# CS6011 Kernel Methods for Pattern Analysis
# Assignment-1
# (Classification (MLFFNN) & Regression (Polynomial fit, MLFFNN))

Arulkumar S (CS15S023), Divya Saglani (CS15M041), Nitish (CS15S028)

3rd March 2016

## 1 Introduction

In this programming assignment, we concentrate on understanding,

- Classification using Multi-layer feedforward Neural networks
- Function approximation using Polynomial regression, Gaussian kernel, RBF kernels, Multi-layer feedforward Neural Networks

## 2 Classification

- In classification problems, the neural networks is trained using Supervised mode. The training examples are given in the format ($x_i$, $w_i$) where $x_i$ is the feature vector & $w_i$ is the class label.
- Every example is considered to be belonging to a single class. i.e, the output expected is to be of discrete in nature.
- for example, if there are three classes, class-1 will be represented as [1 0 0], class-2 will be represented as [0 1 0], class-3 will be represented as [0 0 1].

### 2.1 Procedure

- Generally, to classify the inputs, Final layer of the neural network will have the number of nodes equal to the total unique number of classes that the given dataset has.
- Each hidden layer is assigned of different number of nodes and the performance of neural network is analysed using the accuracy of classification in the validation data.
- In our experiments, we consider the number of hidden layer nodes from 2 to 20 (incremented by 2 at each step).
- The activation function of hidden nodes are assigned as **tansig non-linearity**.
  tansig(a) = $(2 * sigmoid(a)) - 1 = tanh(a)$
- The final layers nodes will have **soft-max activation** function, as the classification output is expected to be a Categorical Probability likelihood (or confidence score from 0 to 1) of the example to be belonging to a particular class.
  softmax$_i$ = $\frac{e^{out_i}}{\sum_{j=1}^{N} e^{out_j}}$ where N is the number of output nodes, $out_i$ is the output from $i^{th}$ output node.
  The example will be assigned the class to which it has the highest probability (or) score.

- The classification loss function of the final layer is considered to be 'Cross-Entropy loss'. Since, In classification, we need to emphasize on the mis-classification rather than the Mean-error (which will be important incase of regression), we use Cross-entropy loss.
  Cross entropy is defined as $-\sum_{i=1}^{N} t_i * log(out_i)$ where N is the number of output nodes, $out_i$ is the output from $i^{th}$ output node.
- The Hidden layer configuration of neural network which is performing best in Validation data is selected and all the hidden layer outputs, final layers outputs are plotted after training for 1,2,10,50,100 epochs.
- The best model is selected based on the Miss percentage on validation data (the less the miss percentage, better the model is) & the number of nodes in the hidden layer.

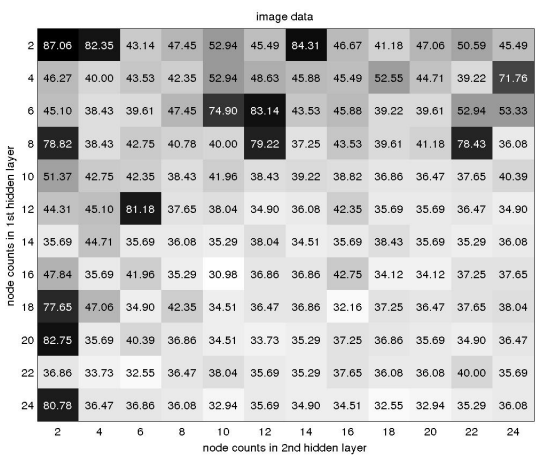
## 2.2 Image dataset

- The images that are given for classification are
  - class1 = hammock (284 images = train (199), validation(43), test(42))
  - class2 = airplanes (800 images = train (560), validation(120), test(120))
  - class3 = binoculars (216 images = train (151), validation(33), test(32))
  - class4 = backpack (151 images = train (106), validation(23), test(22))
  - class5 = ladders (238 images = train (167), validation(36), test(35))
- The given dataset is splitted into training (70%), validation (15%) and test (15%) data.
- A neural network with 2 hidden layers is trained by assigning different node counts (2 to 30 : incremented by 4) and best performing model on validation set is chosen for testing and further analysis.

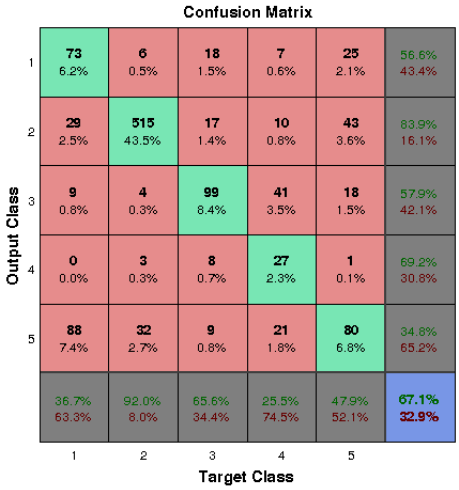
As shown in the plot of Miss percentages, the miss percentage of the model configuration (hidden layer1 = 16, hidden layer2 = 10) on validation data is less than other models. So, it is choosen as the best model.

### 2.2.1 MLFFNN with 2 Hidden layers

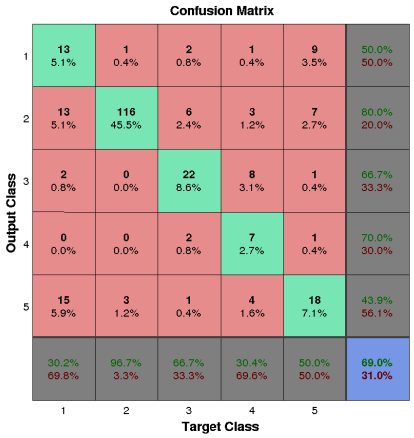
(a) Validation Miss percentage based on model complextites



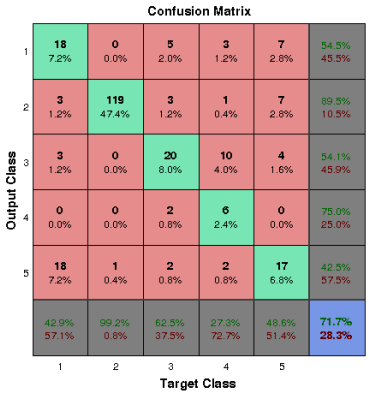

(b) confusion matrix of train-data



(a) confusion matrix of validation-data




(b) confusion matrix of test-data



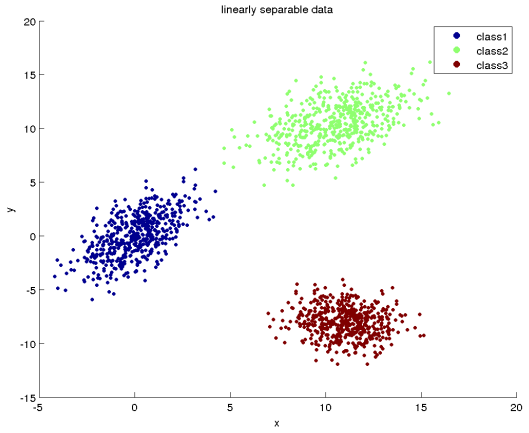

#### 2.2.1.1 Observations

- The classification accuracy of $\tilde{6}9\%$ is achieved on validation data for the model complexity of hiddenlayer-1 nodes = 16, hiddenlayer-2 nodes = 10.
- The miss percentage seems to increase while adding mode nodes in the hidden layers, as shown in the miss percentage table plot.
- We employed min-max normalization in the columns of given features, which did not increase the performance of the trained model.
- Since, the class2 features are more ($\tilde{8}00$ images), the model is able to learn this particular class confidently and the classification accuracy for class2 is 95% on validation data. We are able to infer that having more data is key while training neural networks.
- As the number of features for other classes are less, the neural network model is not able to draw a decision boundary confidently for the other classes.

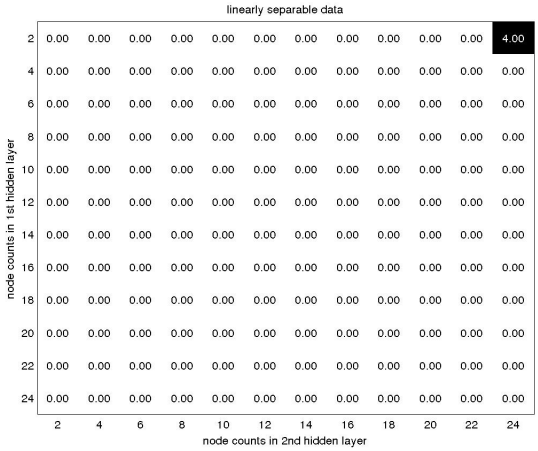
## 2.3 Linearly separable data

The data and the miss percentage obtained for each complexity of the model is shown below. The miss percentage for model complexity is shown as a table plot (The lighter the cell, the better the performance is).
The dataset contains 500 points (train = 250, validation = 150, test = 100) in each of 3 classes.

(a) Plot of the data

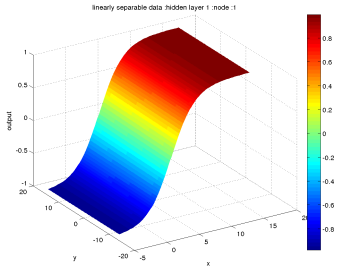
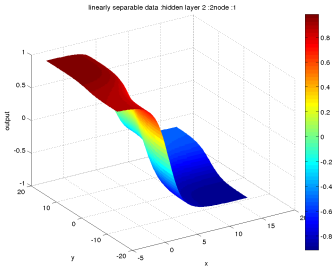
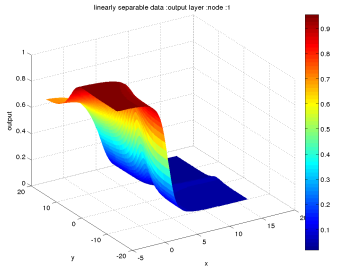
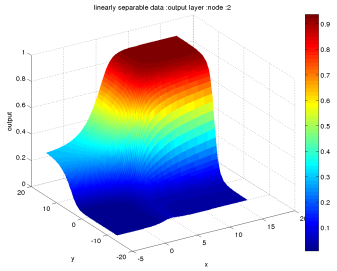
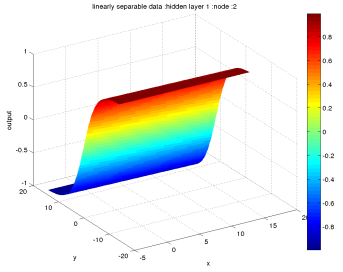
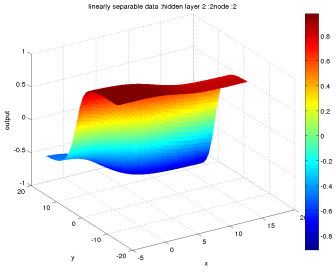
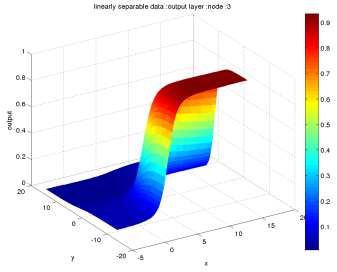



(b) Validation Miss percentage based on model complextites



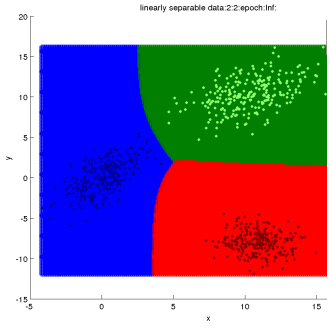

### 2.3.1 MLFFNN with 2 Hidden layers

#### 2.3.1.1 Layer outputs after Training

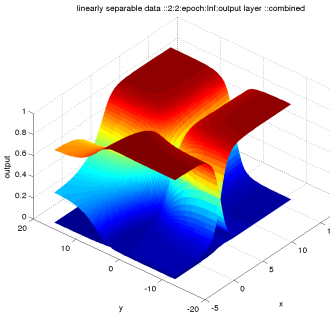
| Hidden layer-1 | Hidden layer-2 | Output layer |
|---|---|---|
|  |  |  |
| | |  |
|  |  |  |

#### 2.3.1.2 Decision boundary & Confusion matrix

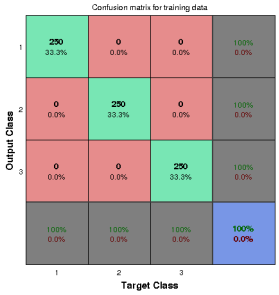
(a) Decision boundary plot



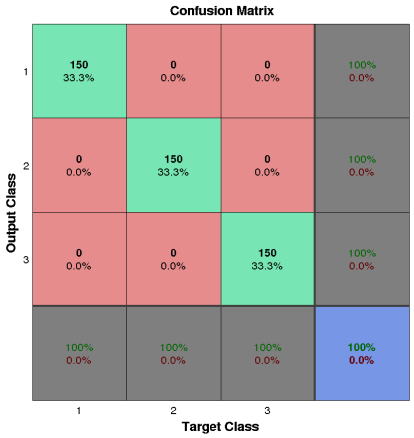

(b) combined output of final layer nodes



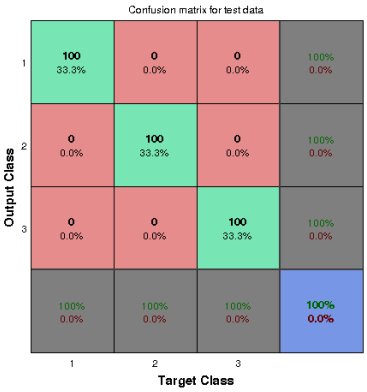

(c) confusion matrix of train data




(a) confusion matrix of validation data




(b) confusion matrix of test data



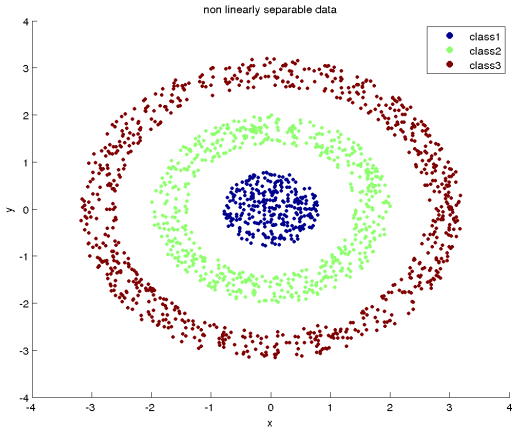

#### 2.3.1.3 Observations

- Since, the dataset is a linearly separable one, the less complex model is able to achieve 100% efficiency.
- As seen in the *Miss percentage image*, almost all of the models are able to give 0% error rate. hence, we choose a simple model with node counts of hidden layer1 = 2, hidden layer2 = 2. This model converges to best performace in 125 epochs.
- As shown in the decision boundary plot, the model is able to easily separate the data. Since, the model hidden layers are made up of *tansig* non-linearities, the decision boundary is a combination of non-linearities from hidden layers.

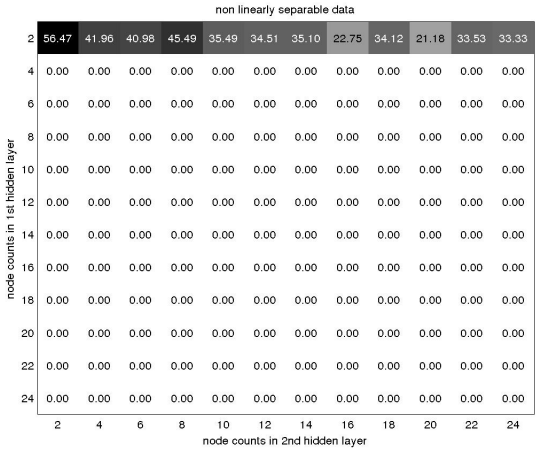
## 2.4 Non-linearly separable data

The visual plot and validation data miss-percentage box plot are shown below.
The dataset contains 300 points (train = 150, validation = 90, test = 60) of class1,
600 points (train = 300, validation = 180, test = 120) of class2,
800 points (train = 400, validation = 240, test = 160) of class3.

(a) Plot of the data



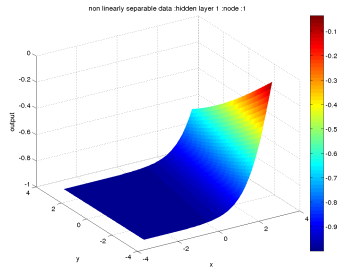
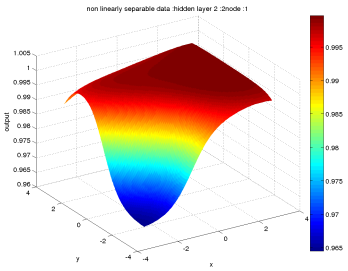
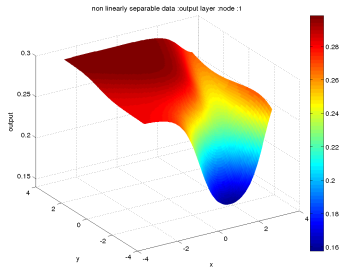
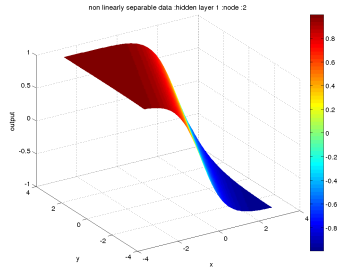
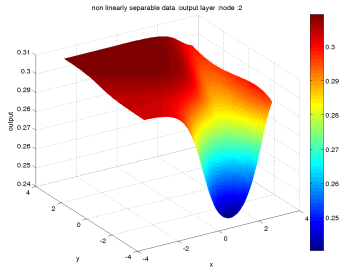
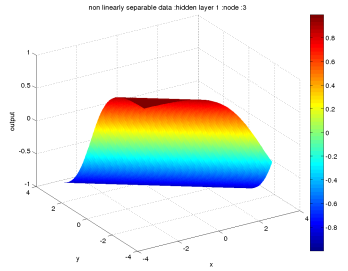
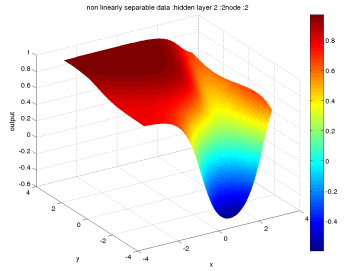
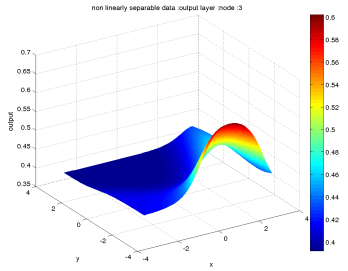
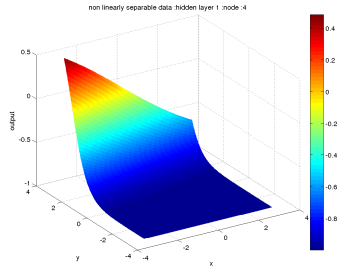

(b) Validation Miss percentage based on model complextites



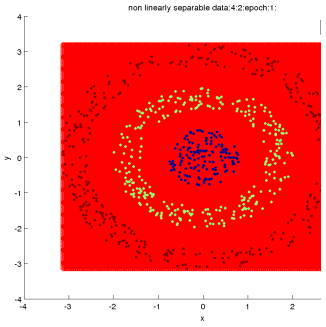

Based on the validation data miss percentage plot, the model of (hiddenlayer1 #nodes = 4, hiddenlayer2 #nodes = 2) is chosen as the best model.

### 2.4.1 MLFFNN with 2 Hidden layers

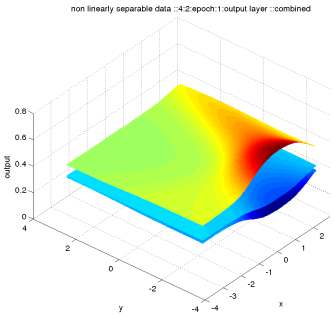
#### 2.4.1.1 After 1 epoch

| Hidden layer-1 | Hidden layer-2 | Output layer |
|---|---|---|
|  |  |  |
|  | |  |
|  |  |  |
|  | | |

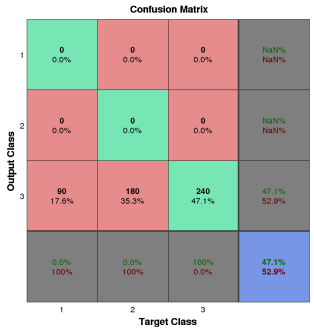
(a) Decision boundary plot



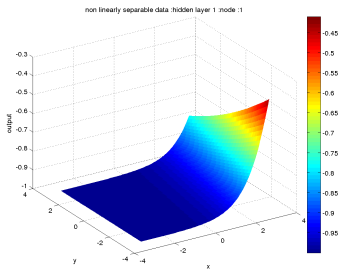
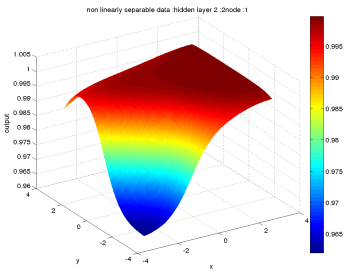
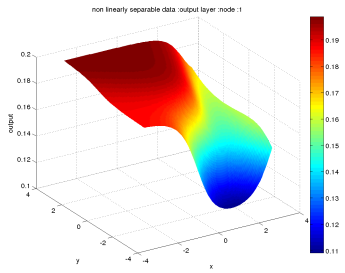
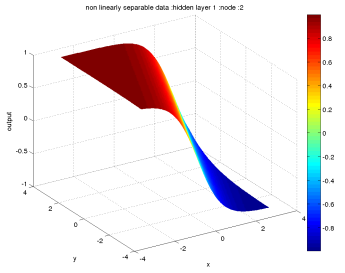
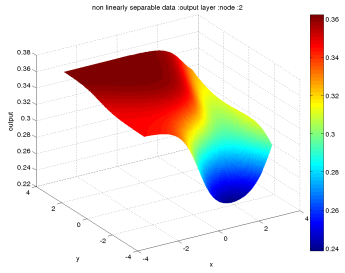
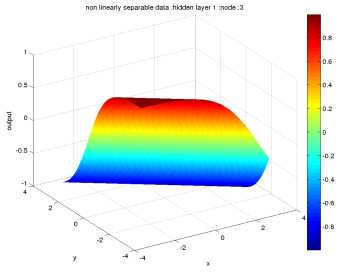
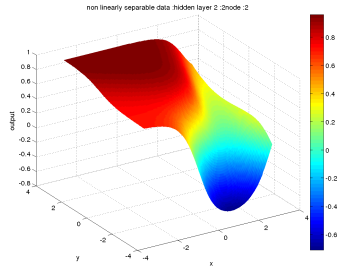
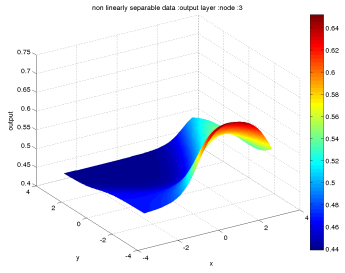
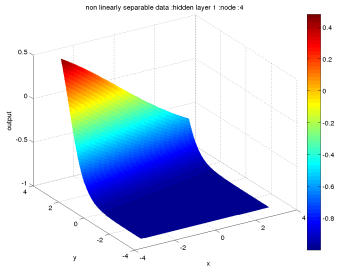

(b) combined output of final layer nodes




(c) confusion matrix of validation data

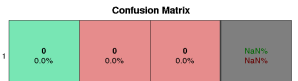



#### 2.4.1.2 After 2 epochs

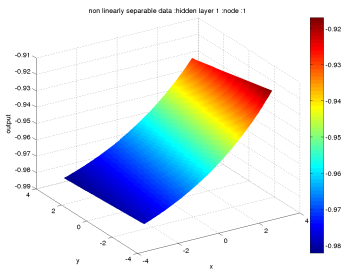
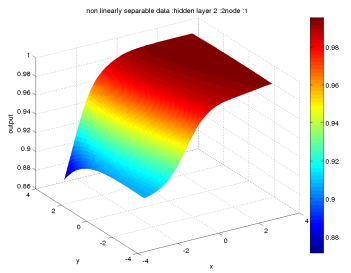
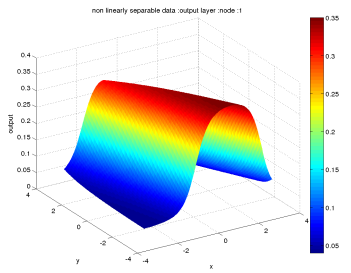
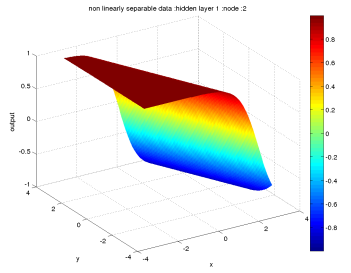
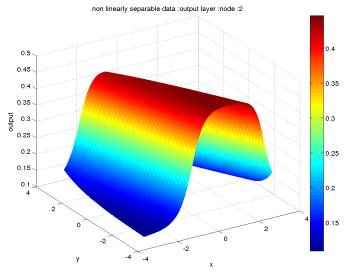
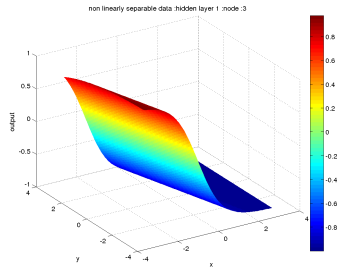
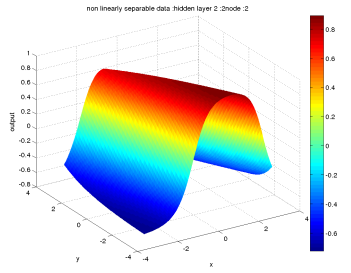
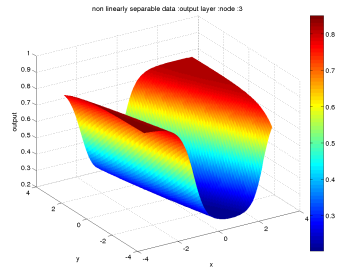
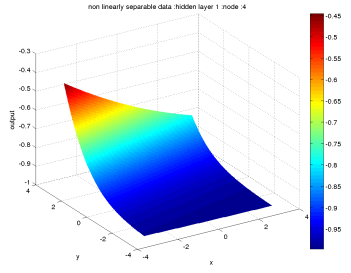
| Hidden layer-1 | Hidden layer-2 | Output layer |
|---|---|---|
|  |  |  |
|  | |  |
|  |  |  |
|  | | |

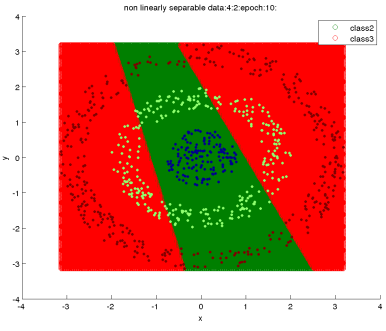


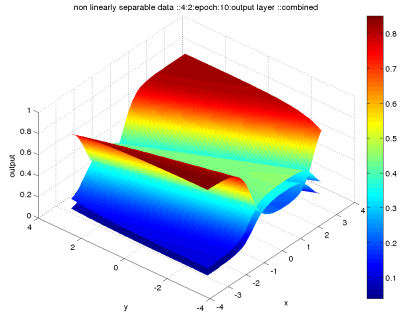


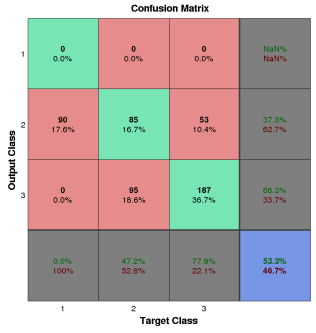


#### 2.4.1.3 After 10 epochs

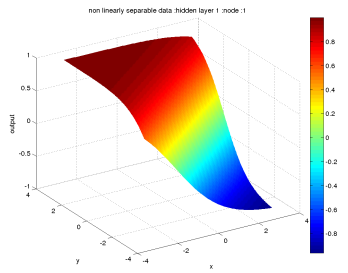
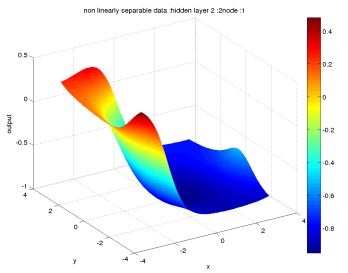
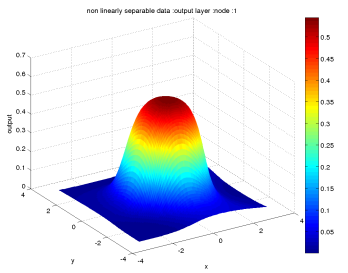
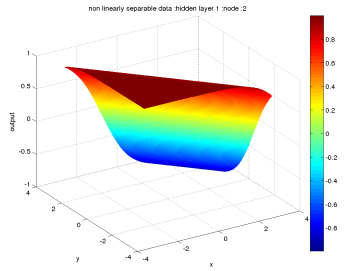
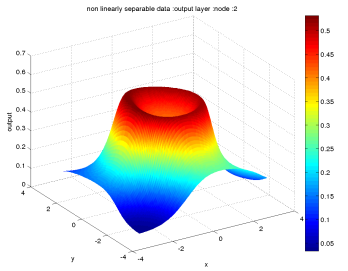
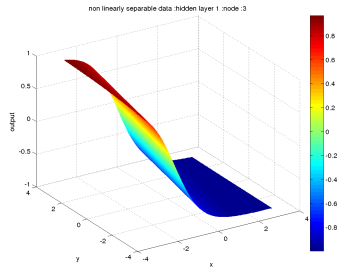
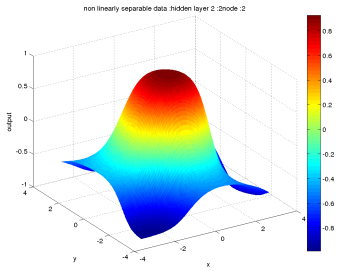
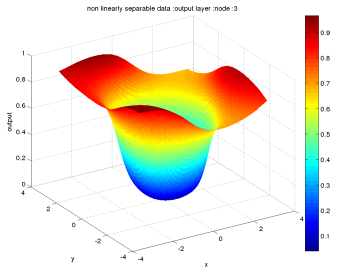
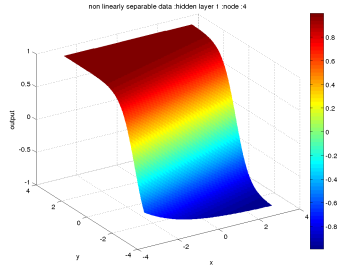
| Hidden layer-1 | Hidden layer-2 | Output layer |
|---|---|---|
|  |  |  |
|  | |  |
|  |  |  |
|  | | |

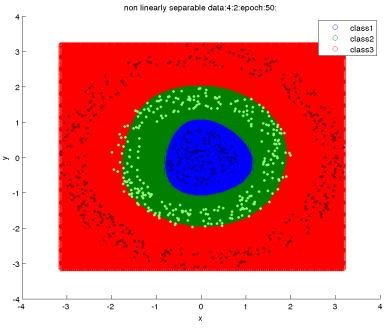


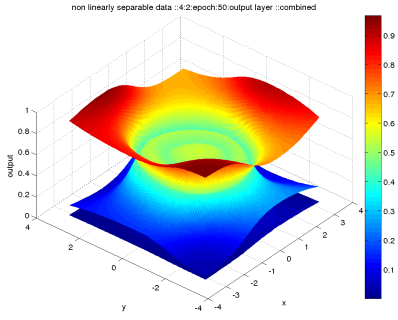



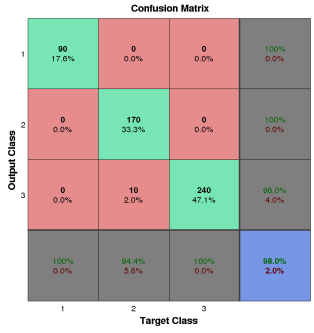



#### 2.4.1.4 After 50 epochs

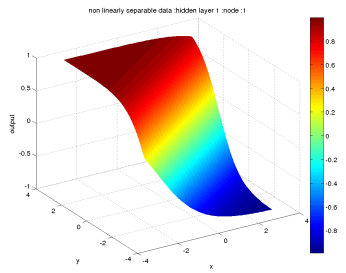
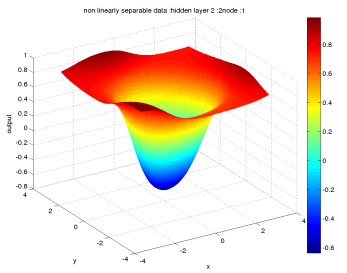
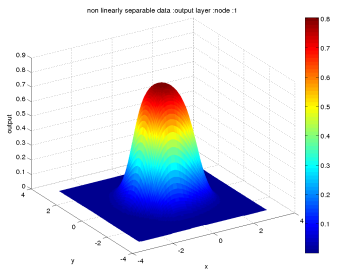
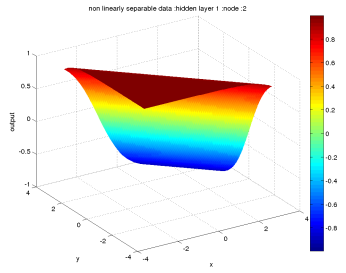
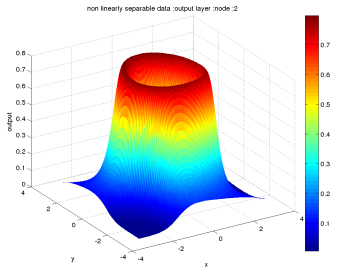
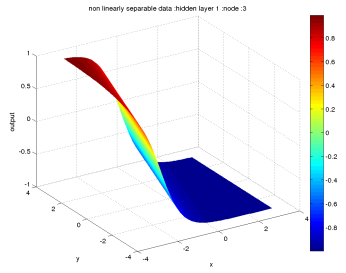
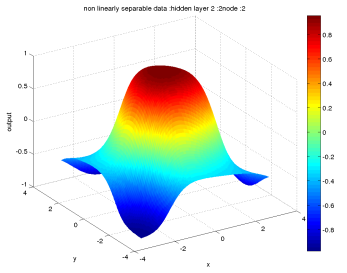
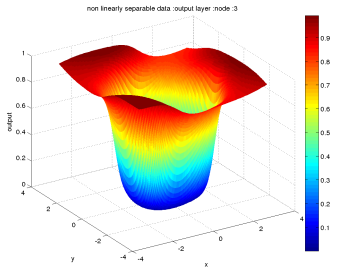
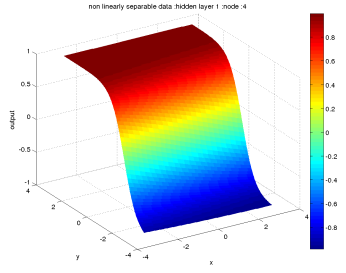
| Hidden layer-1 | Hidden layer-2 | Output layer |
|---|---|---|
|  |  |  |
|  | |  |
|  |  |  |
|  | | |

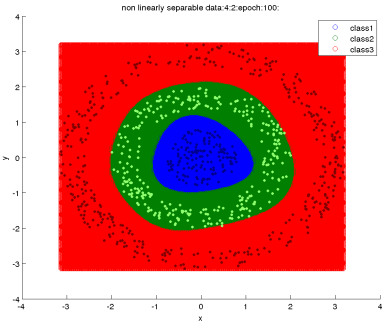


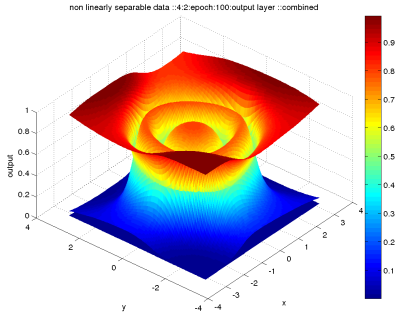



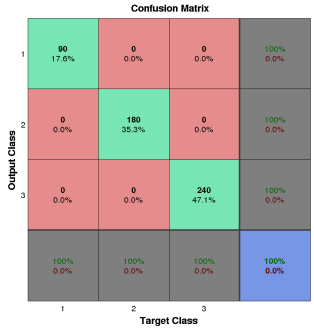



#### 2.4.1.5 After 100 epochs

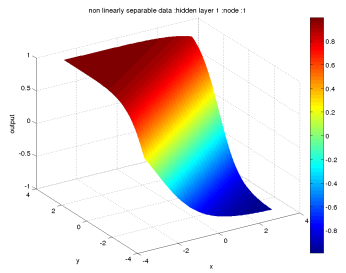
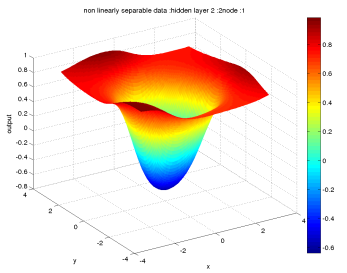
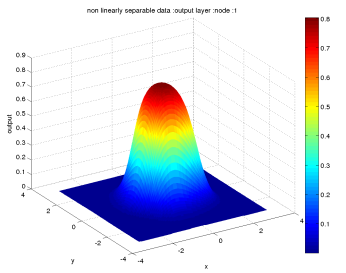
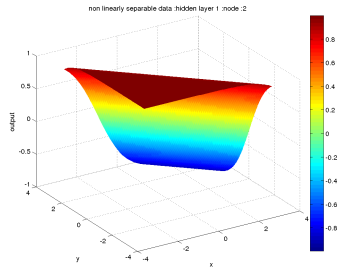
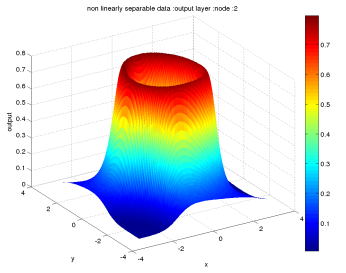
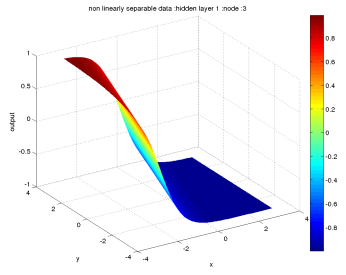
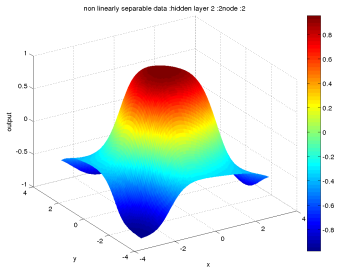
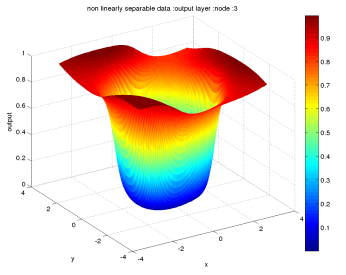
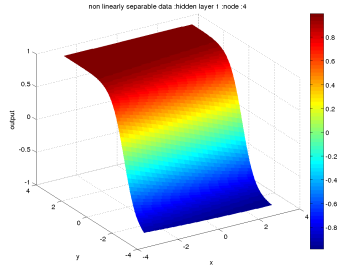
| Hidden layer-1 | Hidden layer-2 | Output layer |
|---|---|---|
|  |  |  |
|  | |  |
|  |  |  |
|  | | |

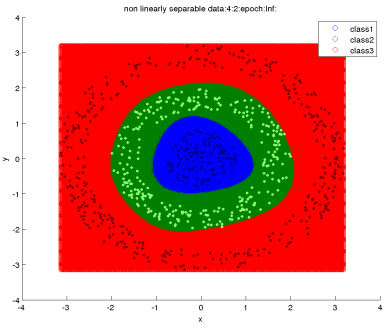


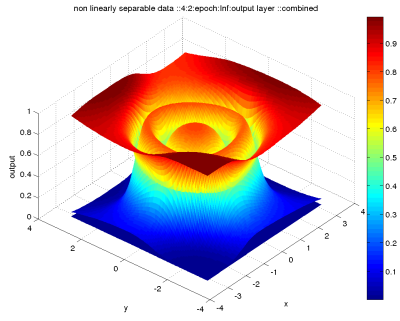



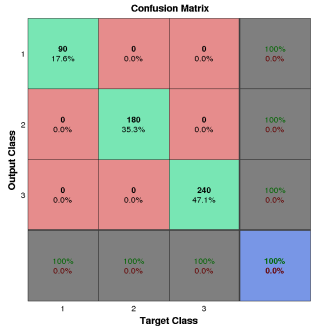



#### 2.4.1.6 After training

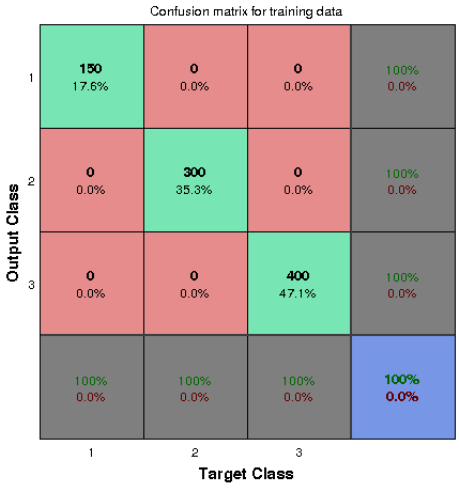
| Hidden layer-1 | Hidden layer-2 | Output layer |
|---|---|---|
|  |  |  |
|  | |  |
|  |  |  |
|  | | |



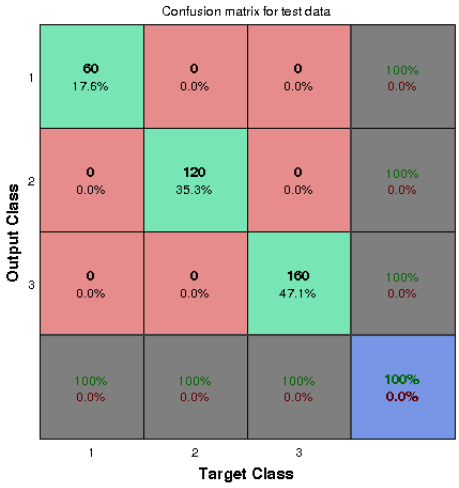



(a) confusion matrix of train data




(b) confusion matrix of test data



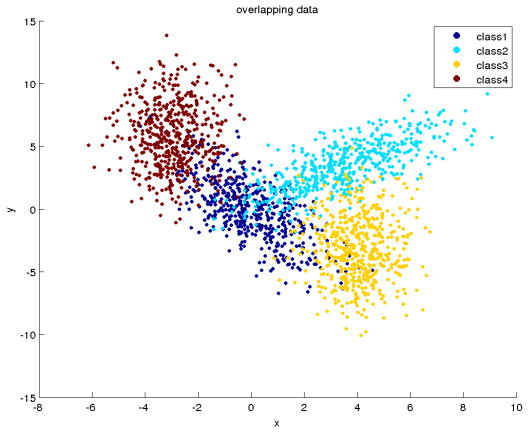

#### 2.4.1.7 Observations

- Except the model complexity with hidden-layer1 count = 2, all other models are giving miss-percentage of 0%.
- During initial epochs (epochs 1, 2) of training, the model is in arbitrary state and it is predicting all the examples as class-3. After the training runs for multiple epochs, at 10th epoch, we can notice that the model is learning about the other classes (class-2).
- At $50^{th}$ epoch, the model has almost learned the boundary of different classes and trying to stabilize the boundaries. Still, we can see that the probabilities of appropriate classes are not high enough, which reveals that the model is not so confident about the class boundaries yet.
- At $100^{th}$ iteration, we can notice from the combined plot of output layer nodes, that the confidence of model has been increased and boundaries are almost freezed to give better efficiency.

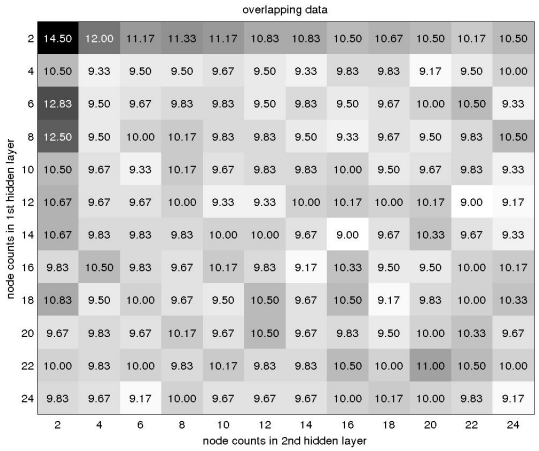
## 2.5 Overlapping data

The dataset contains 4 classes which are overlapped as shown in figure below. The dataset contains 500 points (train = 250, validation = 150, test = 100) in each of the 4 classes.

(a) Plot of the data



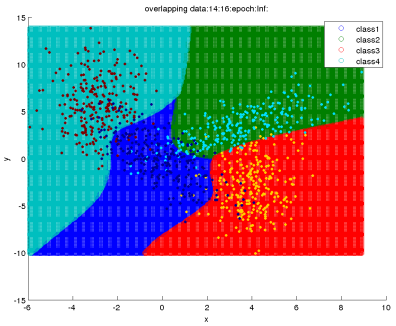

(b) Validation Miss percentage based on model complextites

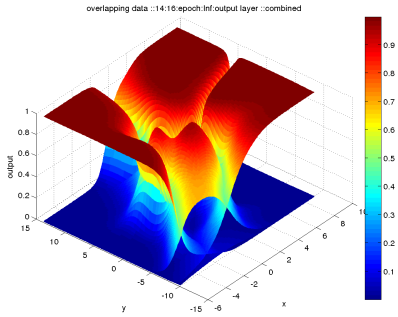


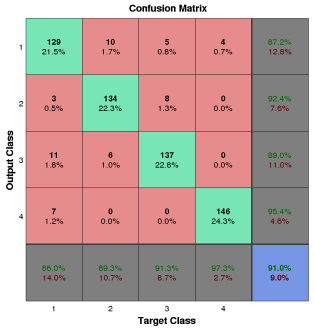

Based on the validation data miss percentage plot, the model of (hiddenlayer1 #nodes = 14, hiddenlayer2 #nodes = 16) is chosen as the best model.

### 2.5.1 MLFFNN with 2 Hidden layers

#### 2.5.1.1 Decision boundary & Confusion matrix



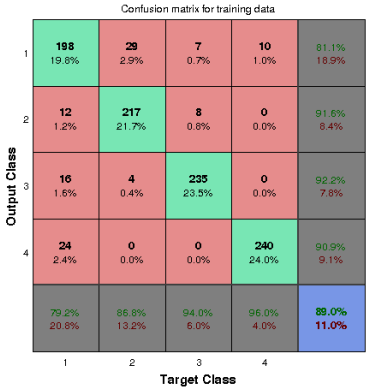



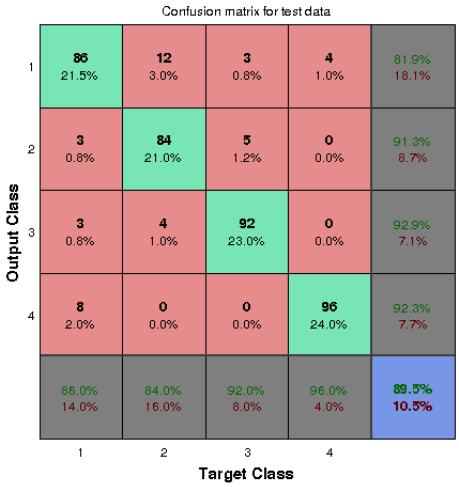

#### 2.5.1.2 Observations

- Since, the data is overlapped, the network needs more hidden nodes to model the data. The best model (hiddenlayer1 #nodes = 14, hiddenlayer2 #nodes = 16) converges in $\tilde{8}0$ epochs.
- The decision boundary learned by Neural network is complex non-linear function.
- Adding further nodes does not seem to increase the accuracy, as we see from the miss percentage plot.

(a) confusion matrix of train data




(b) confusion matrix of test data




..

# 1 Regression

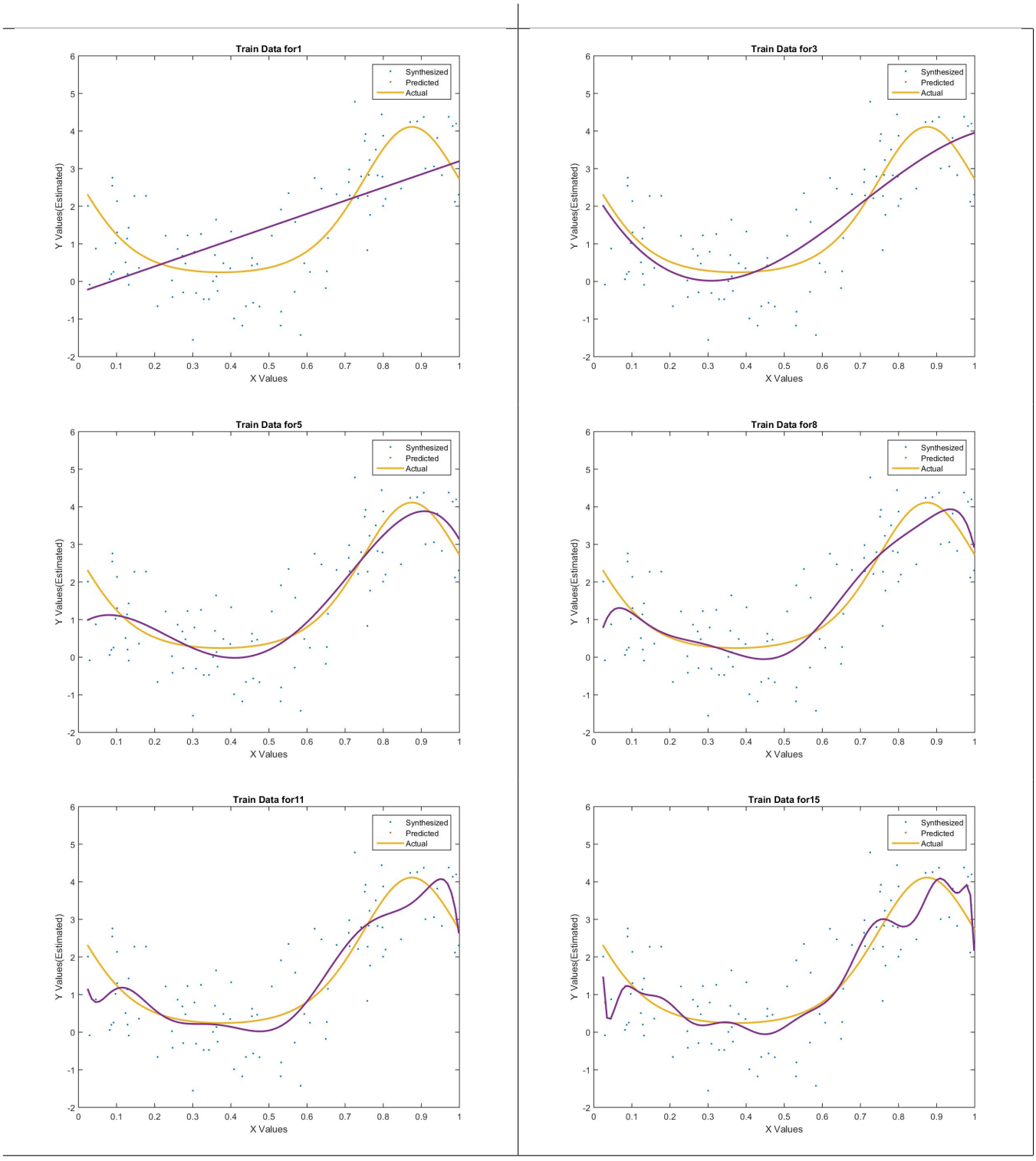
## 1.1 Univariate data

### 1.1.1 Polynomial regression

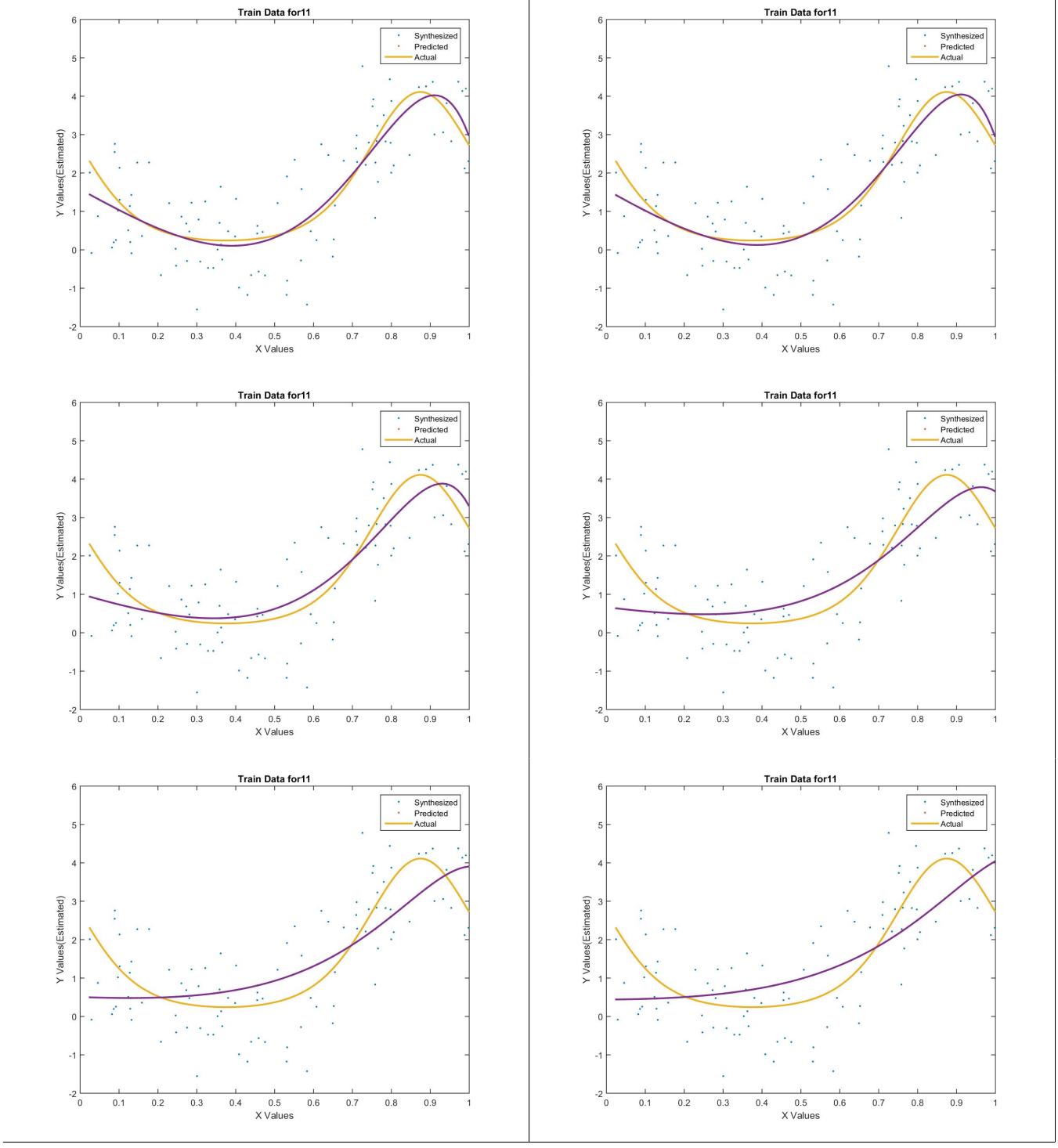
#### 1.1.1.1 Experiments

1. The data was generated by using function $e^{cos(2*\pi*x)-sin(2*\pi*x)}$ and then adding a random noise to it. The random noise was generated using matlab function 'randn' function which generates the numbers with mean zero and variance 1.
2. This data was divided in test, train and validation data.The experiments were performed using validation data of size 200,test data of size 300 and train data of different sizes.
3. The model wass chosen by cross-validation method

## 1.Train Size 100

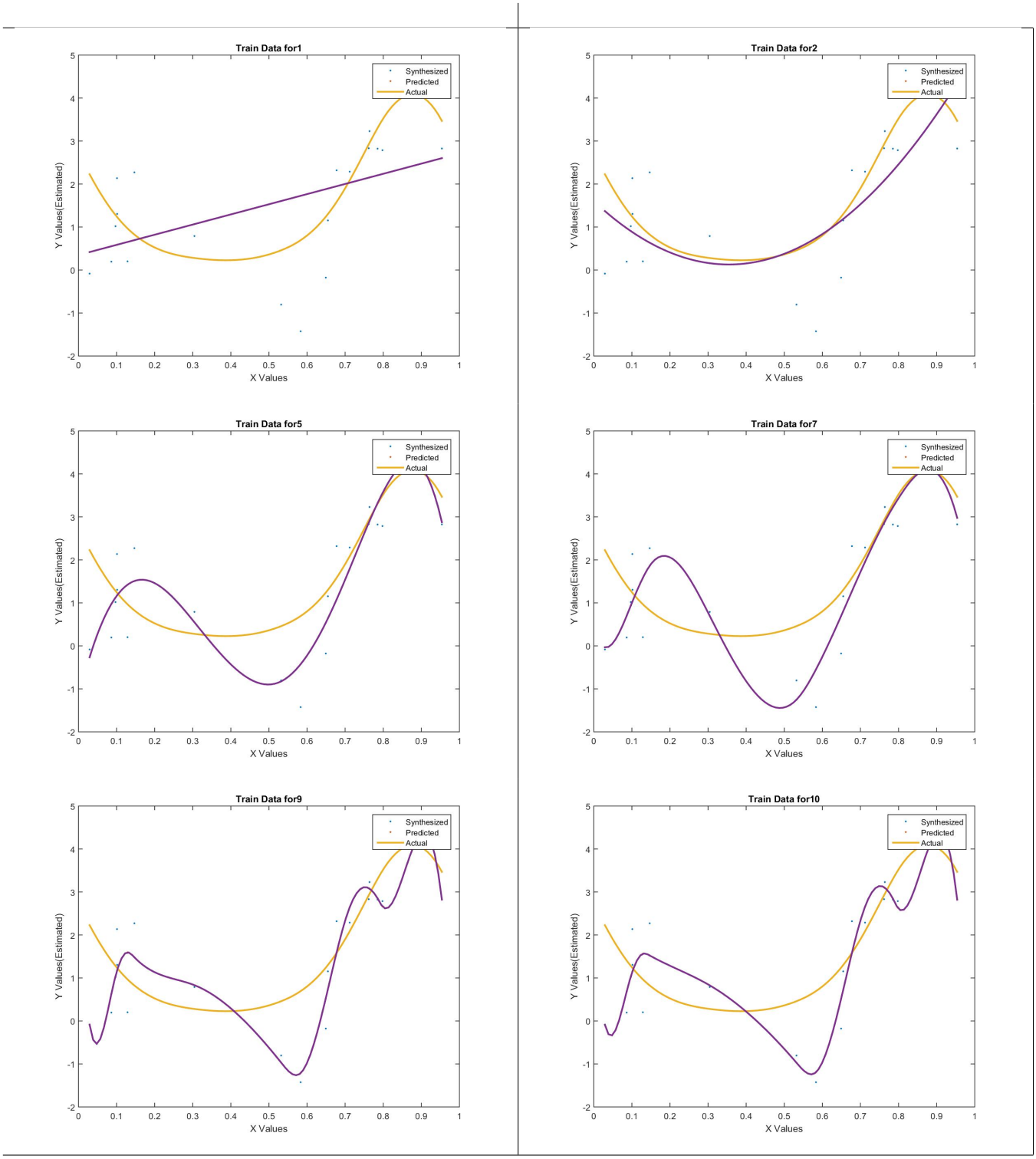



**Plots for different complexities without regularization**

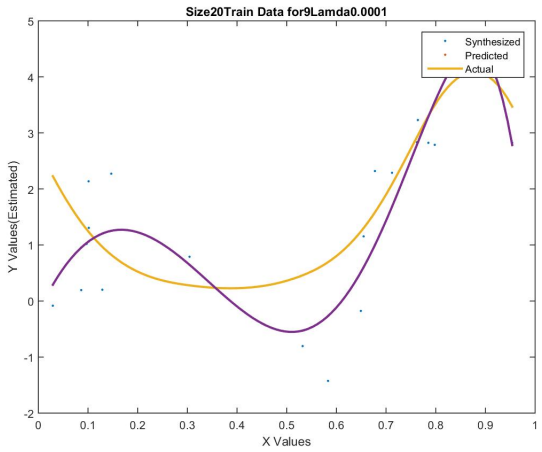


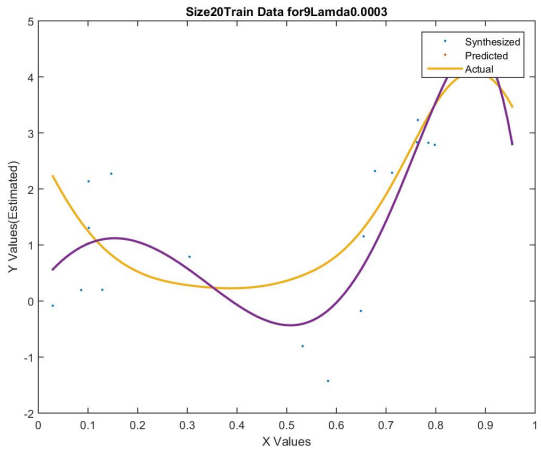

**Plots for complexity 11 and different regularization parameters**

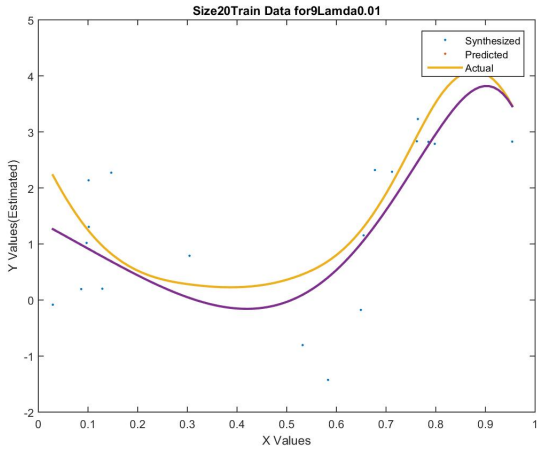
## 2.Train Size 20

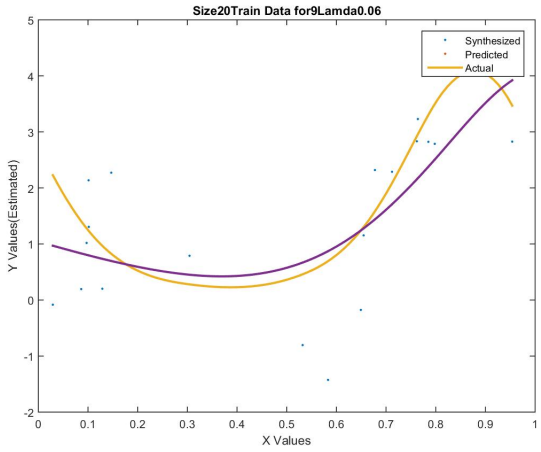


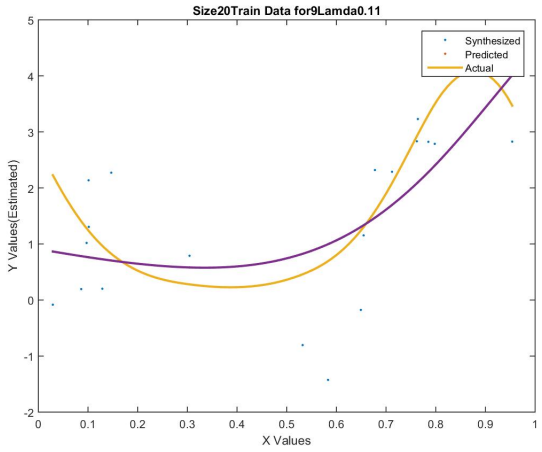

**Plots for different complexities without regularization**

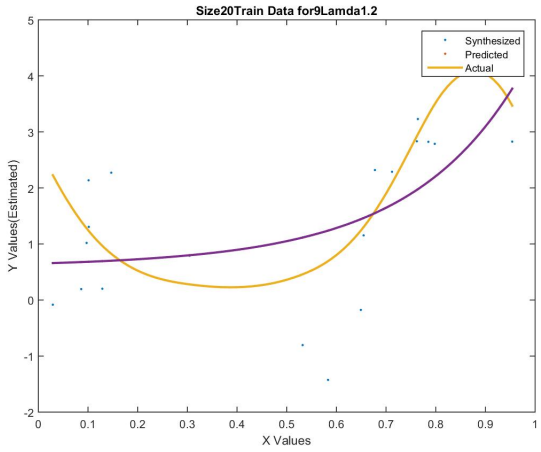


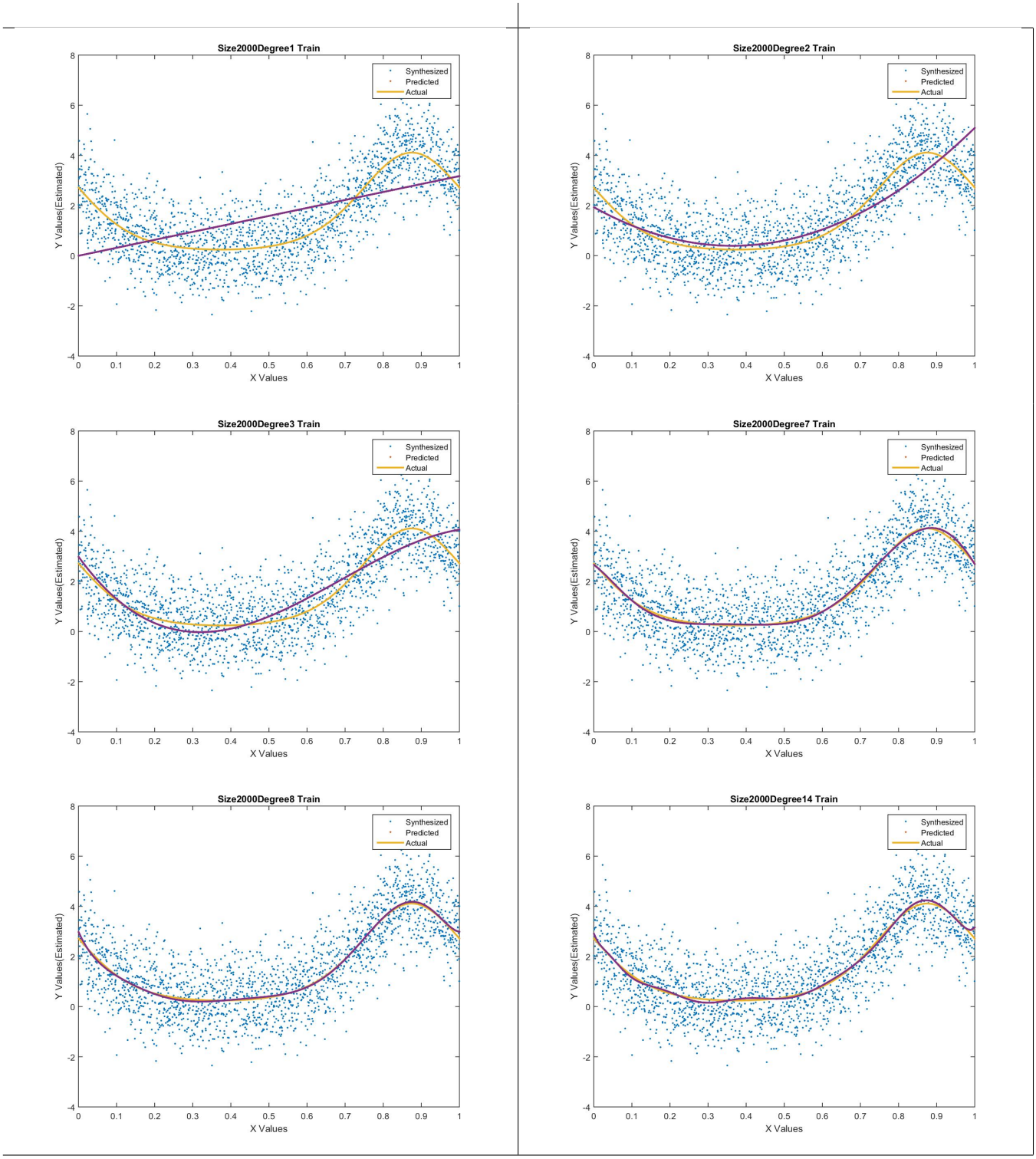




**Plots for complexity 9 and different regularization parameters**

## 3.Train Size 2000




**Plots for different complexities without regularization**

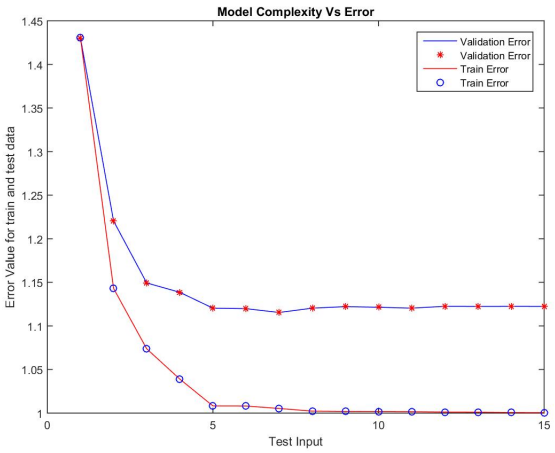



Figure 1: Model Complexity Vs error for train size 2000

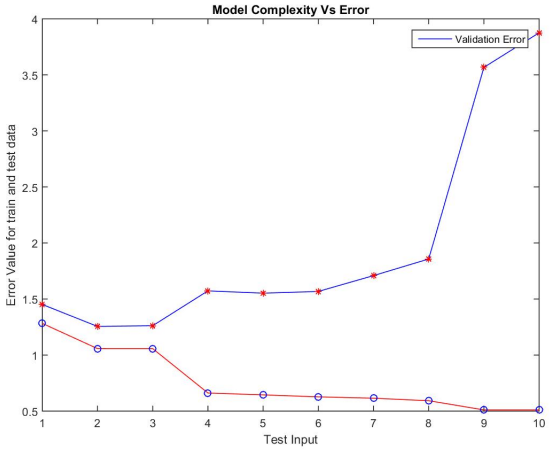



Figure 2: Model Complexity Vs error for train size 20

#### 1.1.1.2 Observations

1. Train Size 100

| Model Complexity | Error |
|---|---|
| 2 | 1.2265 |
| 3 | 1.156 |
| 7 | 1.1736 |
| 9 | 1.1826 |
| 11 | 1.1832 |

2. Train Size 2000

| Model Complexity | Error |
|---|---|
| 2 | 1.220 |
| 3 | 1.4930 |
| 7 | 1.1153 |
| 9 | 1.2224 |
| 11 | 1.223 |

3. Train Size 20

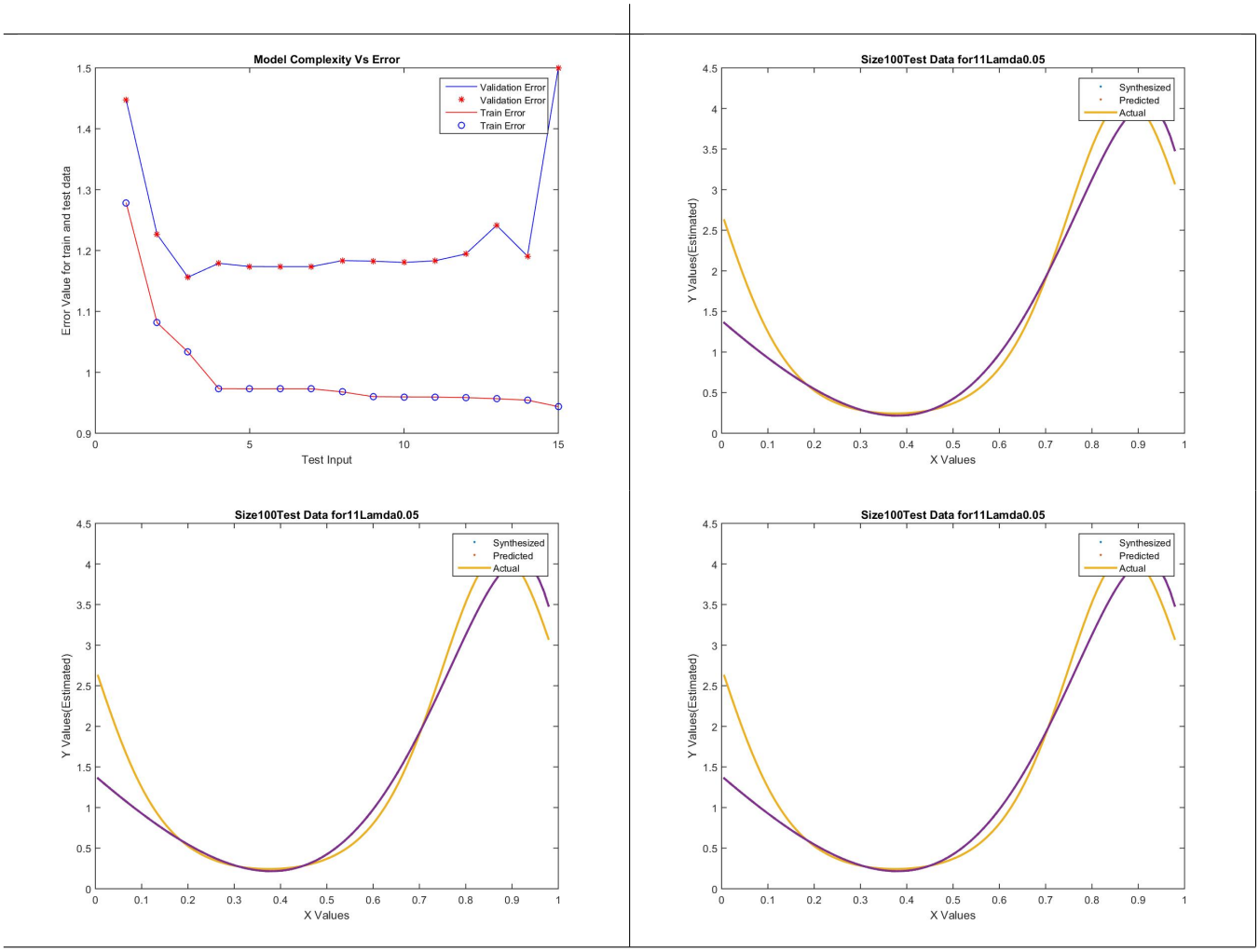
| Model Complexity | Error |
|---|---|
| 2 | 0.1569 |
| 3 | 0.0066 |
| 7 | 0.0495 |
| 9 | 0.0796 |
| 11 | 0.3342 |

#### 1.1.1.3 Conclusion

**1.Train Size 100**

Without regularization, the model complexity came out to be 3. The error of test data for this model was 1.1076. As we can see from the output model for higher degrees 9, 10 and so on it starts overfitting and hence, we go for regularization.

Best Model : Complexity = 9, lamda=0.05. Error for test data=1.0037




**2.Train Size 20**

Without regularization, the model complexity came out to be 3. The error of test data for this model was 1.0509. As we can see from the output model for higher degrees 11, 12 and so on it starts overfitting and hence, we go for regularization.

Best Model : Complexity = 11, lamda=0.05

### 1.1.2 Linear regression using RBF basis

#### 1.1.2.1 Experimentation

Radial Basis Function was performed on the synthesized univariate data.

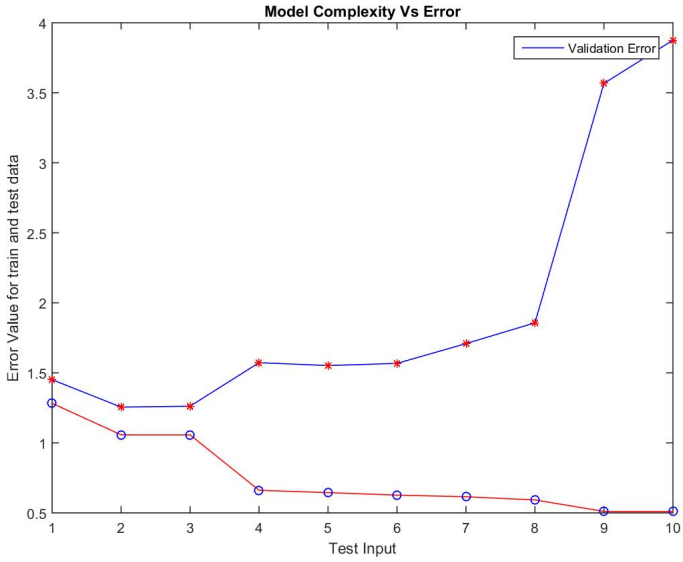
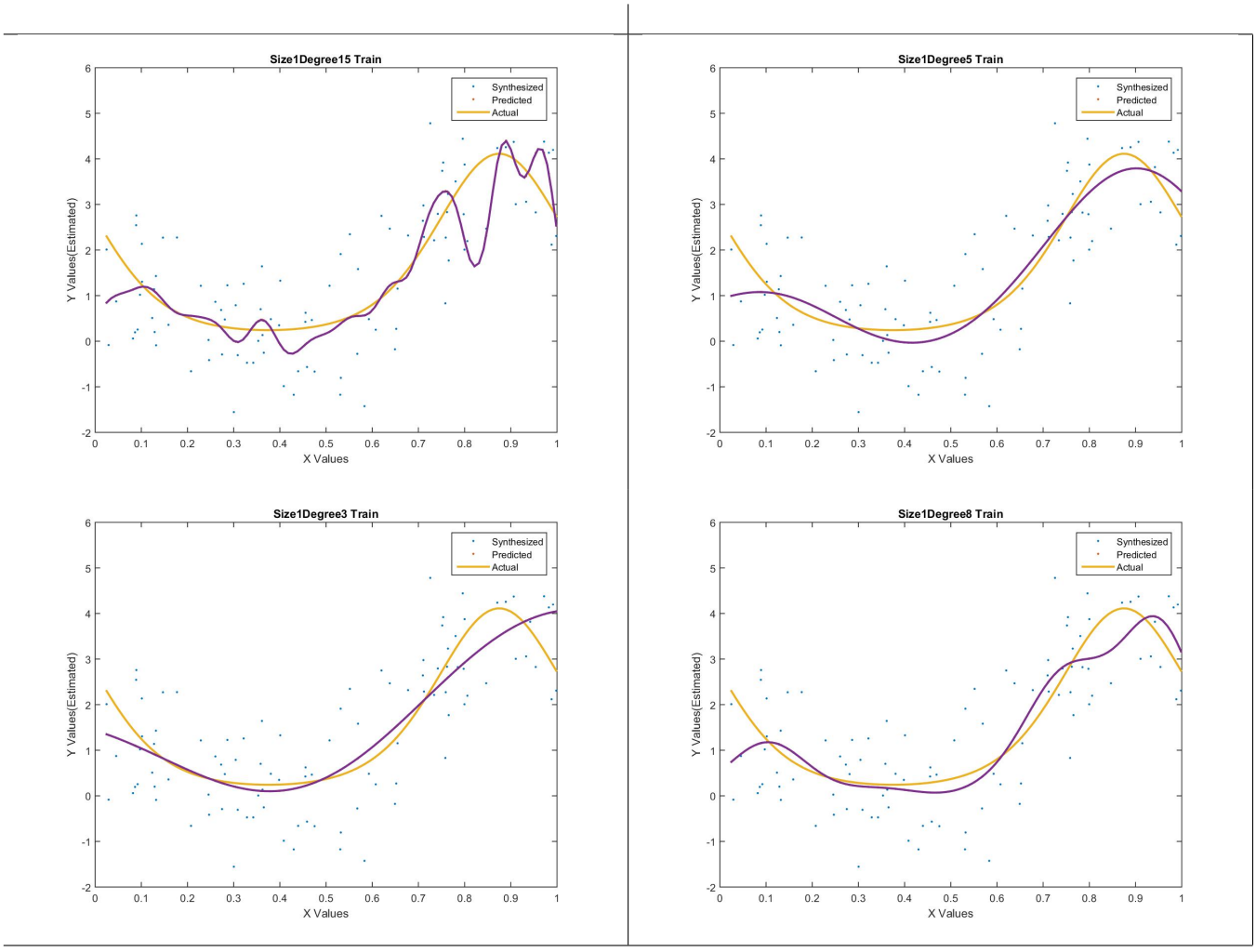
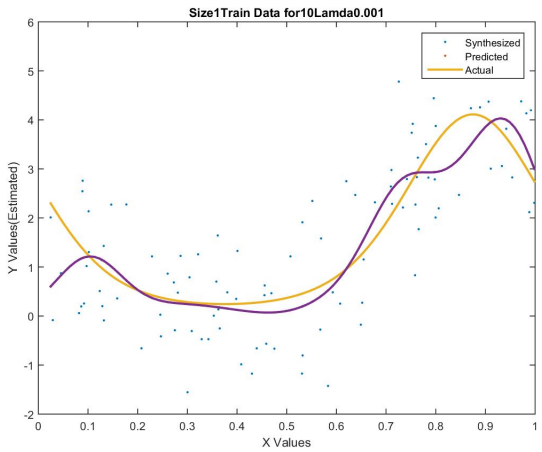
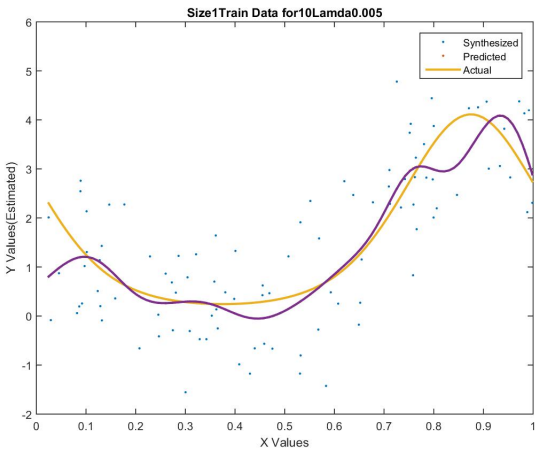


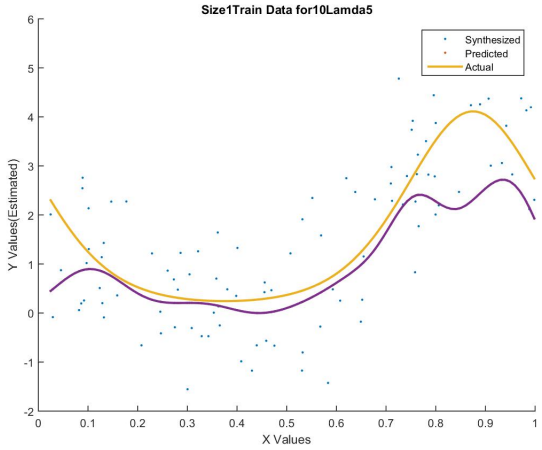

Figure 3: Model Complexity Vs error

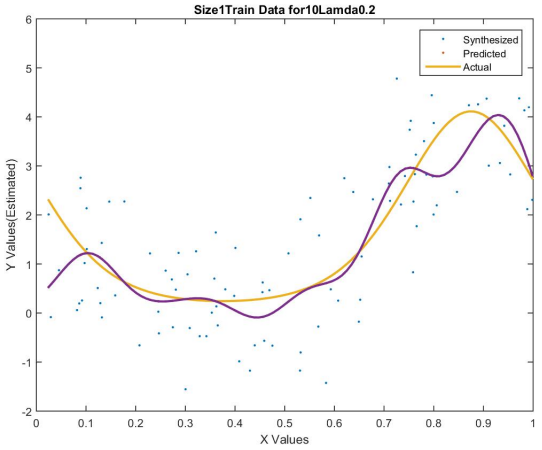



**Plots for different complexities without regularization**




**Plots for different regularization parameters**

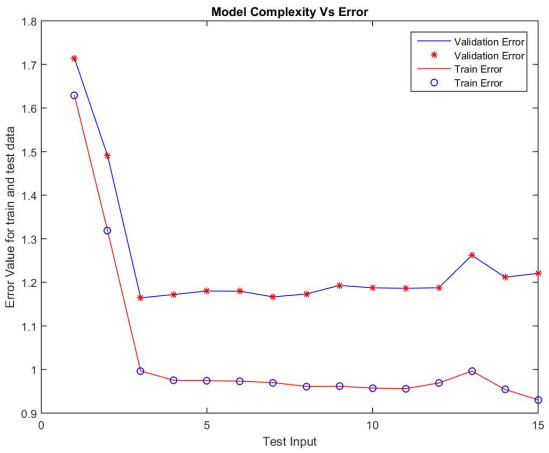
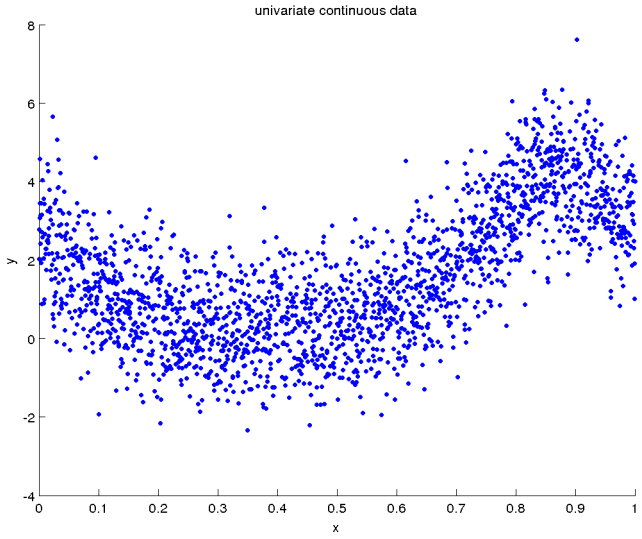


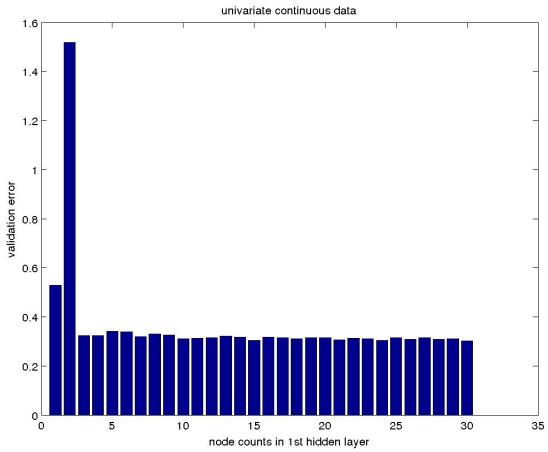

Figure 4: Model Complexity Vs error

#### 1.1.2.2 Observation

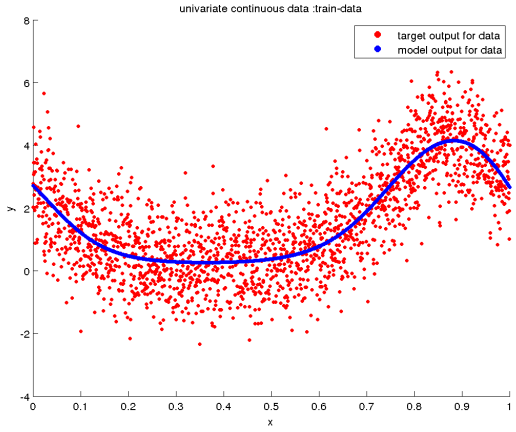
## 3.2 MLFFNN

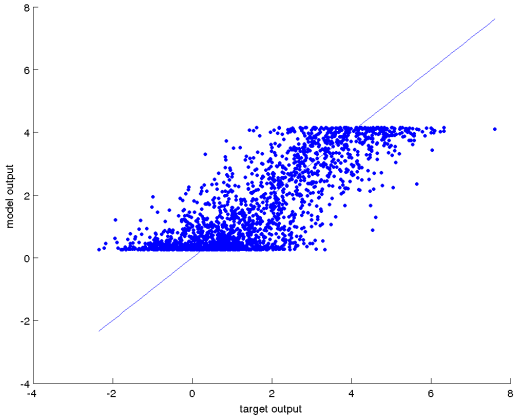
### 3.2.1 MLFFNN with 1 Hidden layer

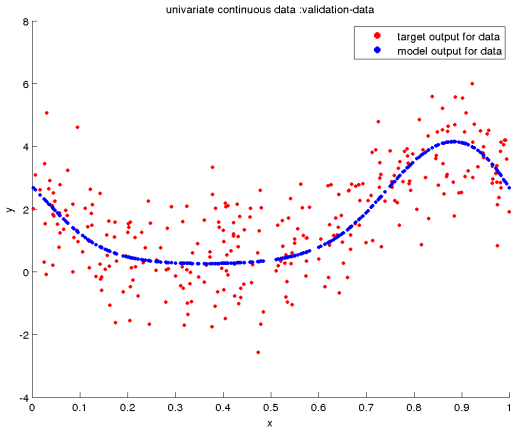


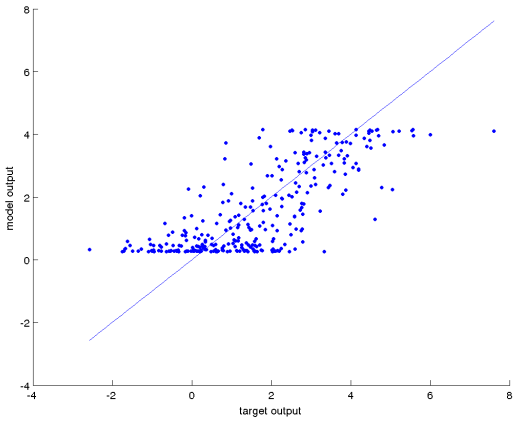



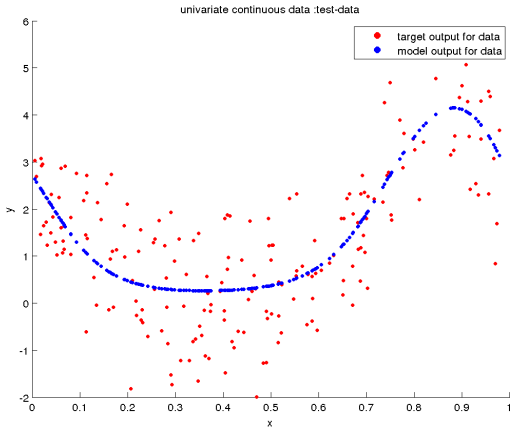
#### 3.2.1.1 Scatter plot with model output and target output of model with HiddenLayer1 nodes = 3

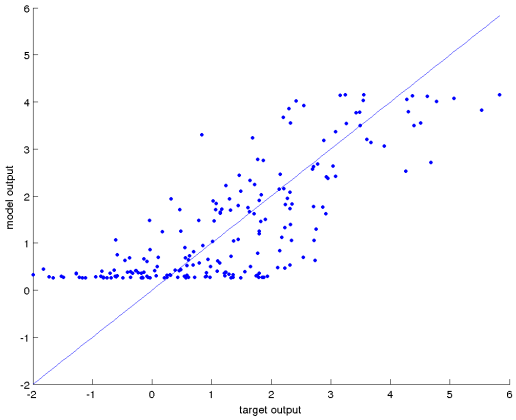


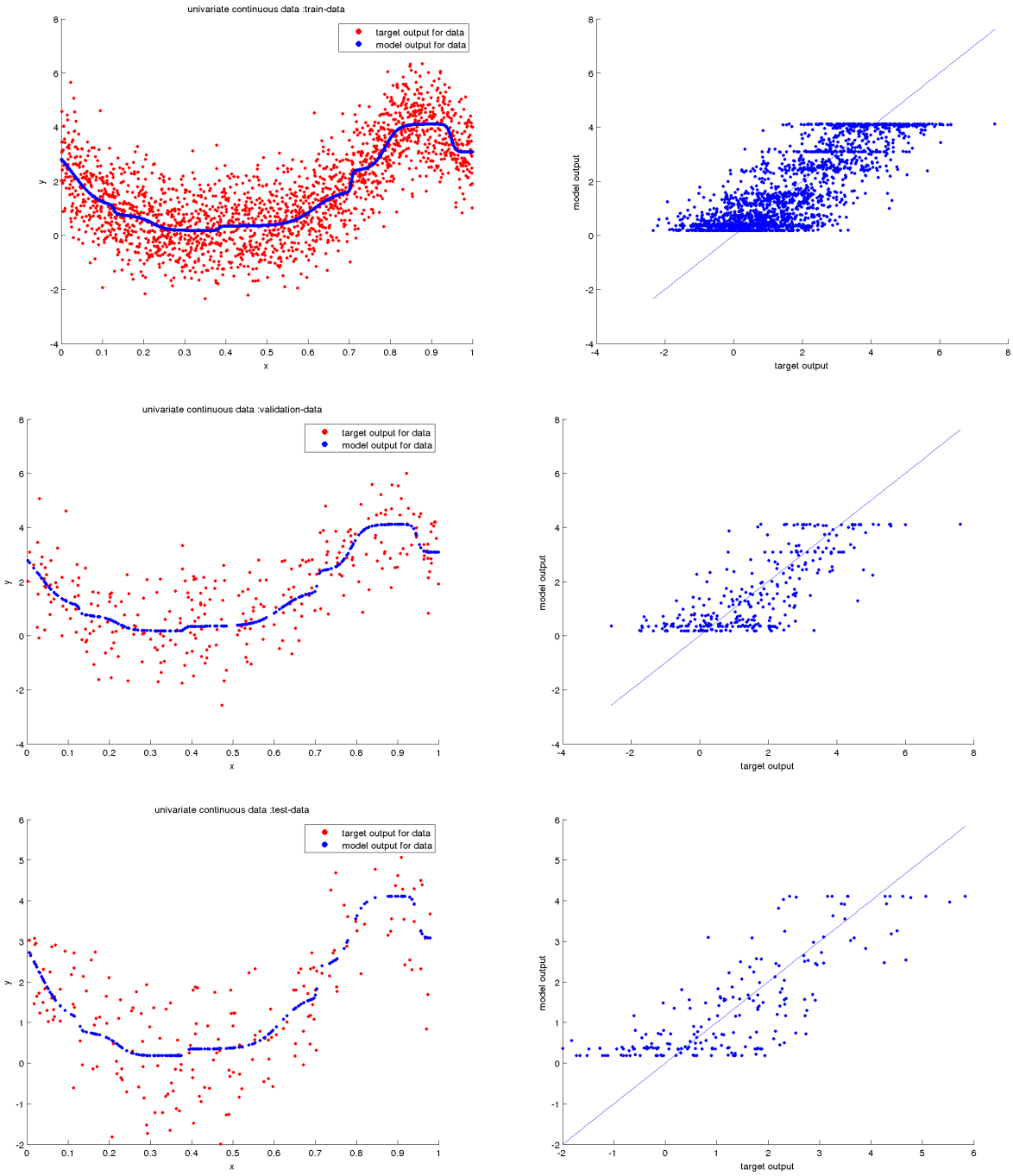



#### 3.2.1.2 Scatter plot with model output and target output of model with HiddenLayer1 nodes = 8




#### 3.2.1.3 Observations

- In univariate function approximation task, single hidden-layer neural network performs well for the given data even with only $\tilde{3}$ nodes in the hidden layer. (unlike high complex model in Polynomial regression)
- We observe that the single-hidden-layer model gives similar Mean squared error for the model complexities of hidden-layer1 count = 3 to 30. So, we choose the model with hidden layer count = 3 as the best model.

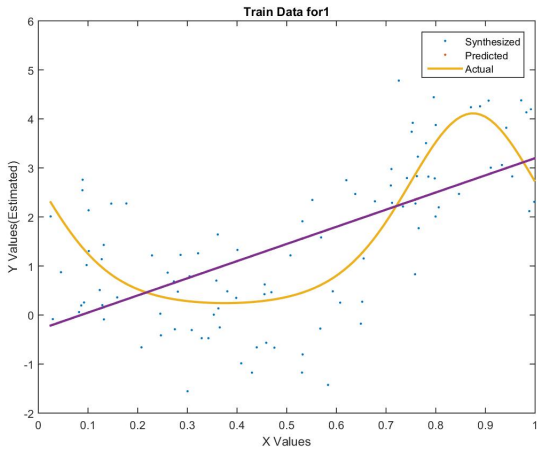
## 1.2 Bivariate data

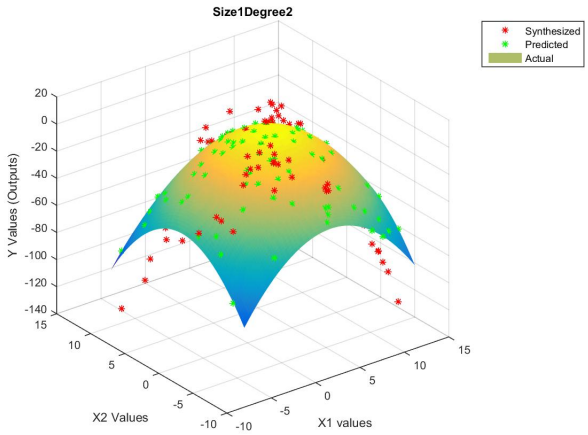
### 1.2.1 Linear regression using Gaussian basis

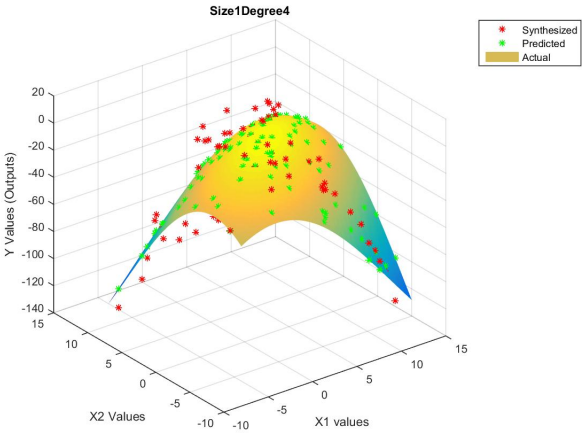
#### 1.2.1.1 Experimentations

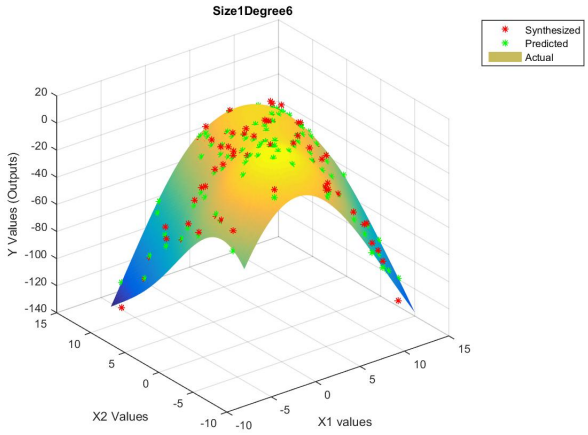
**1.Train Size 100**

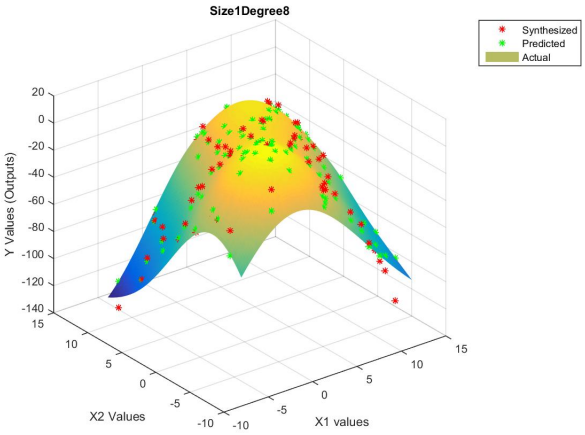
**Plot of target output and model output for training data, for different model complexities**

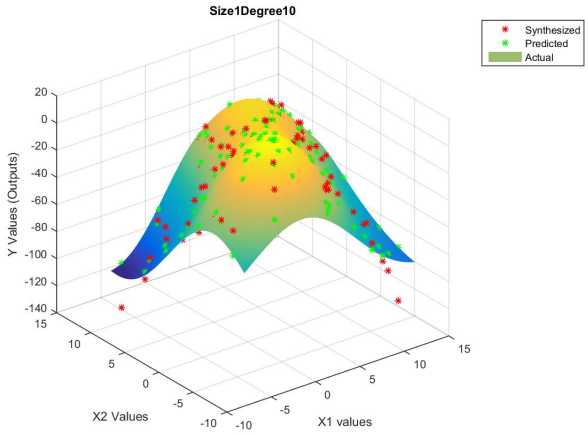


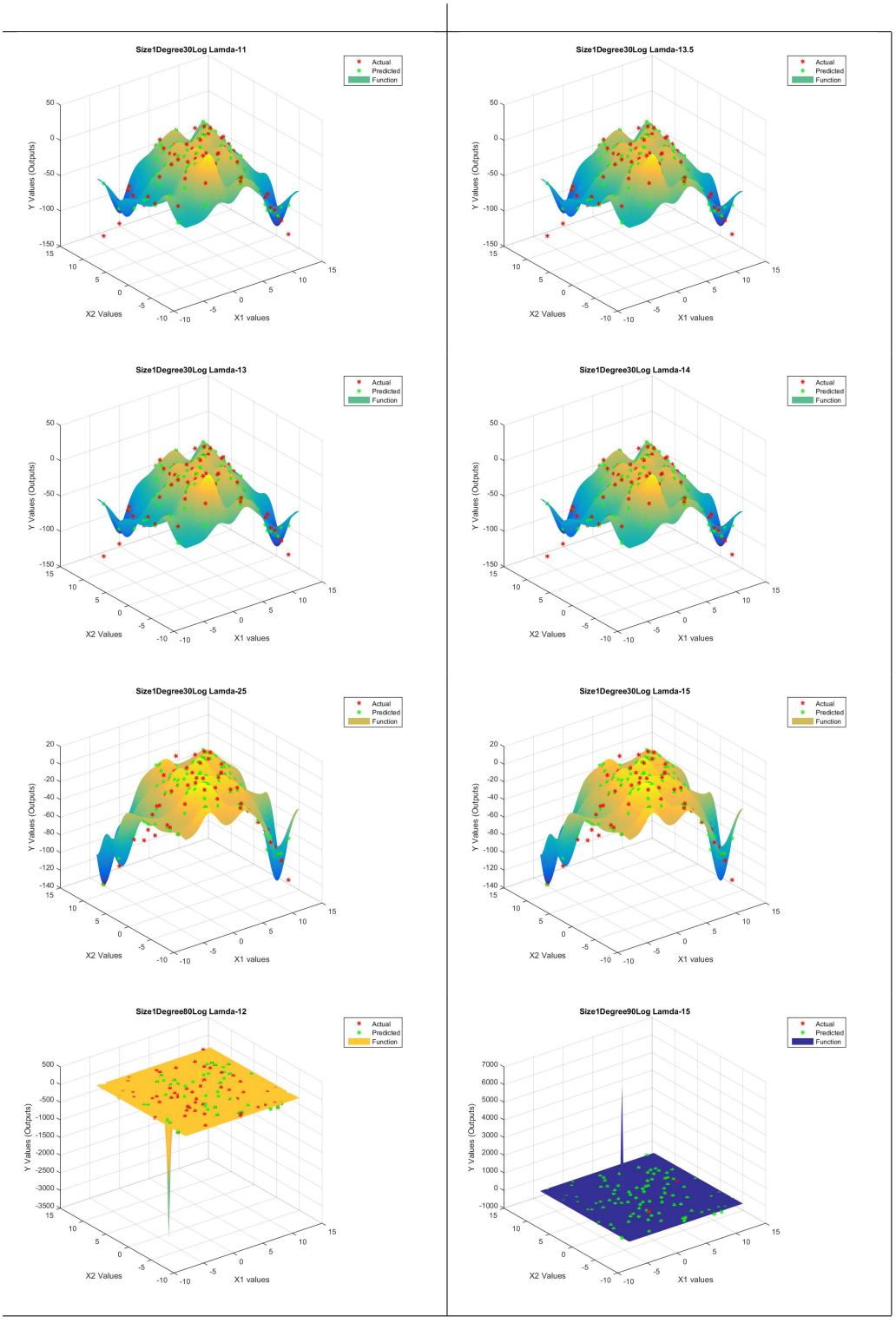






**Plots for different regularization parameters**

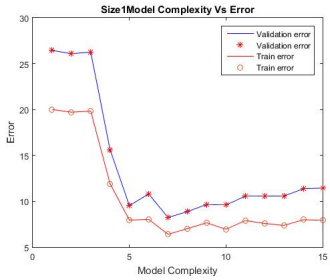



Figure 5: Model Complexity Vs error

#### 1.2.1.2 Obeservation

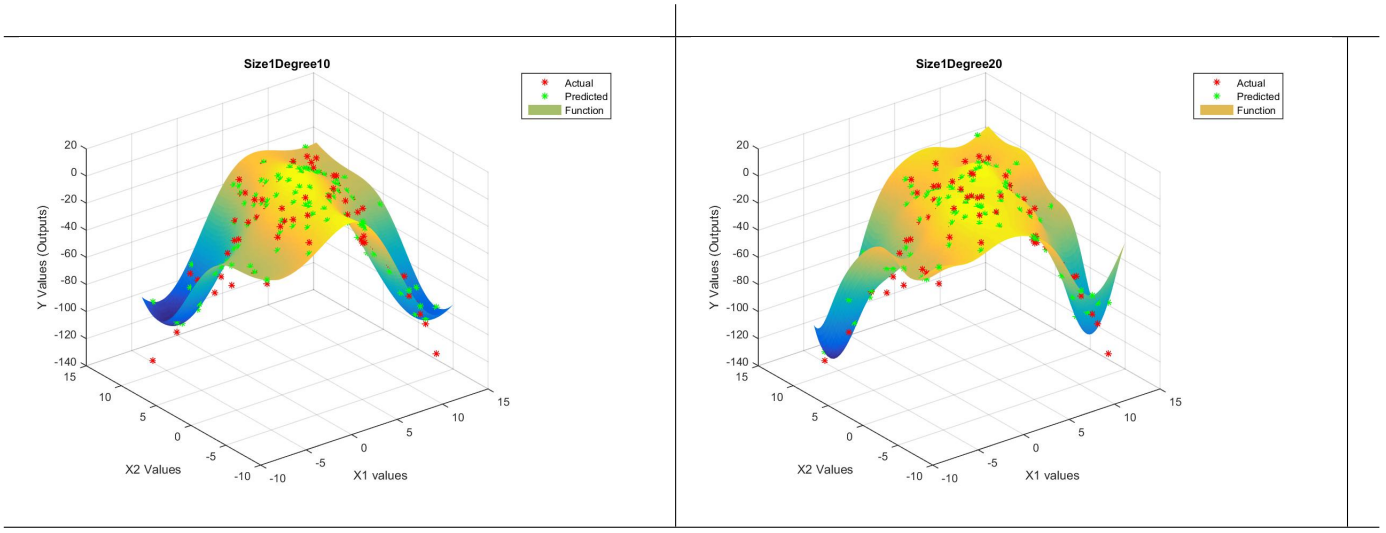
#### 1.2.1.3 Conclusion

1. The best model obtained was of degree 6.
2. Error for testdata was 6.4155.
3. Overfitting was observed at k=30, k=90 and hence regularization was done. The regularization was tried for $\lambda = e^-15, e^-10, e^-5$ but nothing gave good results.

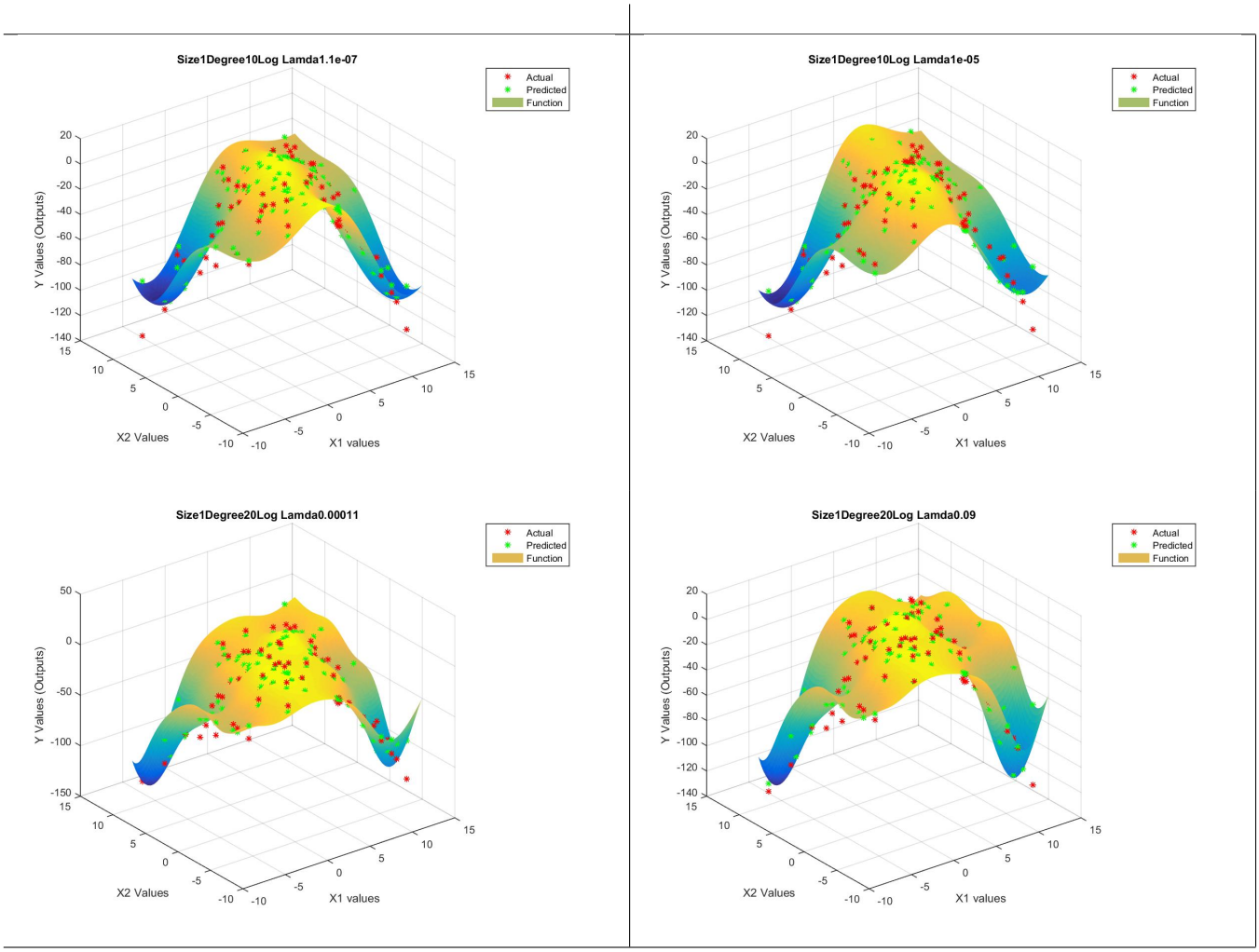
### 1.2.2 Linear regression using RBF basis Function

#### 1.2.2.1 Experiments

**1.Train Size 100**




**Plots for different complexities without regularization**



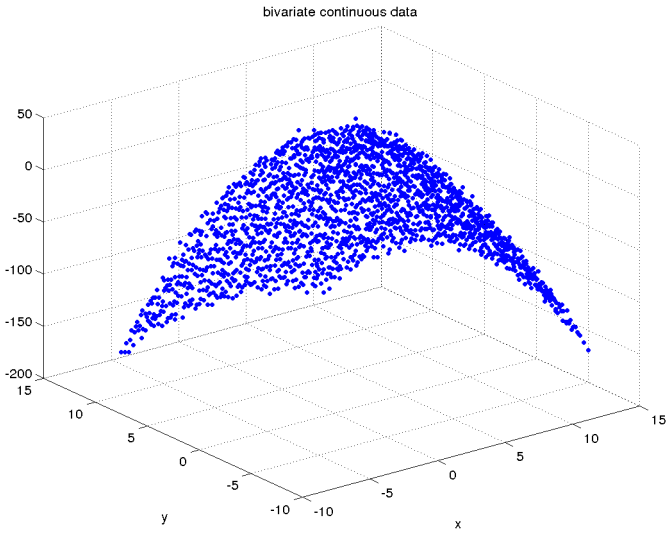

**Plots for different regularization parameters**

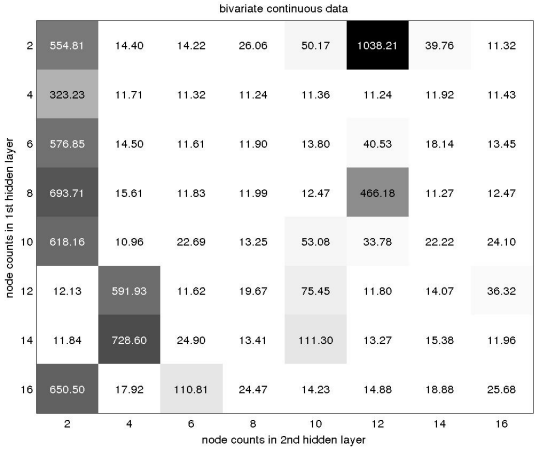
#### 1.2.2.2 Inferences

1. The univariate data was approximated by polynomial curve fitting very well.
2. For small train data, the model complexity was high and it suffered overfitting.
3. Though it gave good output at lower degree(Degree 3) when the higher complexity model was regularized, it was even more better.(At degree 9, $\lambda$=0.01.
4. For large train data the model obtained was good and did not suffer overfitting so no need of regularization.
5. The Gaussian Basis Function for bivariate data gave a good result for complexity 6.
6. So many experiments were performed to get the better model but for values $e^{-}10$, $e^{-}5$,0.001, 0.01 the performance was degraded.
7. The Gaussian Basis Function for bivariate data gave a good result for complexity 6.
8. As we decrease the $\lambda$ value, the smoothening of the function increases thereby reducing the overfitting phenomenon this is because the effect of weight parameters is reduced.
9. As we increase the model complexity, the model gets tuned to the training data values and so there is a possibility of overfitting.

## 3.3 Bivariate data with MLFFNN

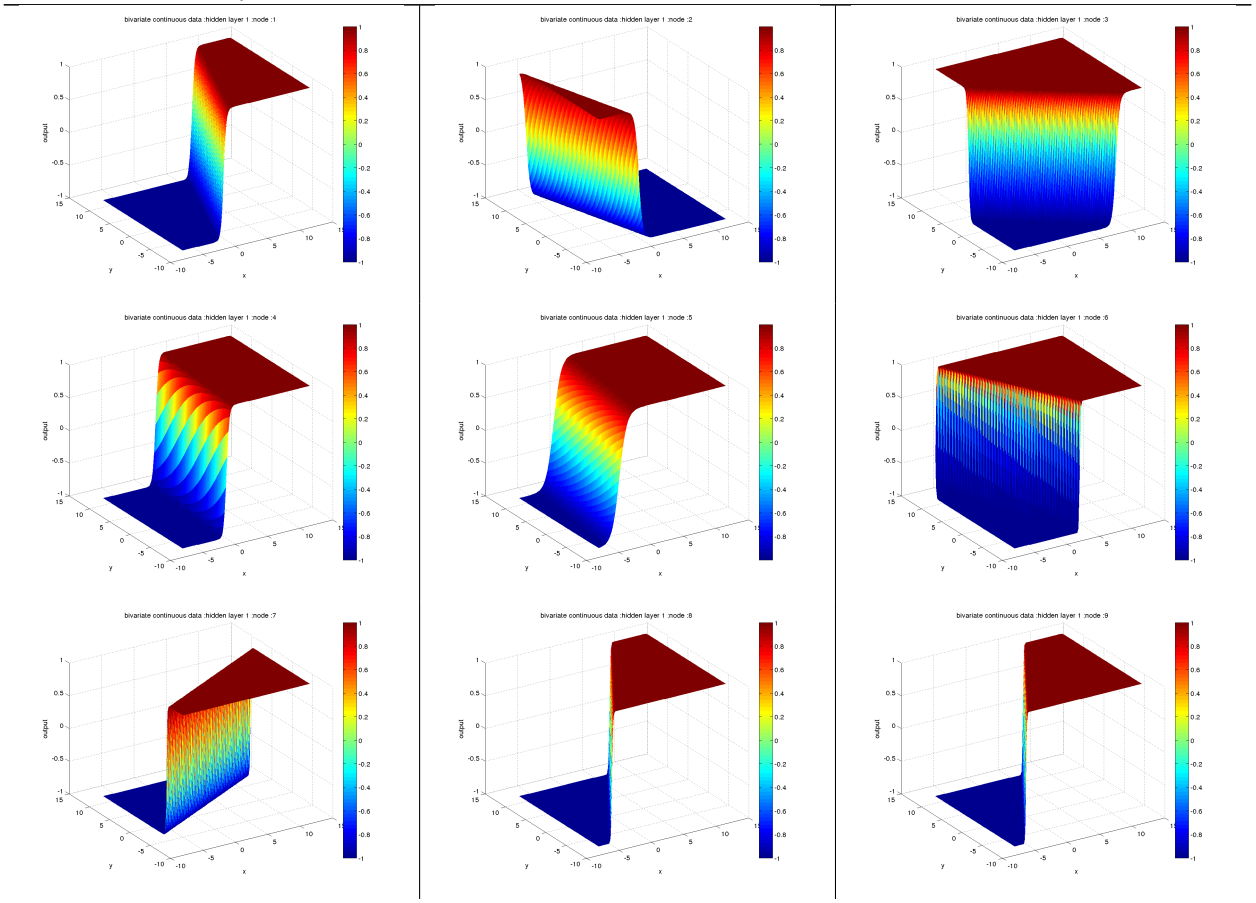
### 3.3.1 MLFFNN with 2 Hidden layers

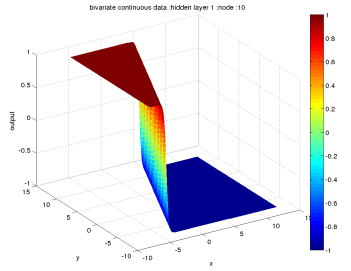


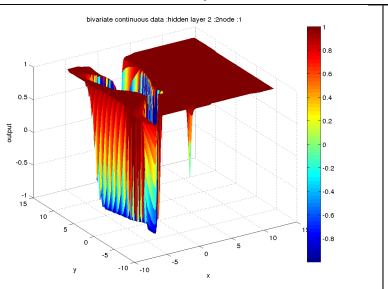

#### 3.3.1.1 After 1 epoch

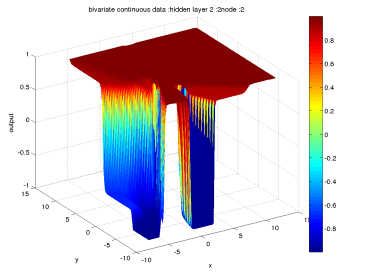
Hidden layer-1

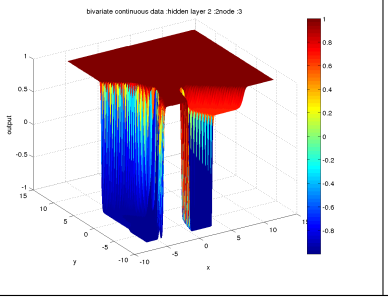


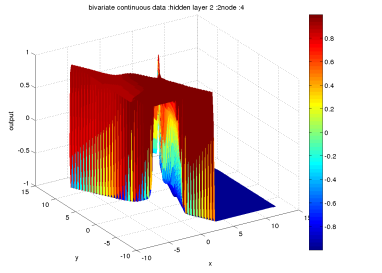



Hidden layer-2

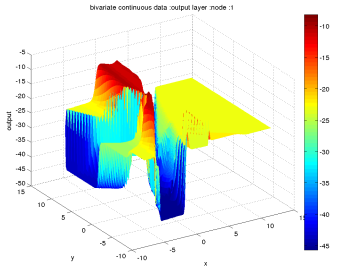


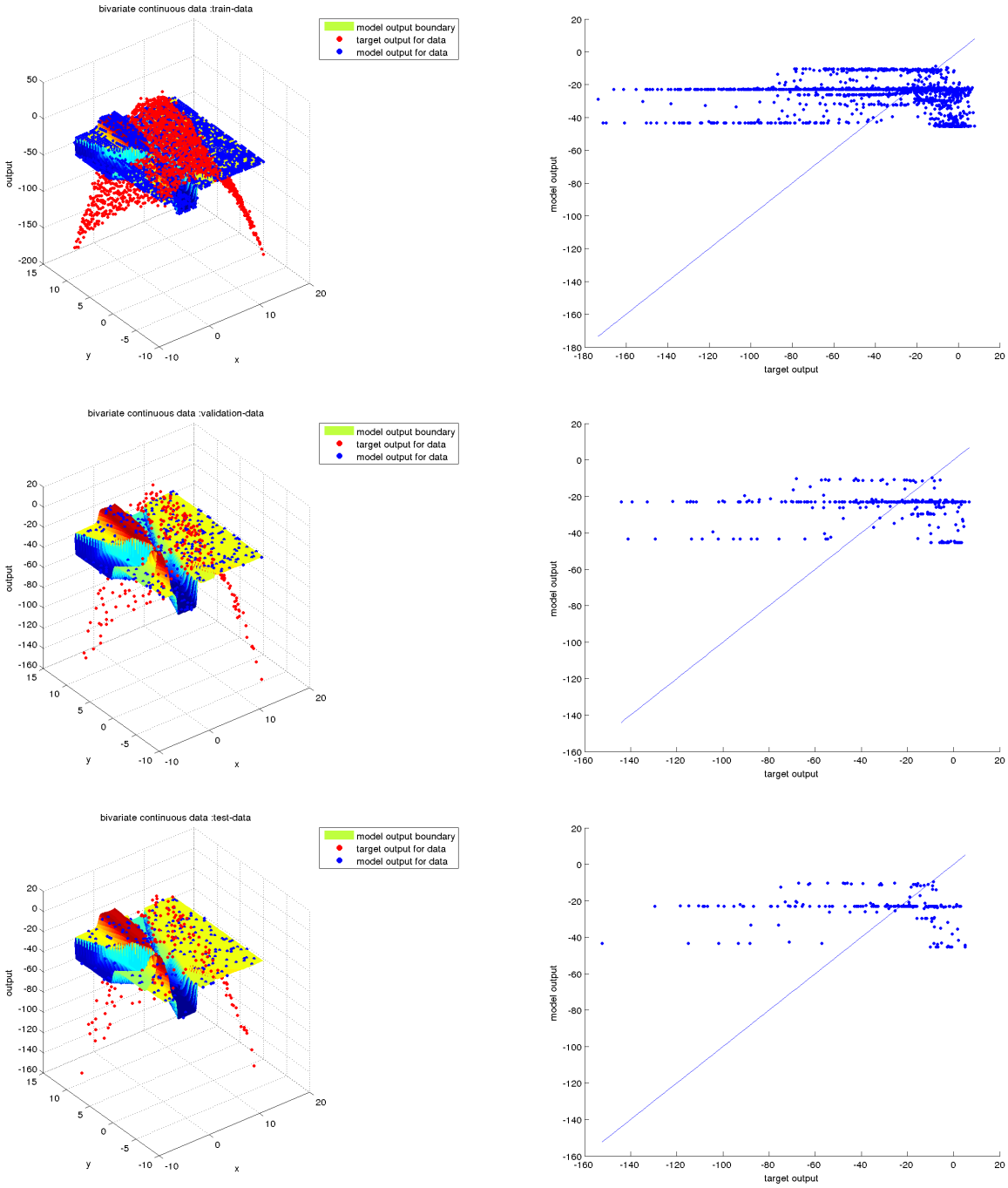




Output layer

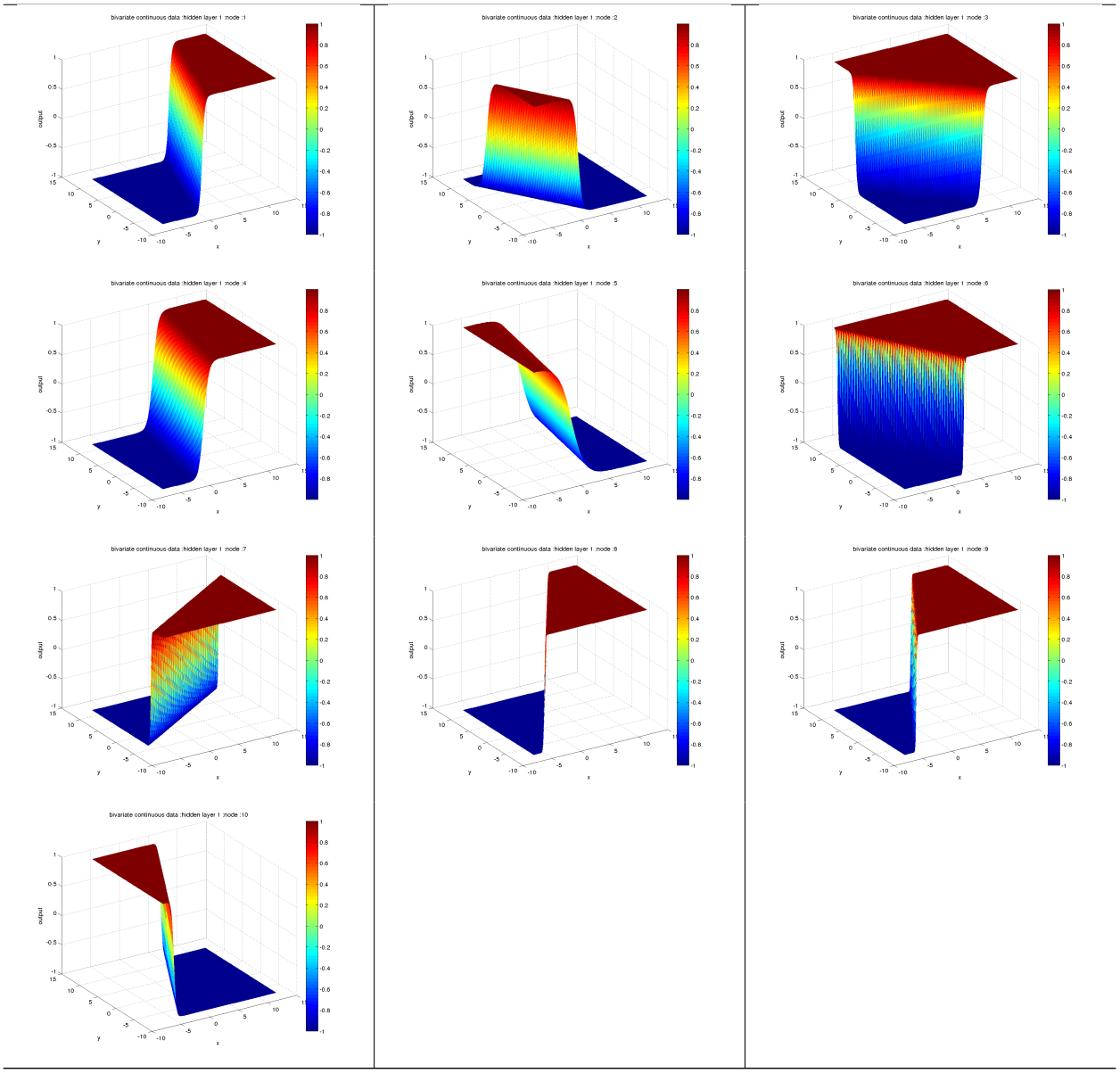


#### 3.3.1.2 Scatter plot with model output and target output

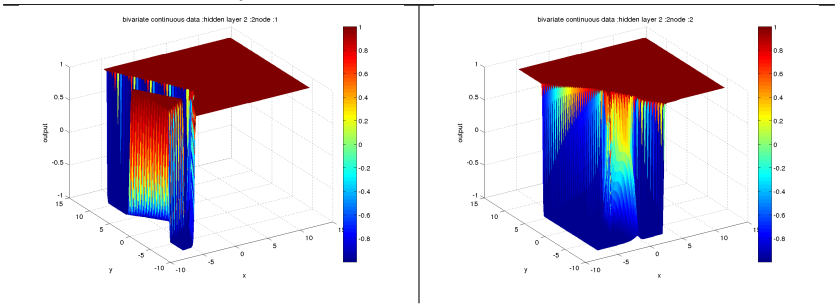


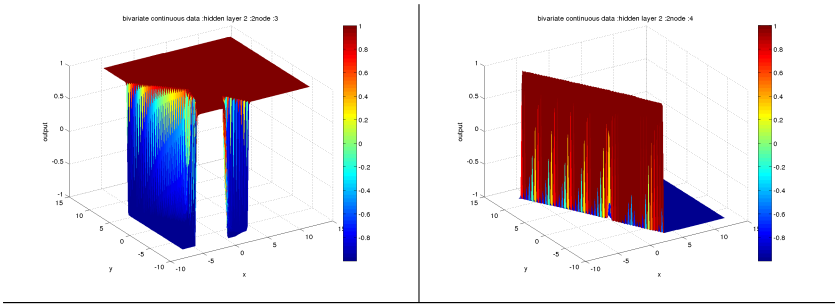

#### 3.3.1.3 After 2 epochs

Hidden layer-1

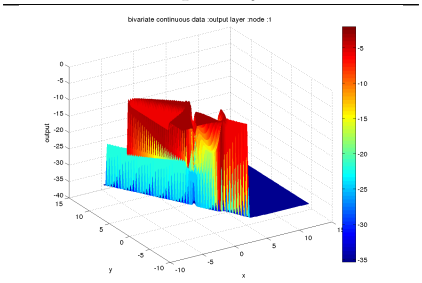



Hidden layer-2

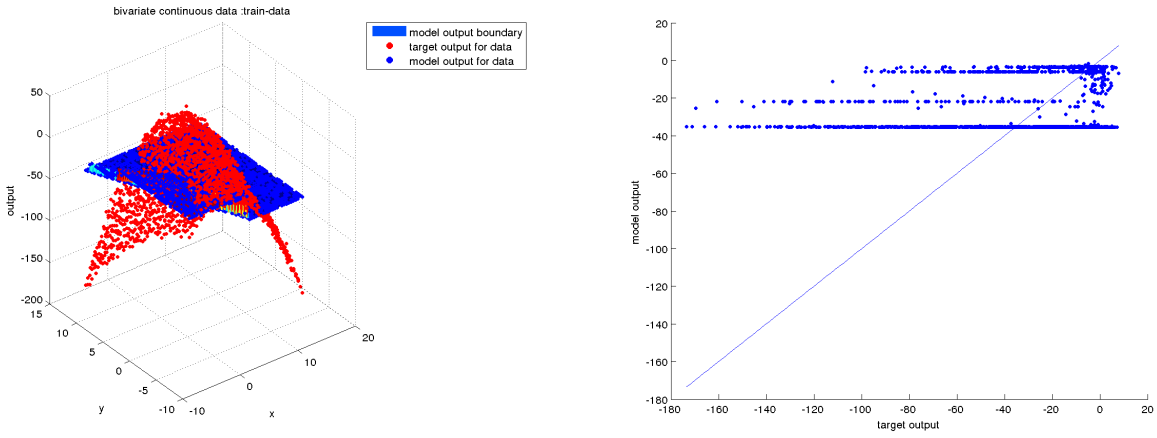


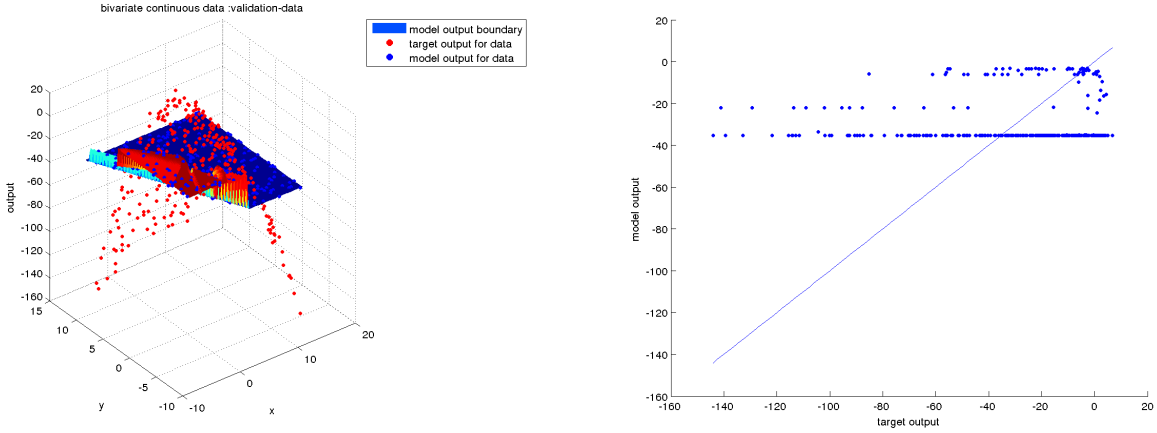


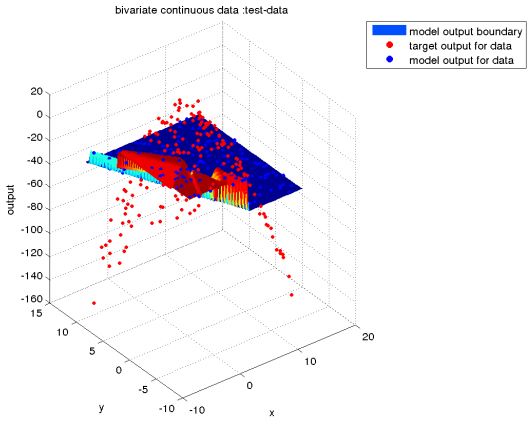

Output layer

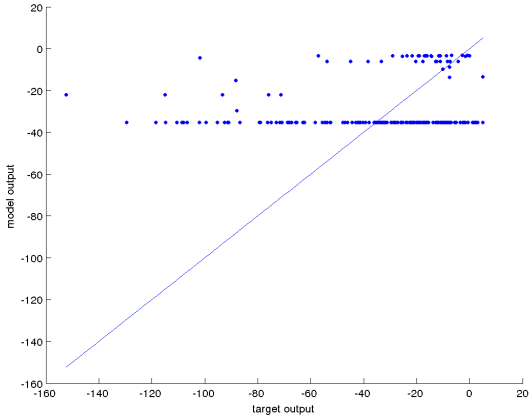



#### 3.3.1.4 Scatter plot with model output and target output



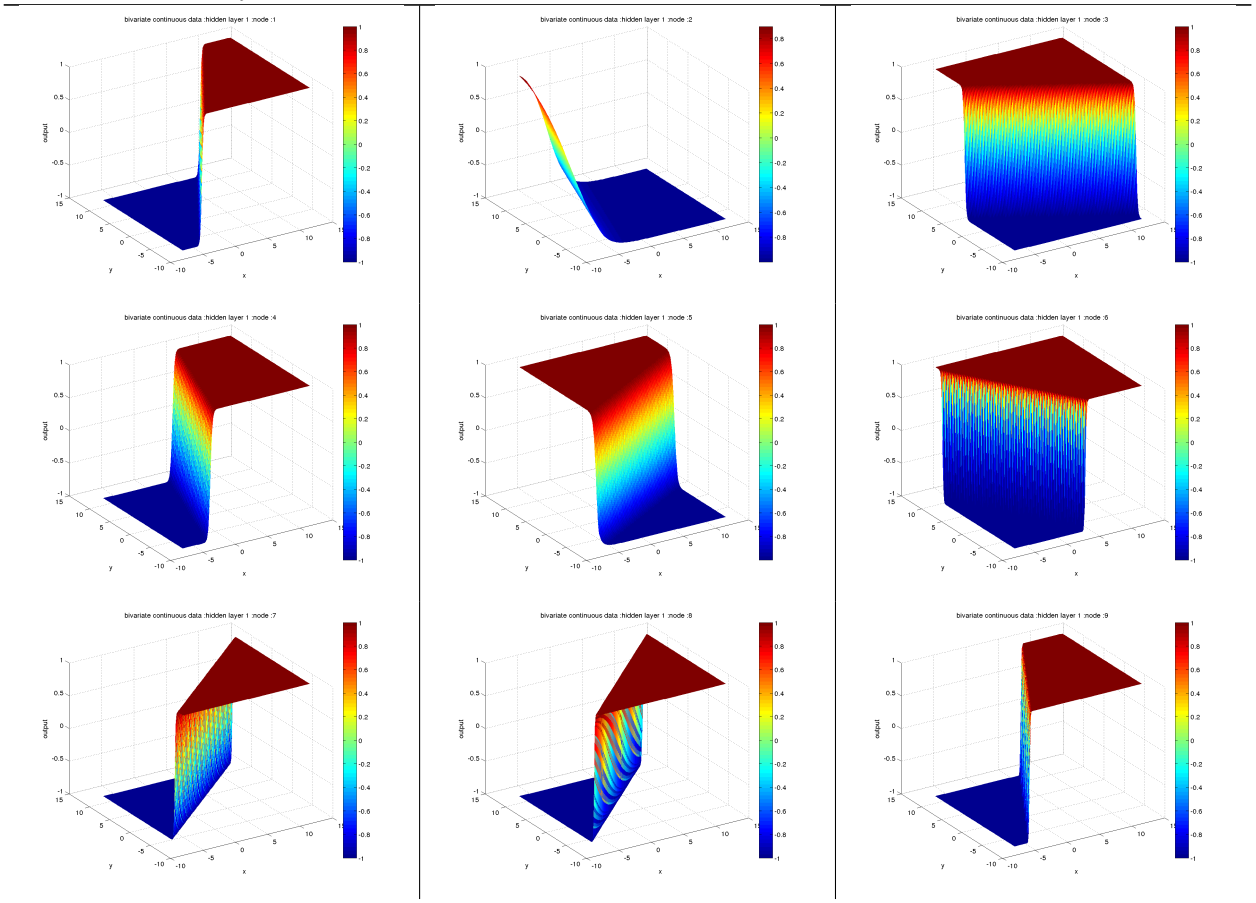



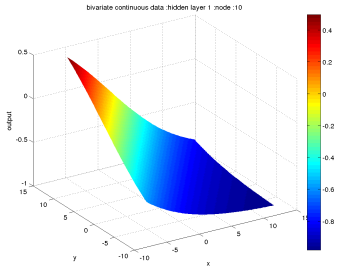


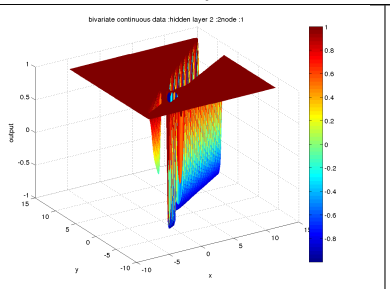

#### 3.3.1.5 After 10 epochs

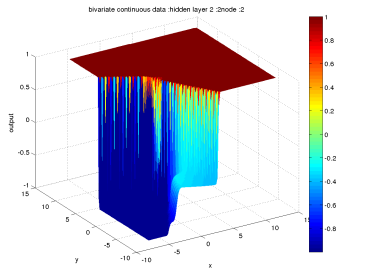
Hidden layer-1

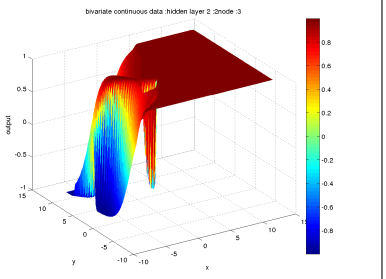


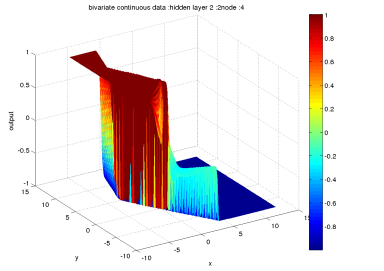



## Hidden layer-2

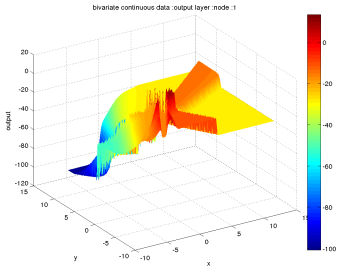


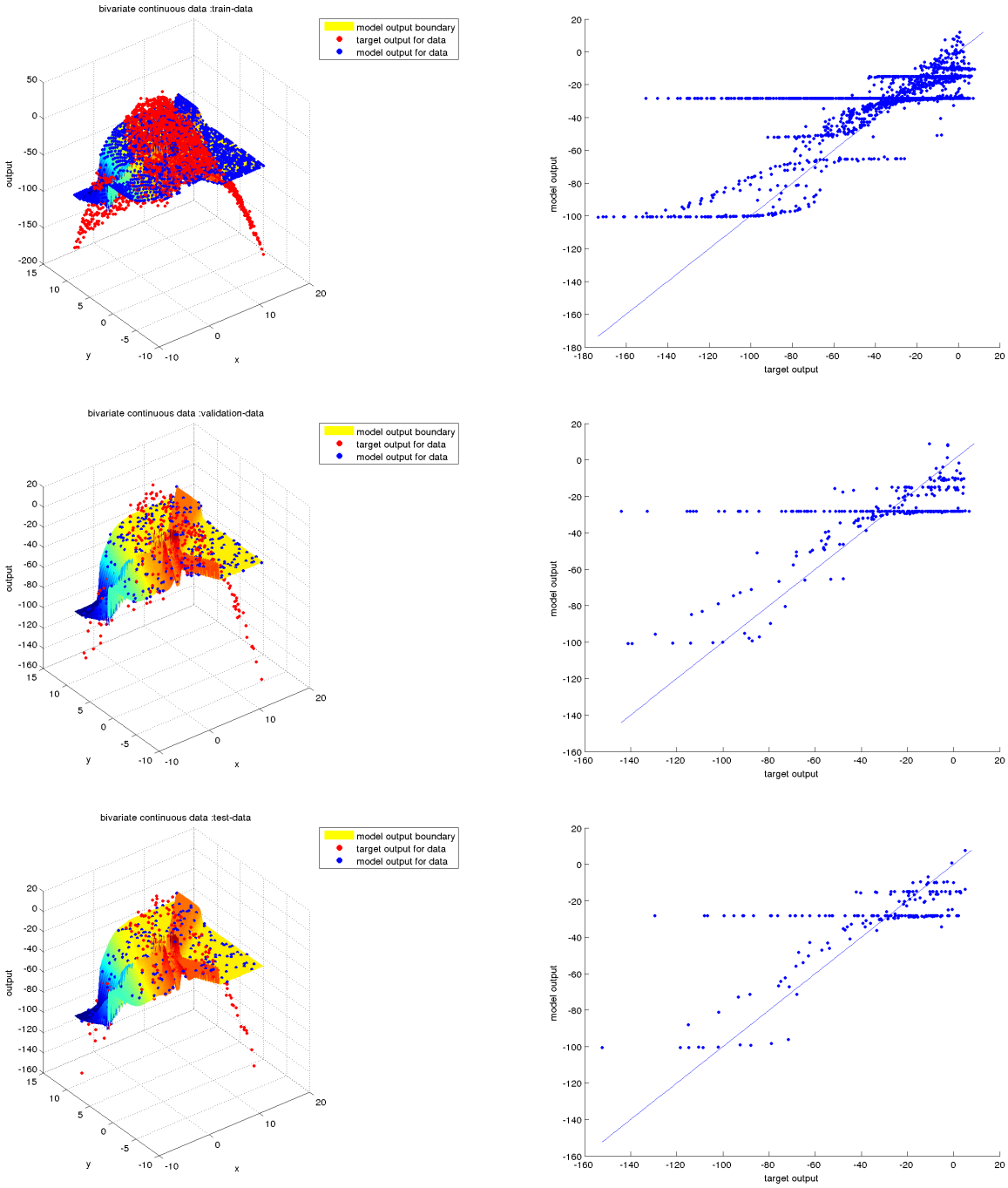




## Output layer

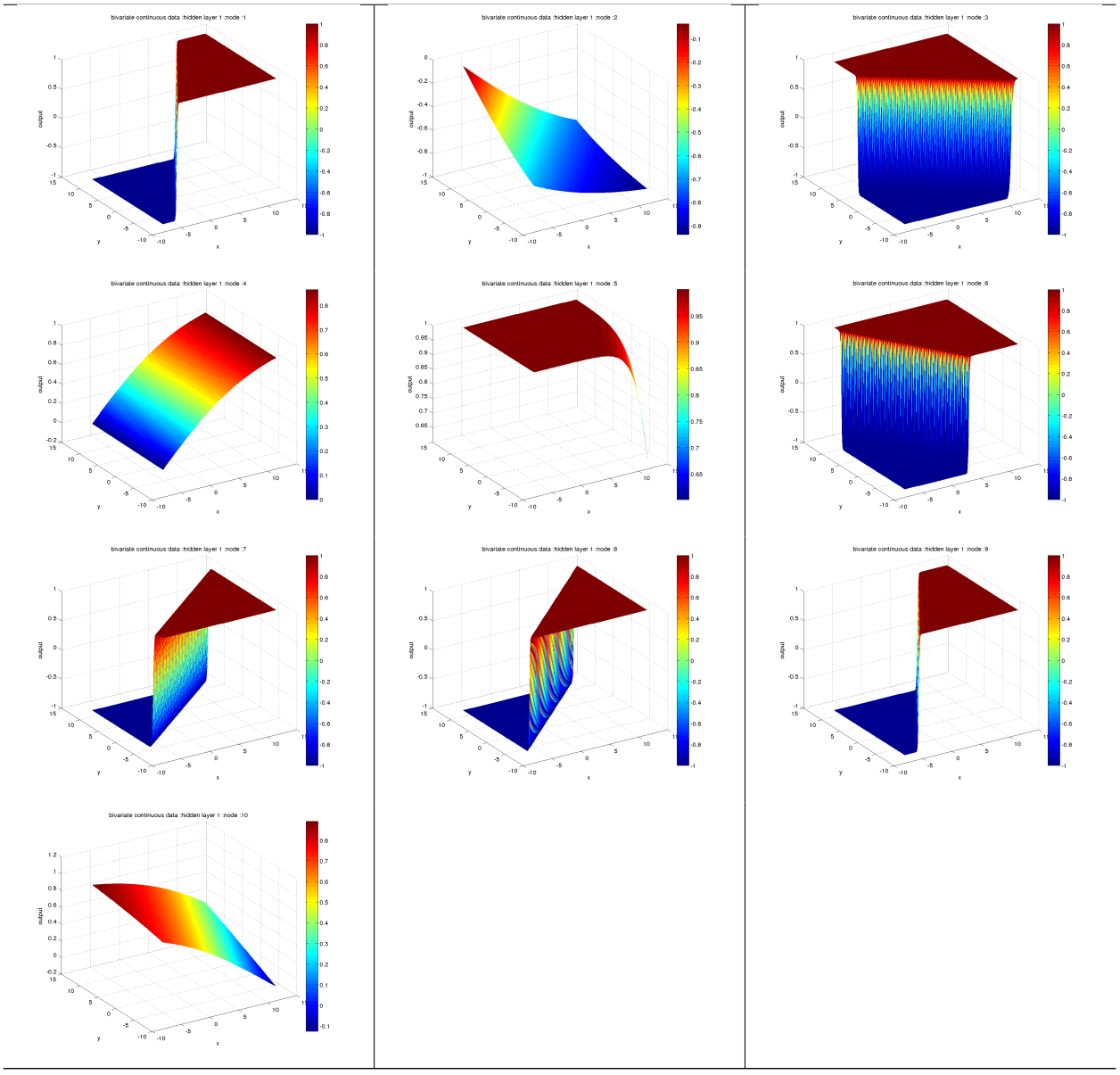


### 3.3.1.6 Scatter plot with model output and target output

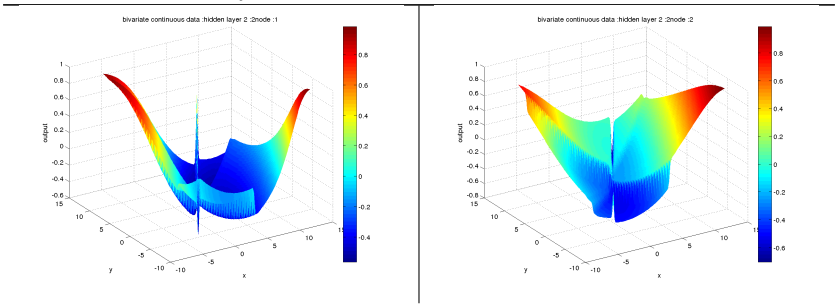


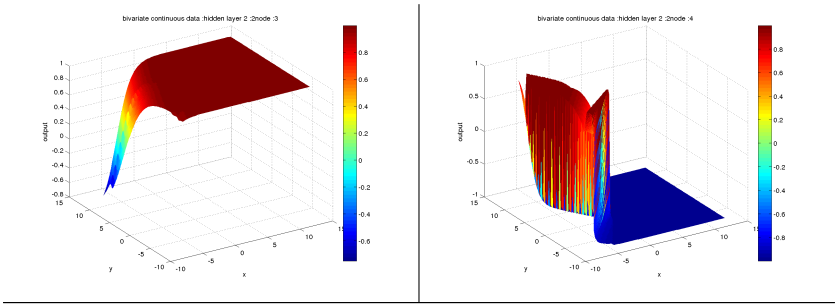

### 3.3.1.7 After 50 epochs

Hidden layer-1

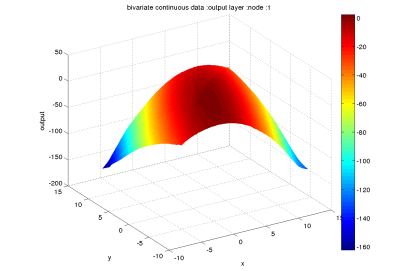



Hidden layer-2

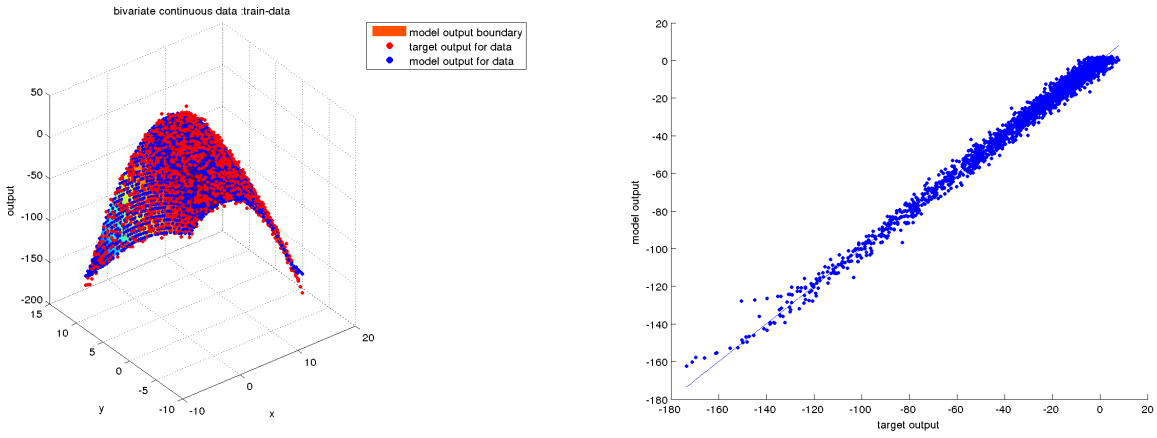


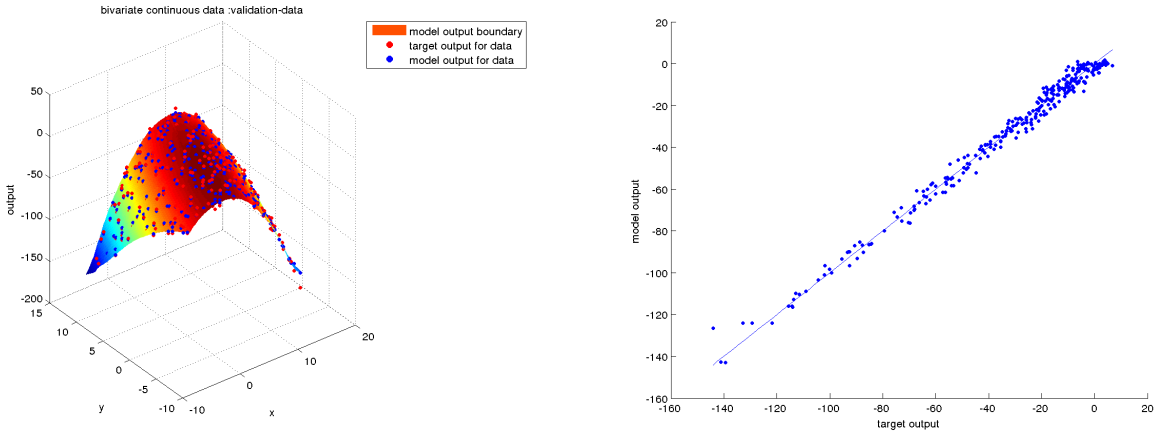


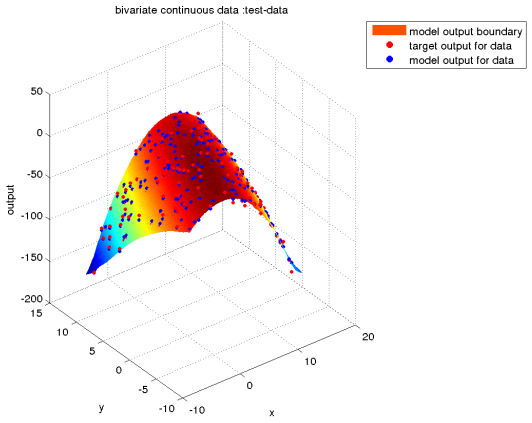

Output layer

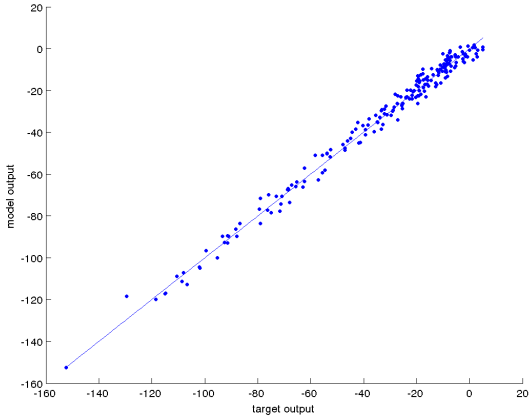



#### 3.3.1.8 Scatter plot with model output and target output



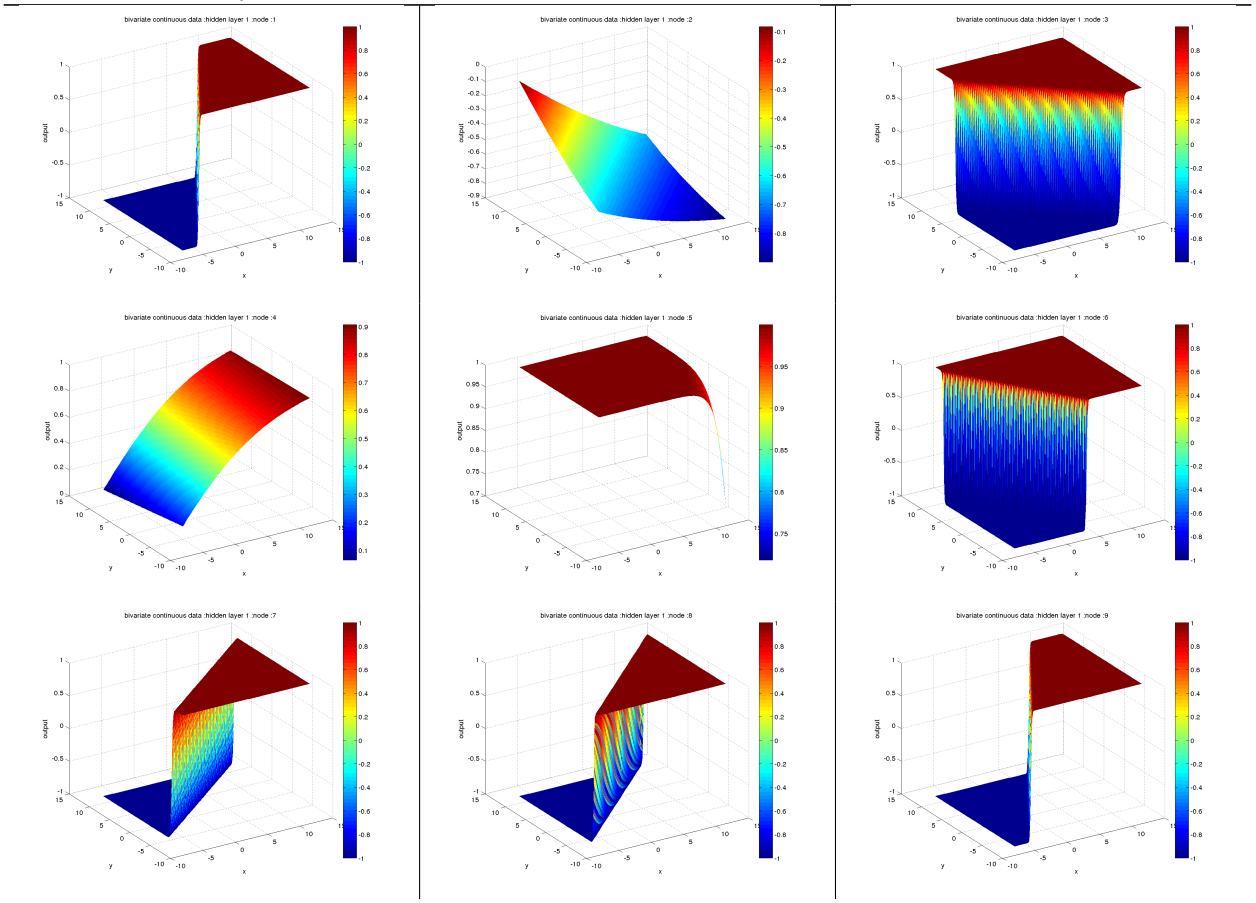



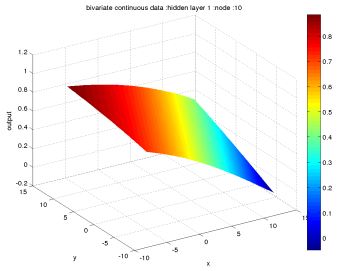


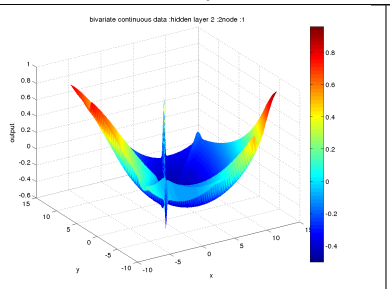

#### 3.3.1.9 After 100 epochs

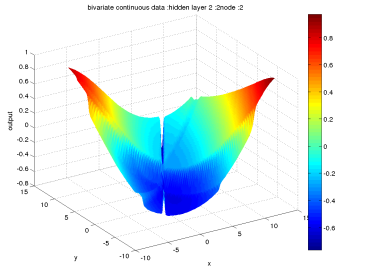
Hidden layer-1

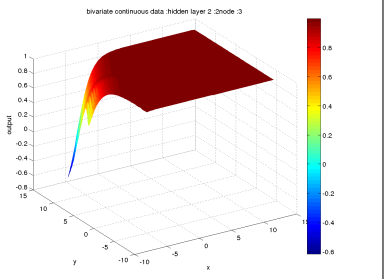


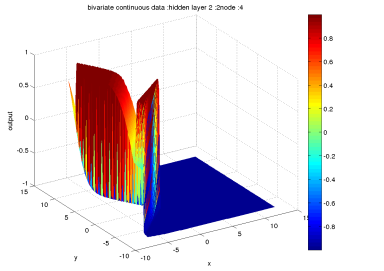



## Hidden layer-2

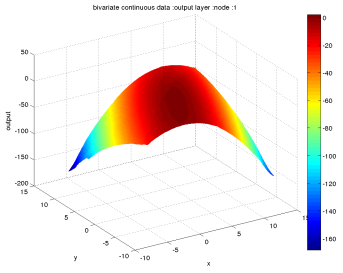


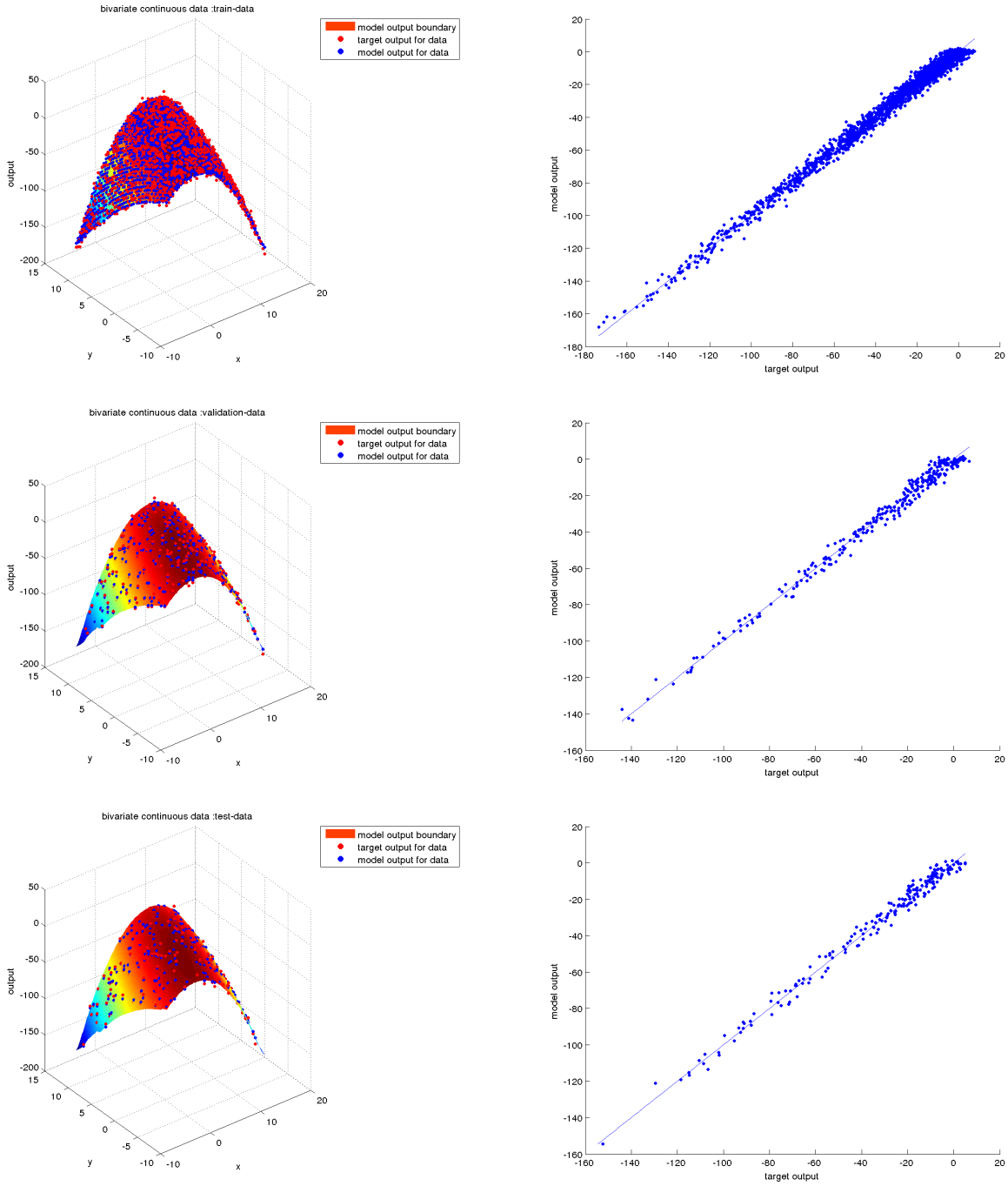




## Output layer

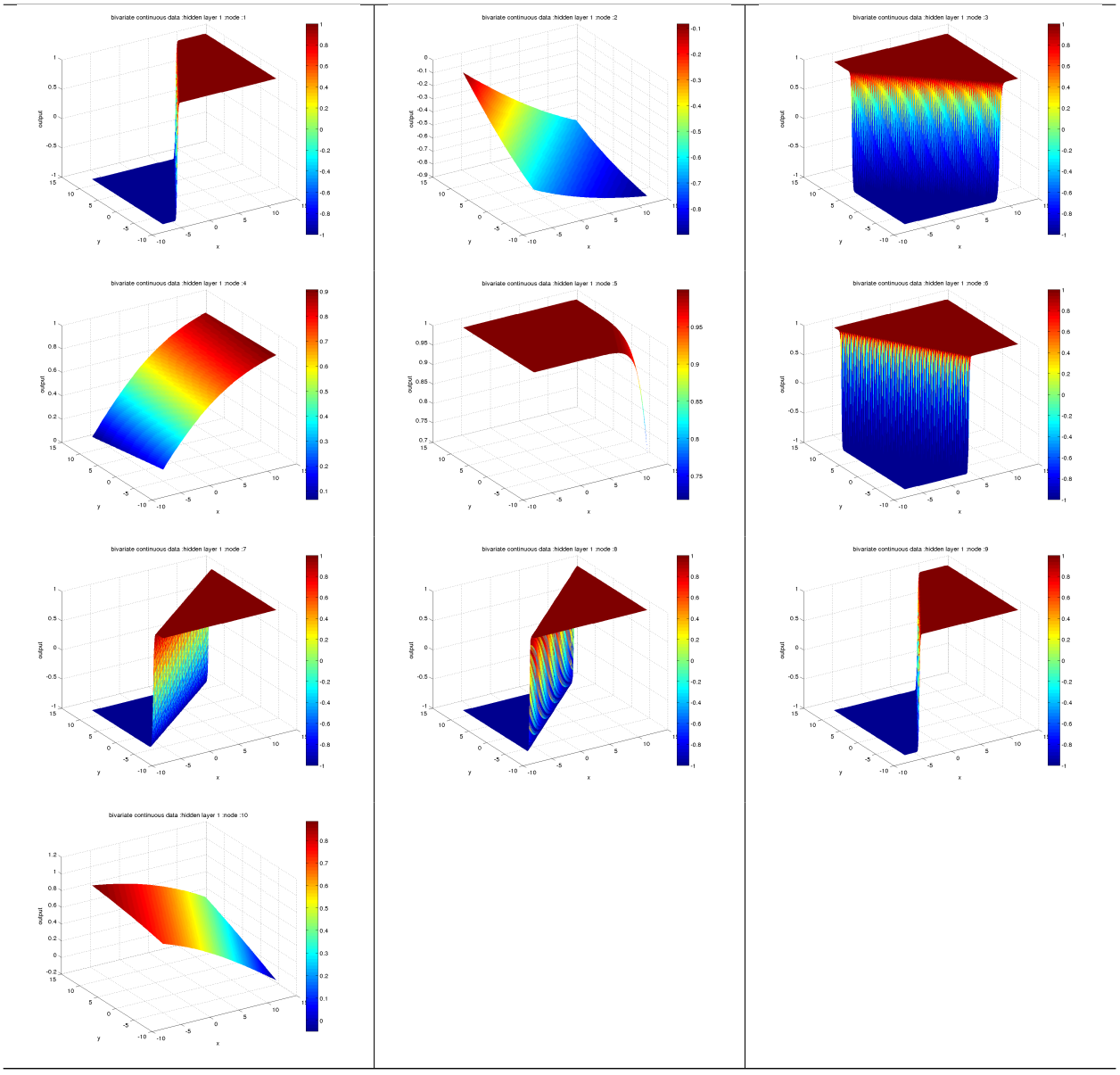


#### 3.3.1.10 Scatter plot with model output and target output

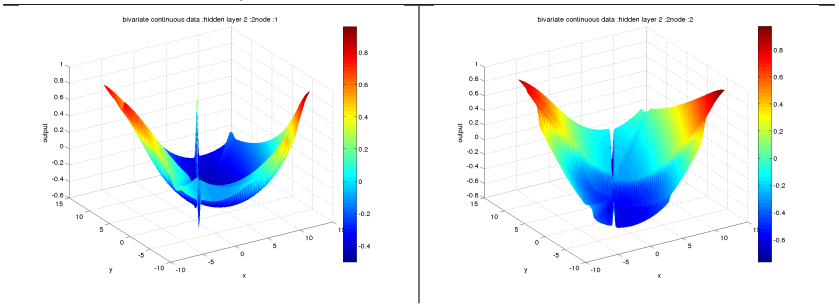


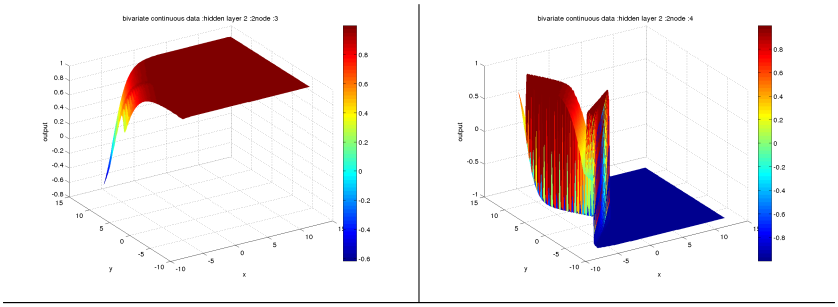

#### 3.3.1.11 After Training

Hidden layer-1

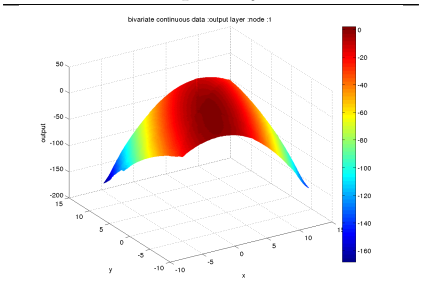



Hidden layer-2

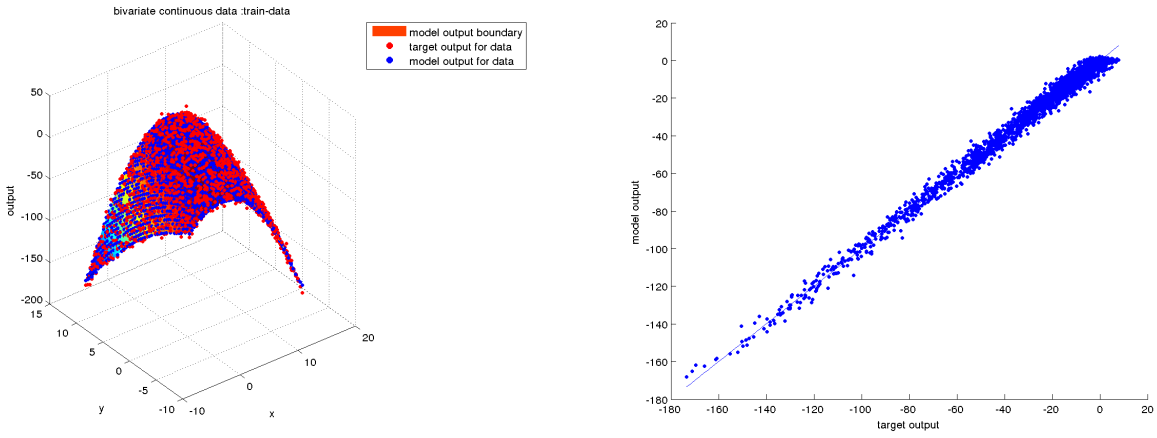


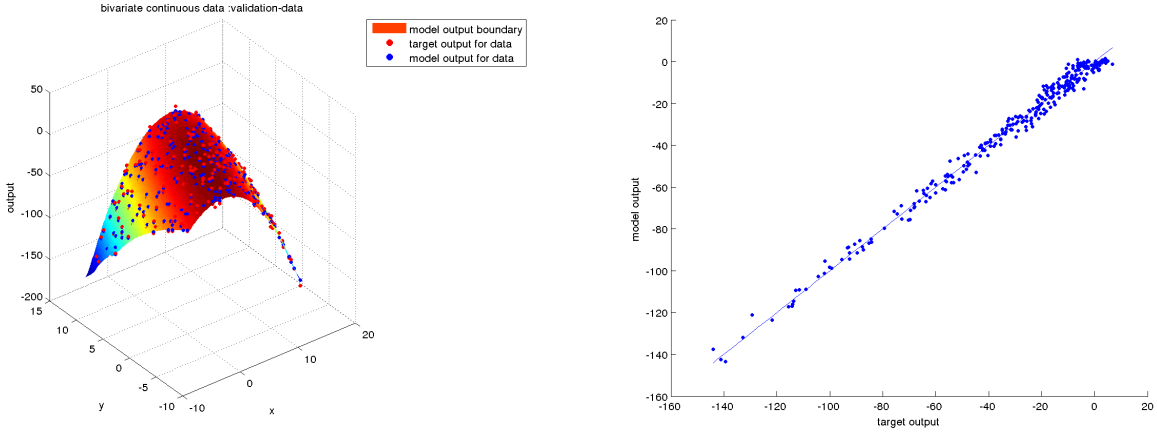


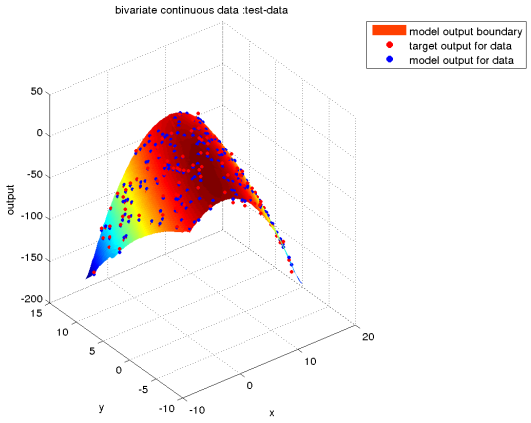

Output layer

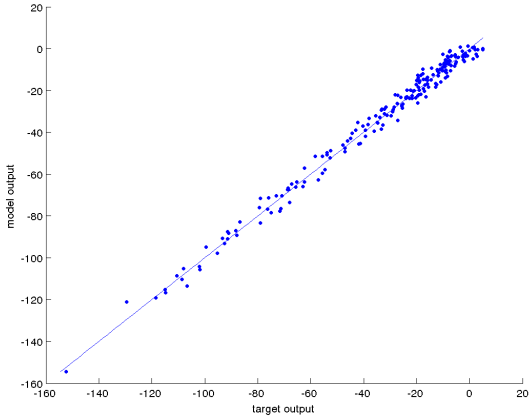



### 3.3.1.12 Scatter plot with model output and target output






#### 3.3.1.13 Observations

- In bivaraite function approximation task, the neural network has 2 hidden layers and it allows the model to involve more non-linearities.
- The best selcted model (hidden layer1 #nodes = 10, hidden layer2 #nodes = 4) seems to perfectly fits the data after full training and the scatter plot of target output, model output shows that the error is very minimum, the obtained result is close to the expected values.