex model to be for n = 11. Plotting errors for different lamda values for n = 11, we get least error for lamda = $10^{-8}$.

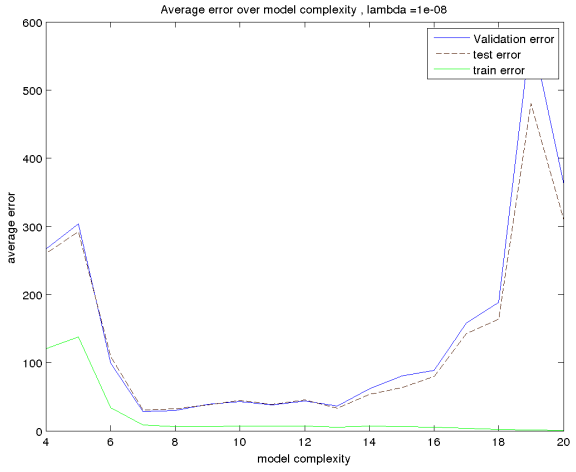



Figure 95: lamda for N = 20, lamda = $10^{-08}$

The best model complexity for lamda = $10^{-8}$, comes out close to 7.
On regularization the model complexity has reduced.

## 6.2 N=100

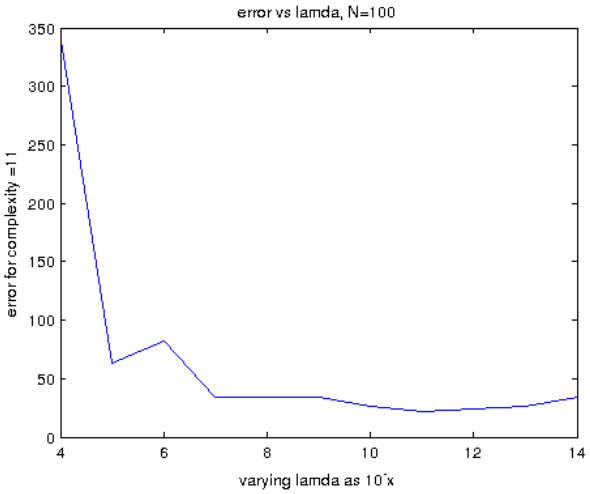



Figure 96: Lamda Vs Error for N = 100, n = 11

From lamda = 0, we get best complex model to be for n = 11. Plotting errors for different lamda values for n = 11, we get least error for lamda = $10^{-7}$.

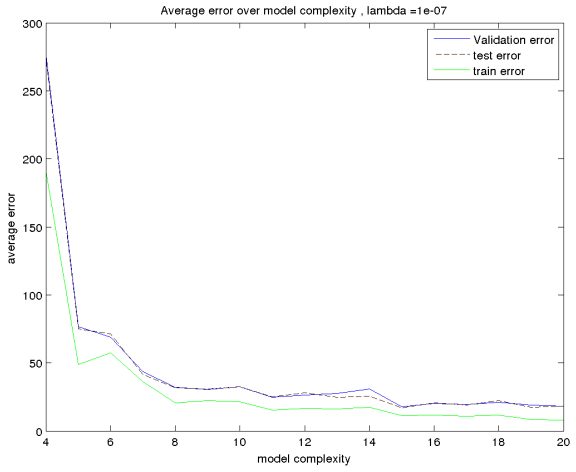



Figure 97: Model output and Target output

The best model complexity for lamda = $10^{-7}$, comes out close to 8.

## 6.3 N = 1000

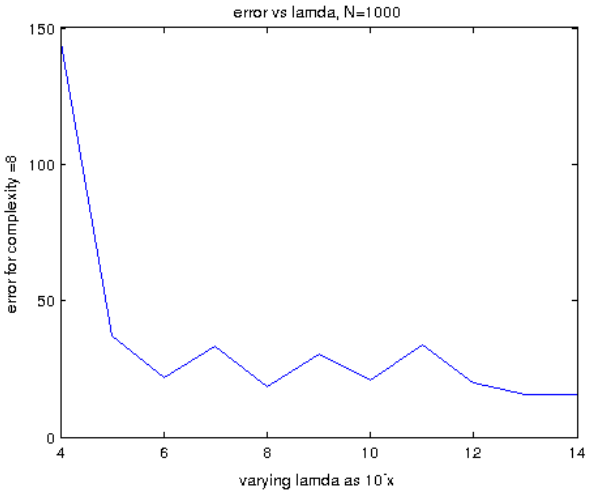



Figure 98: Lamda Vs Error for N = 1000, n = 8

From lamda = 0, we get best complex model to be for n = 8. Plotting errors for different lamda values for n = 8, we get least error for lamda = $10^{-7}$.

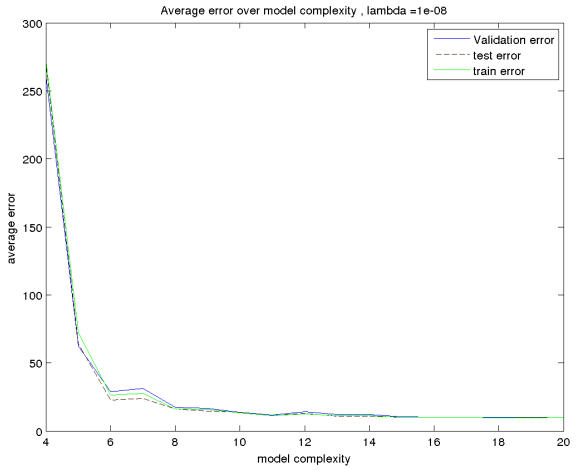



Figure 99: Model output and Target output

The best model complexity for lamda = $10^{-8}$, comes out close to 8.

# 7. Dataset 3

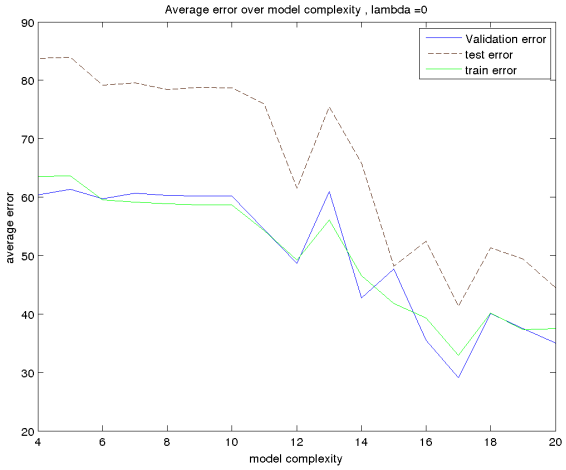



Figure 100: Model output and Target output

Using n = 17 as model complexity, varying lamda:

So we choose lamda in the range of $10^{-70}$. Using lamda = $10^{-74}$, we get

The graphs of Train, Test and Validation are quite close to each other after regularization.

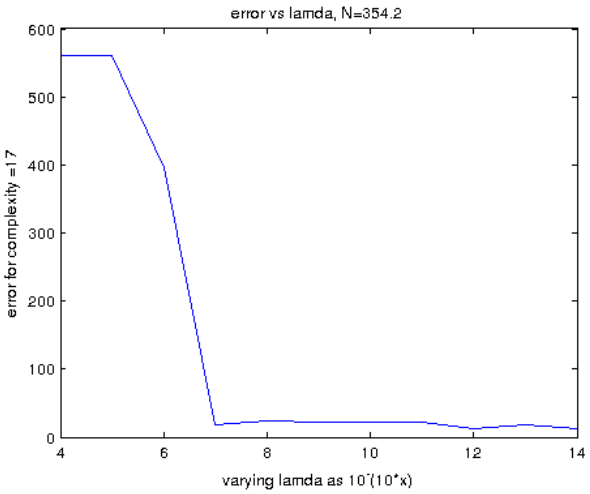



Figure 101: Varying lamda for n = 17 model complexity

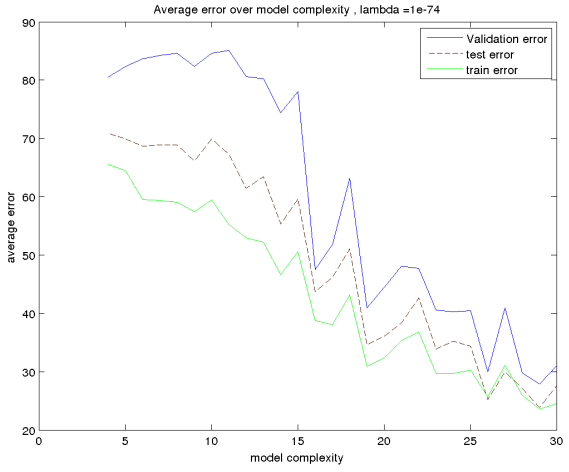



Figure 102: Complexity plot