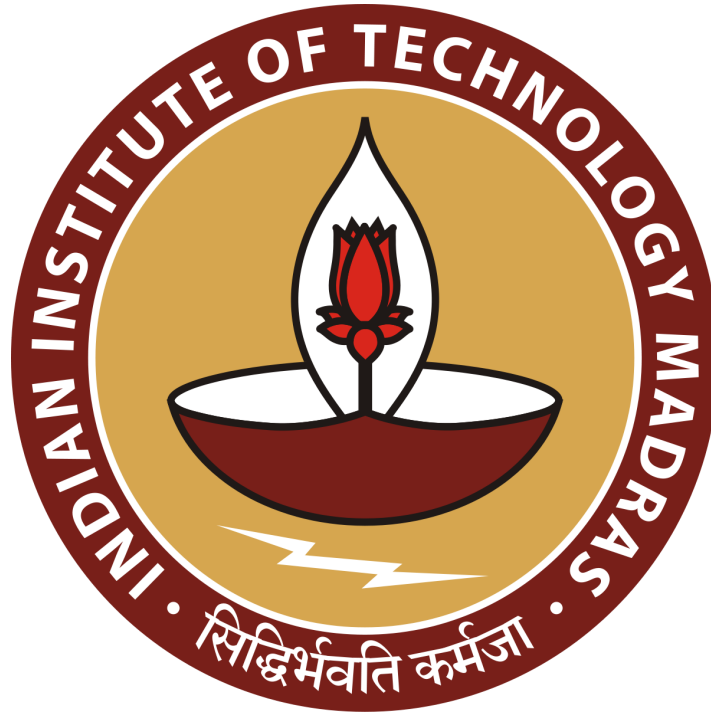


# **PATTERN RECOGNITION AND MACHINE LEARNING**



Indian Institute of Technology Madras  
M.Tech (Computer Science & Engineering)  
SESSION 2018-2019

**SUBMITTED BY:**

OJAS MEHTA - CS18M038  
PRANAV MURALI - CS18M041  
VEDANSH GURUNATHAN - CS18M058

**SUBMITTED TO**

Prof. C. CHANDRA SEKHAR  
IIT Madras

**GROUP NUMBER : 5**

## **TASK 1 : EIGEN ANALYSIS**

### **DATASET 1 : 64 x 64 GRAY LEVEL IMAGE DATA**

#### **1. METHOD :**

Eigen Faces seeks to implement a system capable of efficient, simple and accurate face reconstruction in constrained environment.

Each 64 X 64 image is represented as a vector of 4096 X 1 size. As input has 15 folders and in each one of them there are 10 images we have total of 150 images where we can reconstruct a matrix of size 4096 X 150, each column representing an image.

Let the image to be reconstructed be represented as X.  
Find the covariance matrix as

$$\text{cov}(X) = \frac{\sum_{i=1}^n (X_i - \bar{x})(X_i - \bar{x})^T}{n - 1}$$

Find the Eigenvalues and their corresponding Eigenvectors of the covariance matrix. All the Eigenvectors of the covariance matrix are not necessary.

Have a pre computed array of eigenvectors sorted in nondecreasing order of eigenvalues.

Iterate for the number of EigenVectors and repeat the below procedure

- Fetch the corresponding eigenvector from the calculated eigenvalue.
- Dot product of the EigenVector with the image to be reconstructed.
- The output of (b) will give a scalar value, multiply it by the eigenvector
- Add the output of (c) step into a final array.

Now reshape the final array into 64 X 64 (image size) and display it.









































#### **2. PLOT/RESULTS**

These are the images that are reconstructed in our experiment



Below are the outputs of each image, after considering different number of Eigenvalues.

**Fig 1.1:- Reconstructed images**

# of Eigenvalues	Folder: 6 Image: 10	Folder: 12 Image: 6	Folder: 10 Image: 3	Folder: 16 Image: 2	Folder: 21 Image: 3
1					
10					
20					
40					
80					
160					
320					
640					

### 3. INFERENCES

- a. As the number of eigenvector increases, we are able to represent more features accurately. Largest eigenvalues and their corresponding eigenvectors can be used to represent most of the features of the image. With only 320(7% of total 4096 eigenvectors) largest eigenvectors we are able to retrieve most of the information.
- b. We are converting the image vector  $64 \times 64$  into  $4096 \times 1$ , which can be said as we converted the image into 4096 dimensional space and the axes of this are perpendicular to each other.
- c. Each of the point or feature in 4096 dimensional space is projected on to the eigenvector considered, which gives us the weight(scalar value).
- d. If we define this vector as  $\bar{v}$ , then the projection of our data  $D$  onto this vector is obtained as  $\bar{v}^T D$  and the variance of the projected data is  $\bar{v}^T \sum \bar{v}$ . Since we are looking for the vector  $\bar{v}$  that points into the direction of the largest variance, we should choose its components such that the covariance matrix  $\bar{v}^T \sum \bar{v}$  of the projected data is as large as possible. Maximizing any function of the form  $\bar{v}^T \sum \bar{v}$  with respect to  $\bar{v}$ , where  $\bar{v}$  is a normalized unit vector, can be formulated as the largest eigenvector of the covariance matrix.
- e. In other word, largest eigenvector of the covariance matrix always points in the direction of largest variance of the data, and the magnitude of this vector equals to the corresponding eigenvalue. Second largest eigenvector is perpendicular to the largest one.
- f. The eigenvector corresponding to the maximum eigenvalue contains significant information content and are sufficient to reconstruct with fair amount of accuracy.

## **TASK 2:- REGRESSION MODELS**

### **1) Polynomial curve fitting for Dataset-2 (Univariate data)**

#### **1. Method**

In this section following notations are used:-

- 1) **x**:- Input data.
- 2) **y(x,w)**:- Model output.
- 3) **t**:- System output.
- 4) **w**:- coefficient vector.

In polynomial curve fitting, we fit the data using the polynomial function of the form:-

$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Mx^M$$

Where **M** is the order of the polynomial (Hyper-parameter of the model) .

$w_0, w_1, \dots, w_M$  are polynomial coefficients which are collectively denoted by vector **w**.

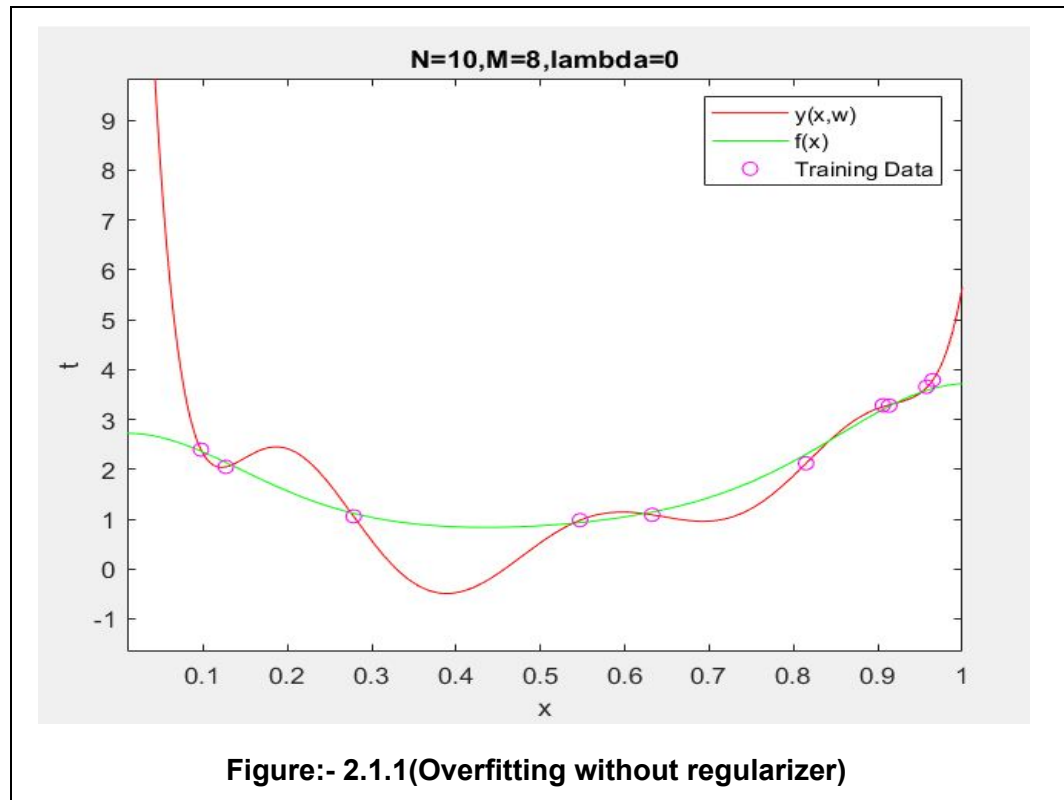
The value of the coefficients are determined by fitting the polynomial to the training data. This can be done by **minimizing** an error function **E(w)** which measures the misfit between the function  $y(x, \mathbf{w})$  for any given value of **w**, and the training set data points.

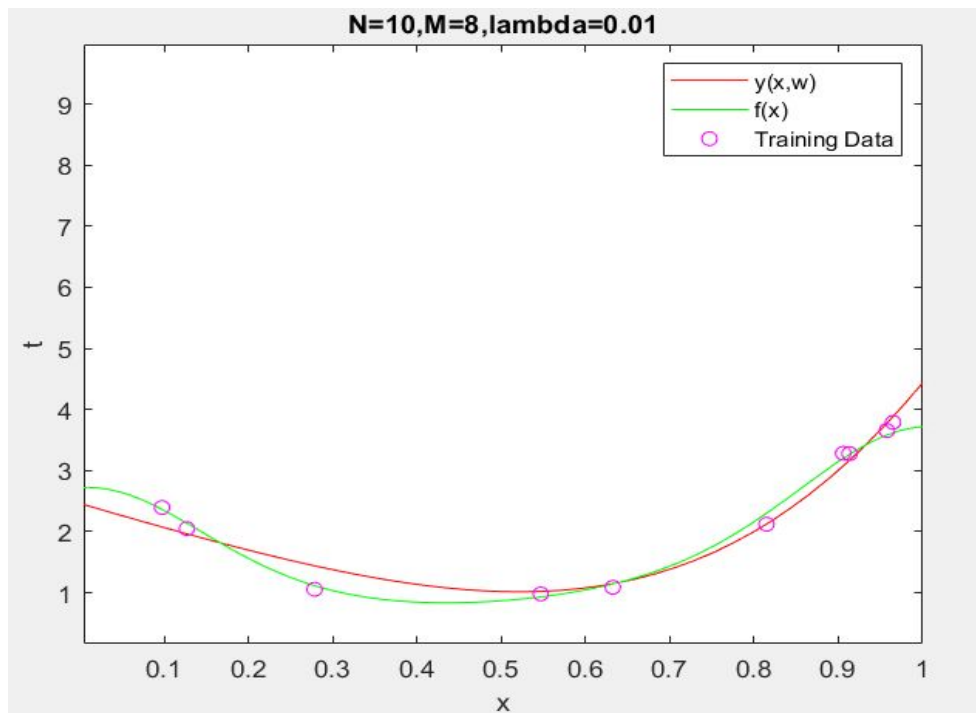
$$E(w) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, w) - t_n\}^2$$

Function for data generation:-

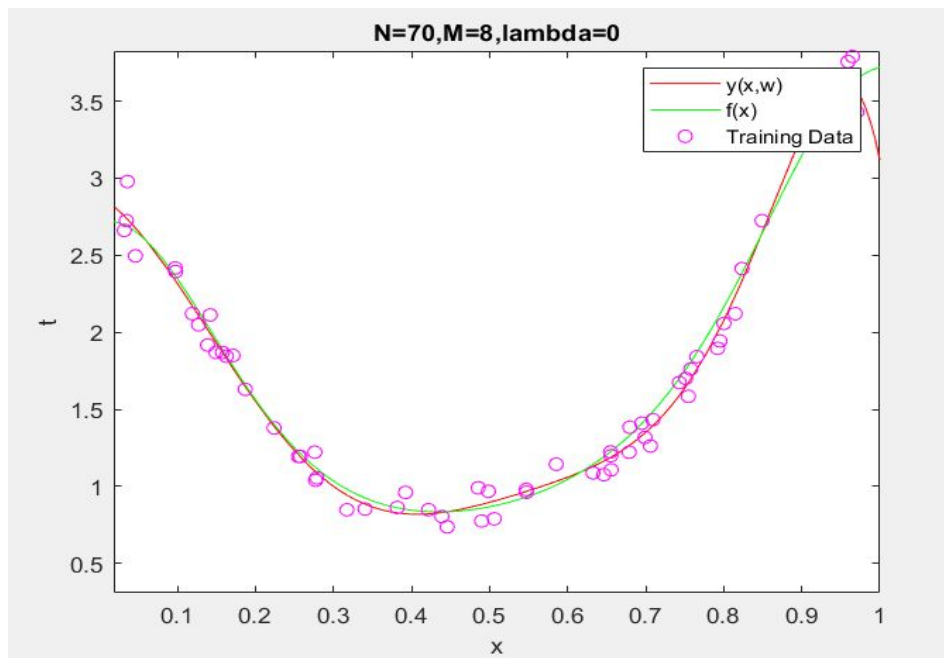
$$\mathbf{F}(\mathbf{x}) = e^{\cos(2\pi \mathbf{x})} + \mathbf{x}$$

## 2. Plots

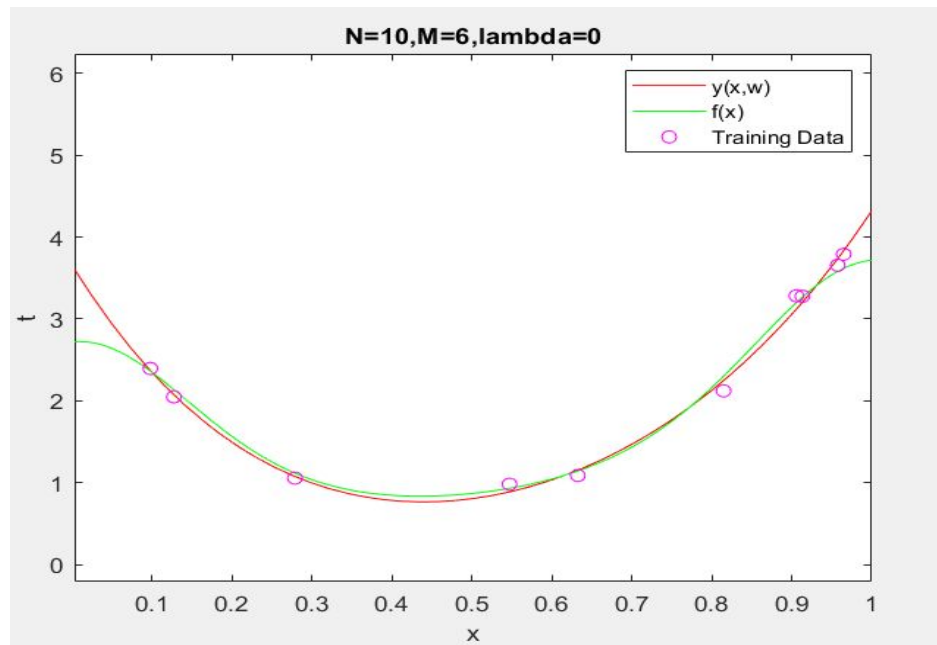




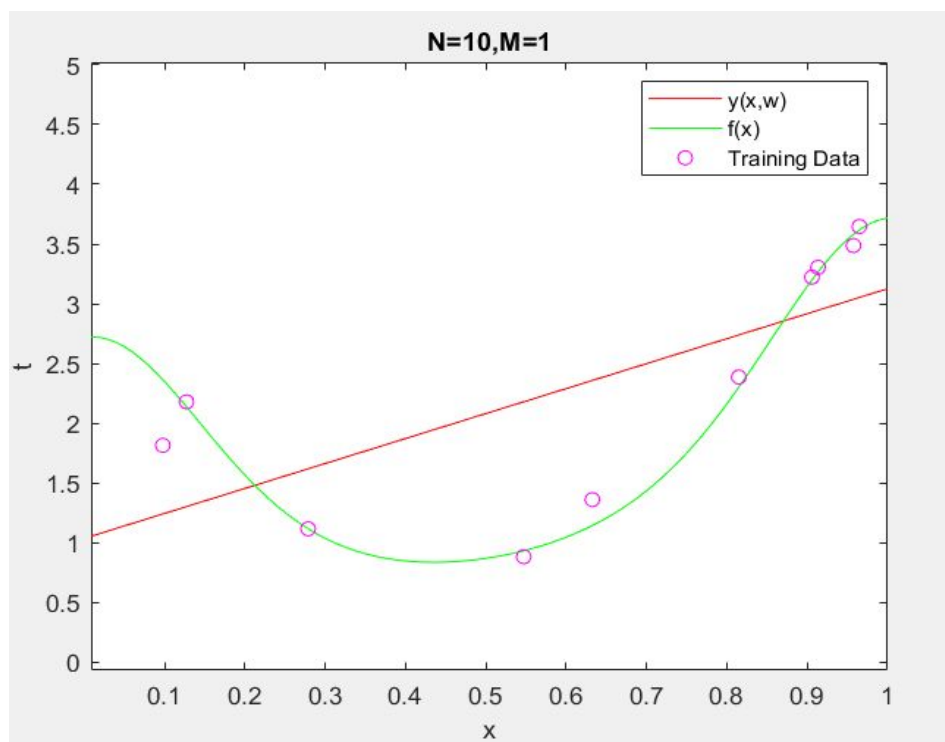
**Figure:- 2.1.2 (Overfitting reduced with  $\lambda=0.01$ ).**



**Figure:- 2.1.3  
(Overfitting reduced with  $N=70, M=8$  and  $\lambda=0$ ).**

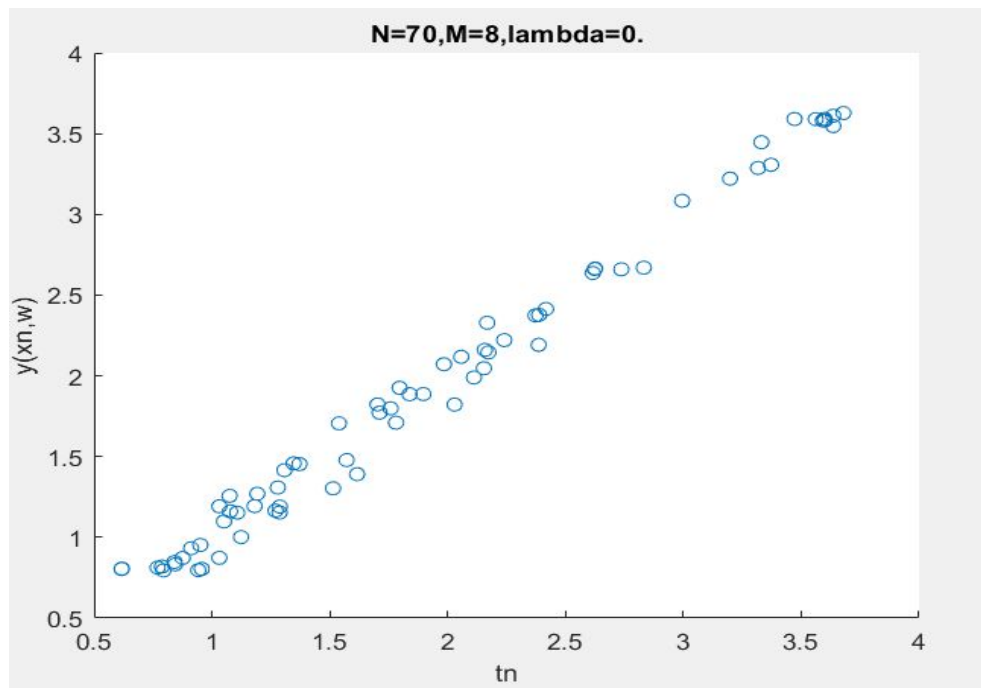


**Figure:- 2.1.4 (Best fit).**

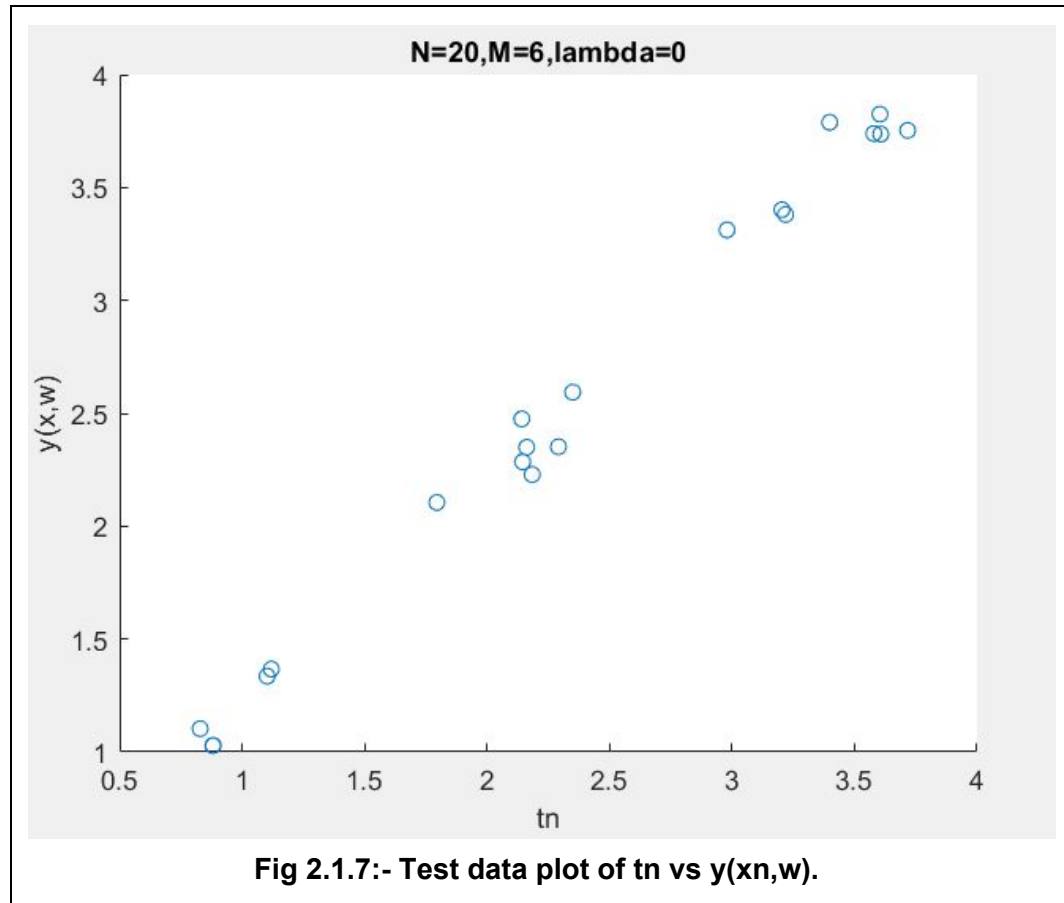


**Fig 2.1.5:- Underfitting**





**Fig 2.1.6:- Training data plot of  $t_n$  vs  $y(x_n, w)$ .**



### 3.RESULTS

**Fig 2.1.8:- Training Data (N=10).**

	Dataset 2 ERMS		
	Training Data		
	$\lambda=0$	$\lambda=25$	$\lambda=100$
M=0	0.9464	0.7288	1.1482
M=1	0.735	1.0186	1.3606
M=5	0.0908	0.6841	0.996
M=6	0.0867	0.7047	0.9762
M=8	0.0789	0.7172	1.0915

**Fig 2.1.9:-Test Data**

	Test Data(N=20)		
	$\lambda=0$	$\lambda=25$	$\lambda=100$
M=0	0.9216	0.7995	1.167
M=1	0.8076	0.9009	1.1947
M=5	0.1203	0.7551	0.9406
M=6	0.0889	0.6462	1.1267
M=8	0.1824	0.1022	0.9606

**Fig 2.1.10:- Parameters for different value of degree of polynomial (M)**

		M=8	
	$\ln(\lambda)=-100$	$\ln(\lambda)=-18$	$\ln(\lambda)=0$
w0*	3	2.7257	1.921
w1*	-5	2.0927	-1.5602
w2*	-50	-74.4946	-0.1673
w3*	-867	223.3093	0.5296
w4*	5143	-184.2926	0.7629
w5*	-15404	-99.2255	0.7822
w6*	26870	116.6562	0.7095
w7*	-28555	114.6903	0.6018
w8*	18289	-33.7042	0.4858
w9*	-6506	-108.7824	0.3737

**Fig 2.1.11:- Change in Parameters with change in regularization constant(M=8)**

	M=0	M=1	M=6	M=8
w0*	1.7529	1.2577	2.6656	2
w1*		0.9043	3.7136	18
w2*			-105.3403	-376
w3*			421.0685	2724
w4*			-744.2982	-11152
w5*			633.8048	27868
w6*			-207.7526	-42884
w7*				39487
w8*				-19885
w9*				4199

### **3. INFERENCES**

1. In Fig 2.1.4, degree of polynomial (M) is 6. Curve with M=6 and regularization constant  $\lambda=0$ , perfectly fits the data with less error.
2. In Fig 2.1.1, degree of polynomial (M) is 8 and  $\lambda=0$ . Curve with M=8, overfits the data. As the polynomial degree is very high and value of regularizer is low, it also fits the noise and hence the prediction is not very accurate.
3. In Fig 2.1.6, degree of polynomial (M) is 8 and  $\lambda=0.01$ . Here even though the degree of polynomial is high, the curve is smoother when regularizer value is changed.
4. From Fig 2.1.3, we see that increasing the value of N from 10 to 70 has reduced the over-fitting problem as we train the model (with M=8) on more number of examples. Also, it follows the rule that number of examples should be at least 5-10 times the value of M.
5. Fig 2.1.6 and Fig 2.1.7 shows the plot of target output v/s model output. We can see that predicted output is almost same as target output.
6. From Fig 2.1.8 and Fig 2.1.9, we can see that the rms error decreased as the model complexity was increasing till M =6. But on further increase in the complexity, although the model was perfectly fitting the training the data, the error on test data was increased. This is overfitting

## 2) Linear model for regression using polynomial basis function (Dataset-2)

### 2.2.1) Method

In this section, the following notations are used

$\bar{x}$  - Vector x, the training data

$\bar{w}$  - Model parameters, coefficient

$\Phi$  - Design matrix

N - Dataset Size

$X^T$  - Transpose of the matrix X or vector x

In Linear model for regression we are following the below steps:

- a. Calculating the expression of x and y up to the degree of the model.

$$\text{Expression} = (x + y)^0 + (x + y)^1 + (x + y)^2 \dots + (x + y)^M$$

For eg the expression with degree 3 will be the following

$$\text{Expression} = x^3 + 3x^2y + x^2 + 3xy^2 + 2xy + x + y^3 + y^2 + y + 1$$

- b. Evaluate the design matrix as

$$\Phi = \phi(\bar{x}_n)^T$$

- c. Calculate the pseudo-inverse of the design matrix and add the regularization term to the Design matrix.

$$\bar{w} = ((\Phi^T \Phi) + \lambda I) \Phi^T \bar{t}$$

- d. From the coefficient and the expression evaluated, the model is built  
 $\lambda$  is regularization constant.

## 2.2.2) Plots

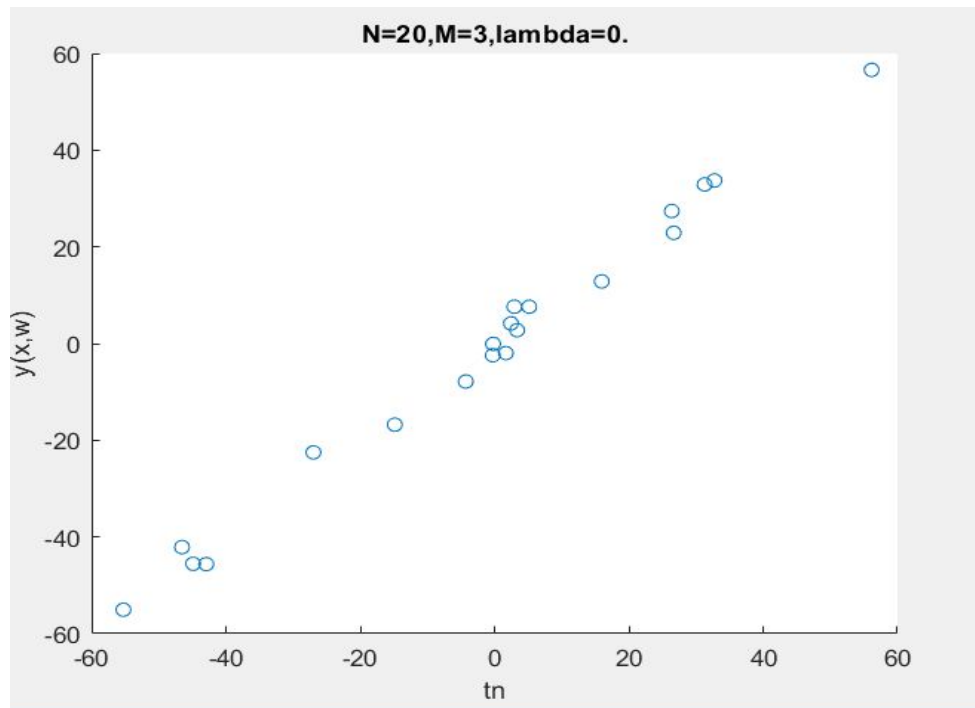


Figure 2.2.1:- Training Data plot for  $t(n)$  versus  $y(x,w)$ .

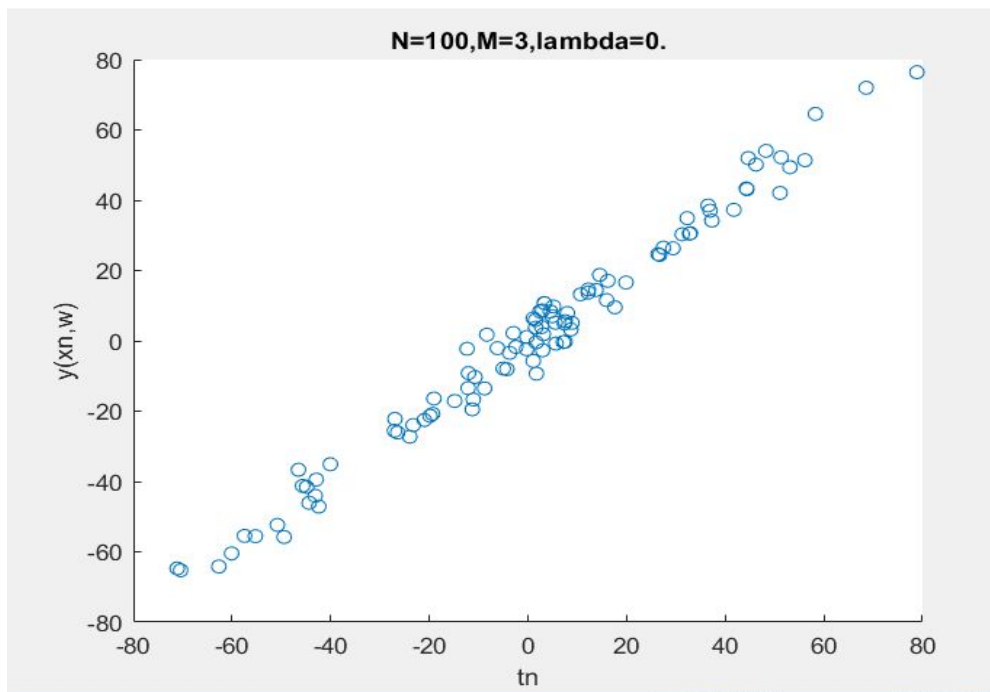


Fig 2.2.2:- Training data plot of  $t_n$  versus  $y(x,w)$ .

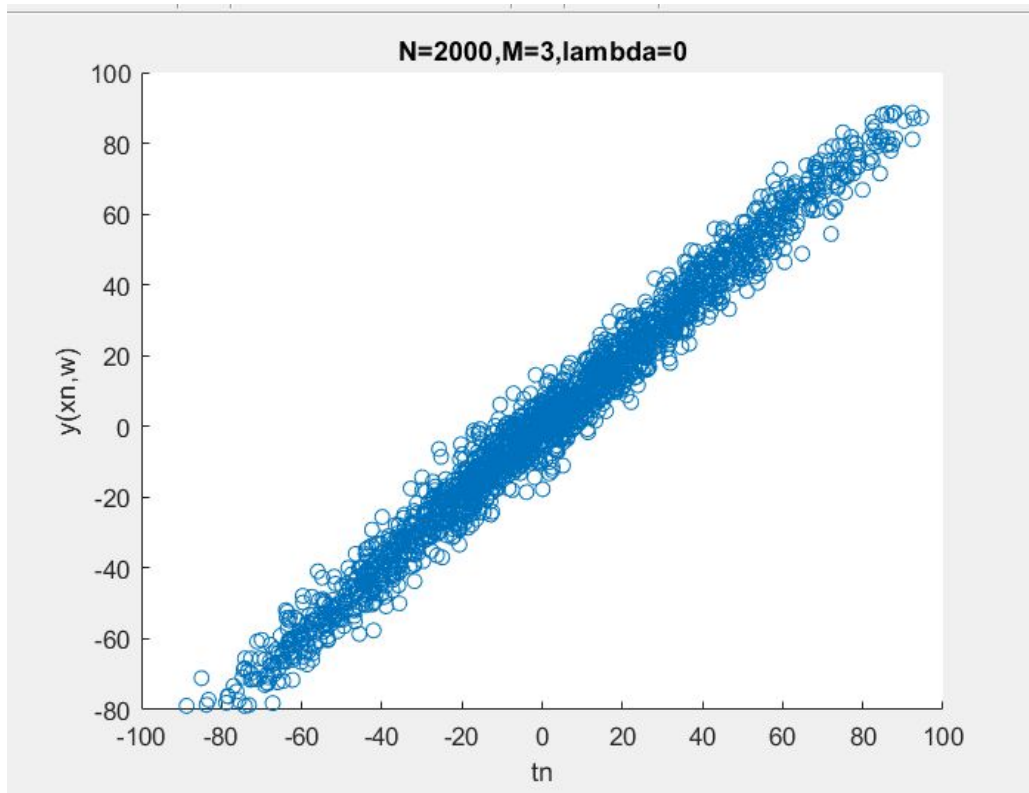


Figure 2.2.3:- Training Data plot for  $t(n)$  versus  $y(x, w)$ .

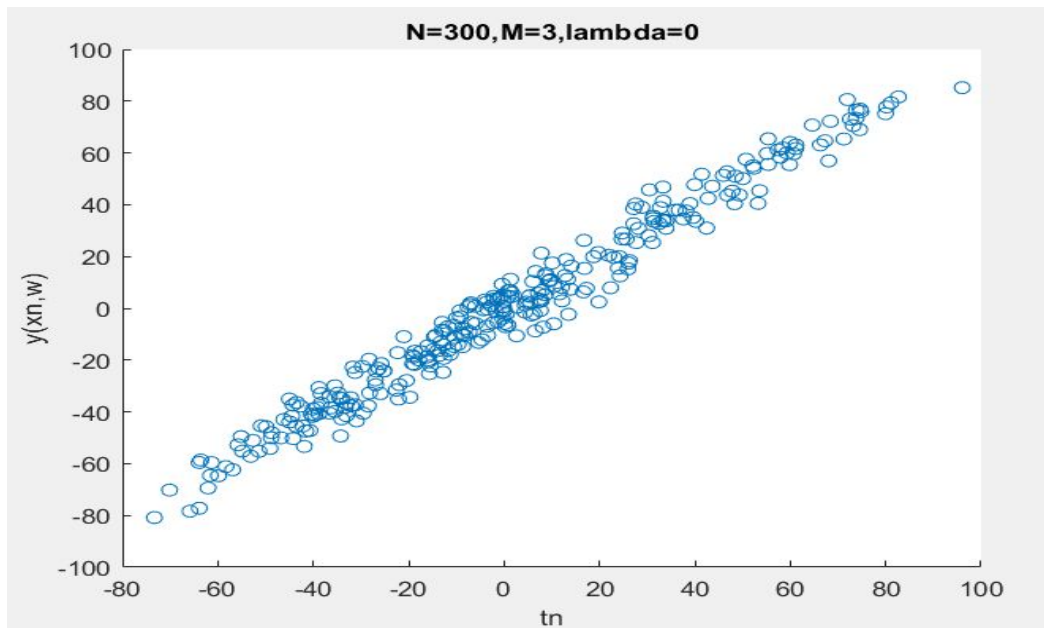


Figure 2.2.4:- Validation data plot for  $t(n)$  versus  $y(x, w)$ .

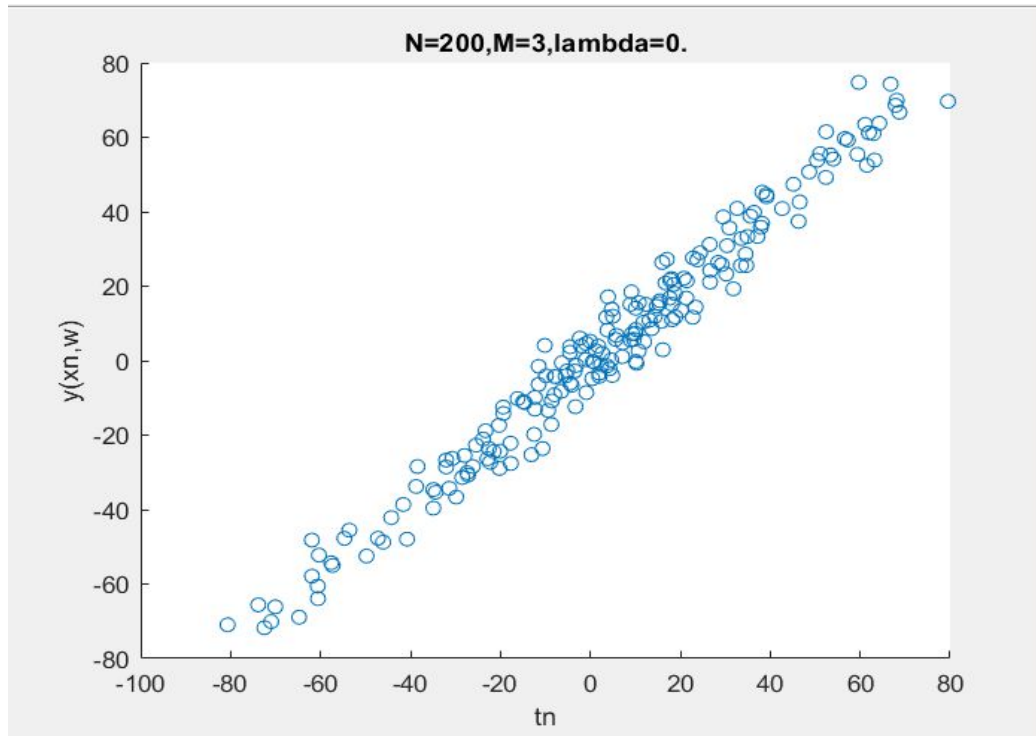


Figure 2.2.5:- Test data plot for  $t(n)$  versus  $y(x, w)$ .

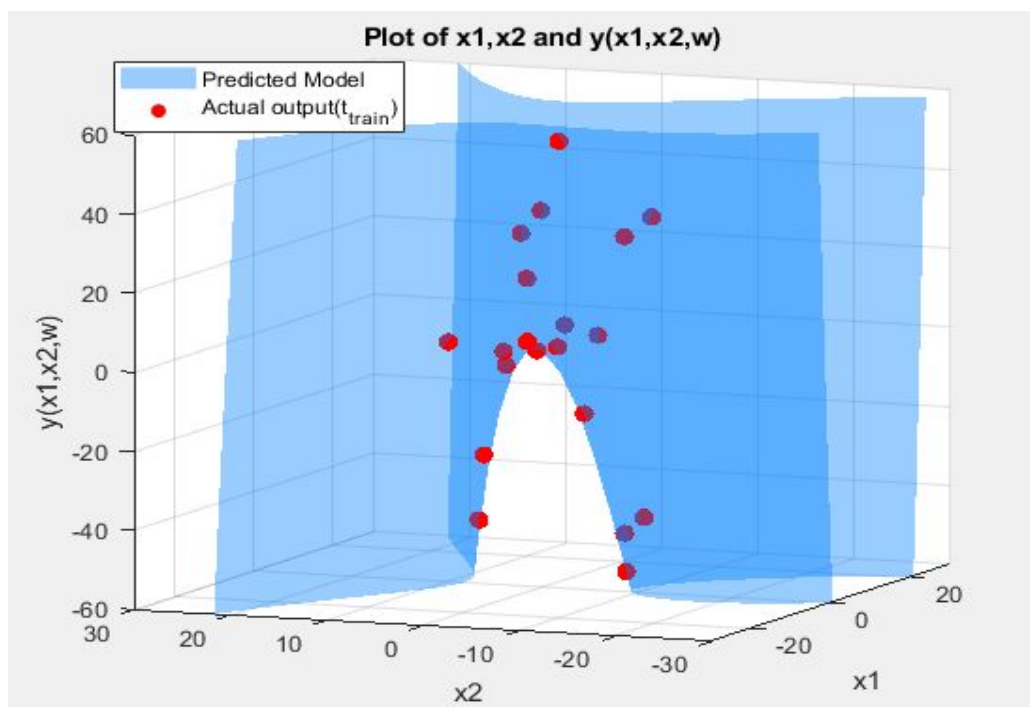
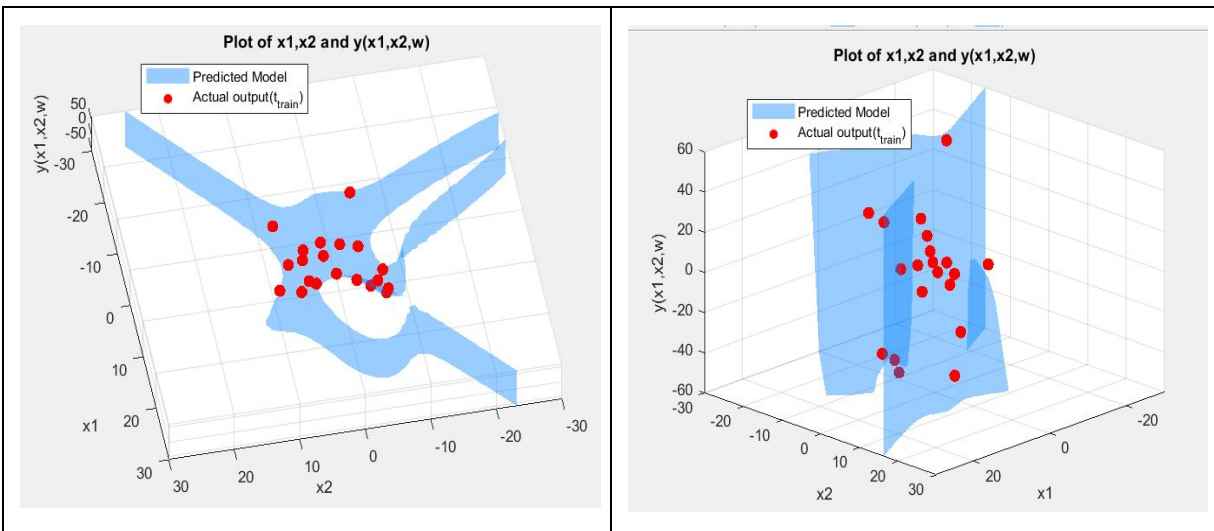
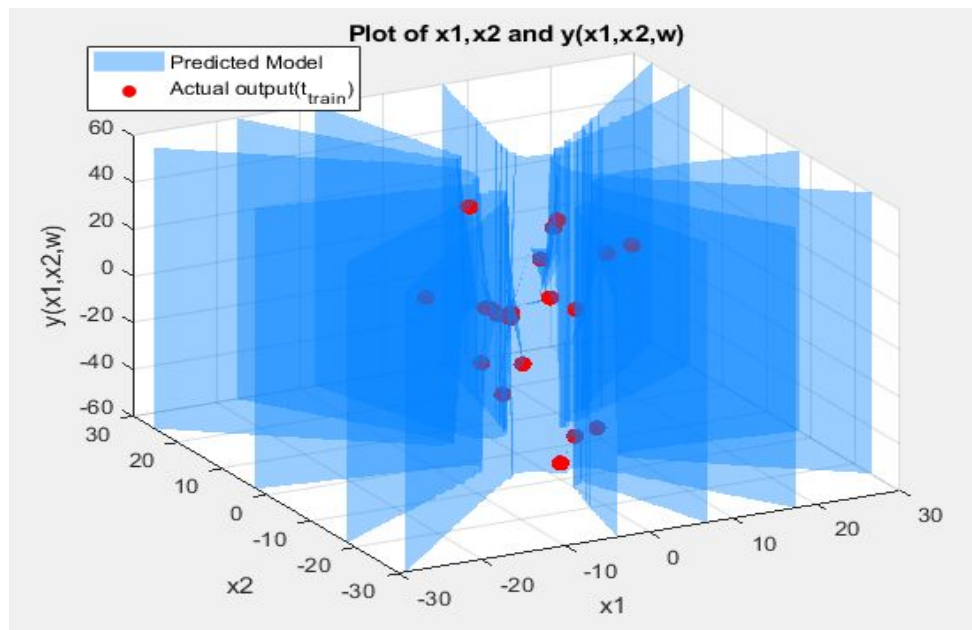


Figure 2.2.6:- Training data plot( $x_1, x_2$  Vs  $y$ ) ( $N=20, M=3, \lambda=0$ )

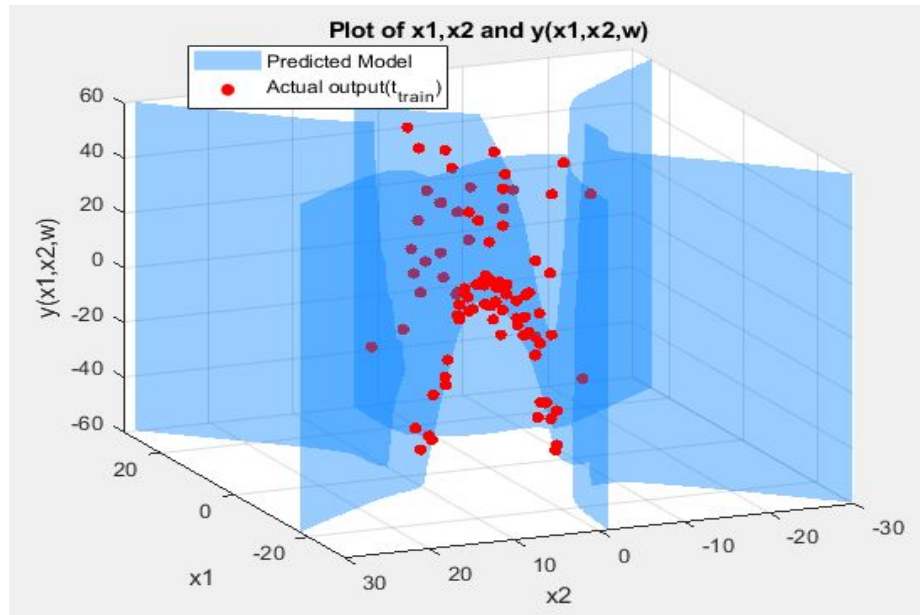




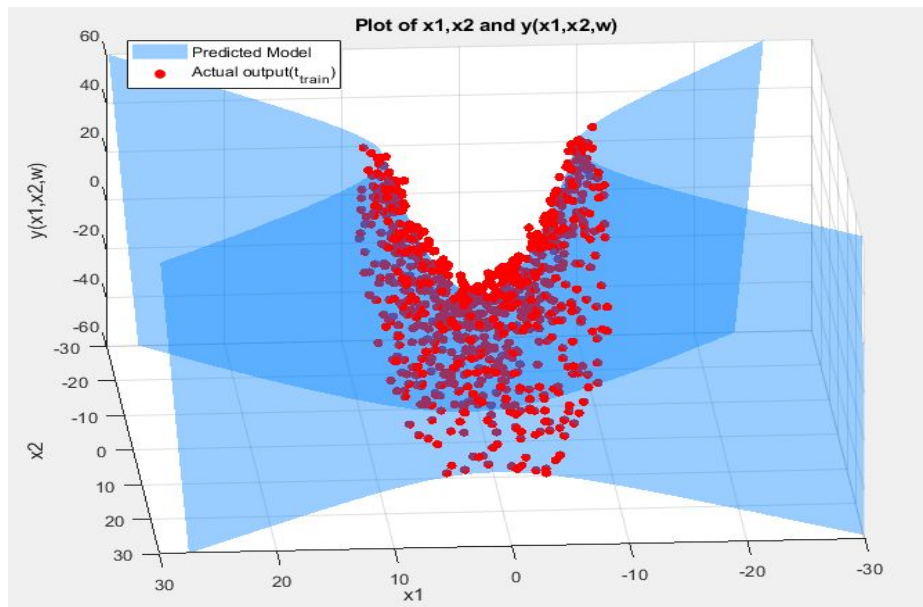
**Fig 2.2.7: Training data plot( $x_1, x_2$  versus  $y$ ) ( $N=20, M=5, \text{Lambda}=0$ )..**



**Fig 2.2.8 : Training data plot( $x_1, x_2$  versus  $y$ ) ( $N=20, M=5, \text{lambda}=2$ )**



**Fig 2.2.9:- Training data plot( $(x_1, x_2)$  Vs  $y$ ) ( $N=10$ ,  $M=7$ ,  $\text{Lambda}=0$ )**



**Fig 2.2.10 : Training data plot( $(x_1, x_2)$  Vs  $y$ ) ( $N=1000$   $M=3$   $\text{Lambda}=0$ )**

## 2.2) Results

**Fig 2.2.11:  $E_{rms}$  for training data (N = 20)**

	Train20		
	Training Data		
	$\lambda=0$	$\lambda=0.005$	$\lambda=1$
M=0	29.08417	29.08417	29.08427
M=1	28.83347	28.83347	28.83359
M=2	3.638586	3.638586	3.638639
M=3	2.663412	2.663412	2.664335
M=4	2.172339	2.17234	2.197952

**Fig 2.2.12  $E_{rms}$  for test data (N = 20)**

	Test Data		
	$\lambda=0$	$\lambda=0.005$	$\lambda=1$
M=0	38.49926	38.4992	38.48801
M=1	39.49384	39.49365	39.45716
M=2	6.508403	6.508404	6.508673
M=3	10.6091	10.60848	10.49826
M=4	26.74986	26.74927	26.43031

**Fig 2.2.13:  $E_{rms}$  for training data (N = 100)**

	Train100		
	Training Data		
	$\lambda=0$	$\lambda=0.005$	$\lambda=1$
M=0	29.21532	29.21532	29.2142
M=1	28.9874	28.9874	28.98635
M=2	4.633195	4.633183	4.630907
M=3	4.314521	4.314513	4.312926
M=4	3.977927	3.977924	3.97737

**Fig 2.2.14:  $E_{rms}$  for test data (N = 100)**

	Test Data		
	$\lambda=0$	$\lambda=0.005$	$\lambda=1$
M=0	38.17804	38.17805	38.179
M=1	38.10501	38.10501	38.10736
M=2	6.449405	6.449388	6.449388
M=3	6.031032	6.03103	6.0307
M=4	6.227477	6.227463	6.224788

**Fig 2.2.15:  $E_{rms}$  for test data (N = 1000)**

	Train1000		
	Training Data		
	$\lambda=0$	$\lambda=0.005$	$\lambda=1$
M=0	32.22161919	32.22161876	32.22153257
M=1	32.18026646	32.18026618	32.18021177
M=2	4.447886398	4.447885868	4.447781001
M=3	4.43894663	4.438945688	4.438758736
M=4	4.428253354	4.428252674	4.428118578
M=6	4.507340019	4.50733831	4.506999803

**Fig 2.2.16:  $E_{rms}$  for test data (N = 1000)**

	Test Data		
	$\lambda=0$	$\lambda=0.005$	$\lambda=1$
M=0	34.08098966	34.08098882	34.08082045
M=1	33.96957687	33.96957557	33.969317
M=2	5.525187648	5.525188537	5.525364921
M=3	5.439468838	5.43946947	5.439594874
M=4	5.520124956	5.520125578	5.520248448
M=6	5.59442296	5.594422212	5.594273849

## 2.2) Inferences

1. In all the experiments that were conducted, error on training data is much less than error on test data. For example in Fig 2.2.13 for  $M=3$ ,  $E_{rms}(\text{training}) = 4.314521$  while in Fig 2.2.14 for  $M = 3$ ,  $E_{rms} = 6.031032$ . The reason being that model was trained on training data so its parameters were adjusted accordingly and hence it approximates training data much better than test data.
2. The error on training data is not zero but some significant value despite model being trained on it. For example in Fig 2.2.11, for  $M = 2$ ,  $E_{rms}$  on training is 3.638586 because model with that particular  $M$  is not able to perfectly fit the training data but just approximates it to the best of its capacity which is also limited by its degree of freedom.
3. As the size of training data is increasing the model is fitting the data more accurately and hence the error is decreasing. For instance In Fig 2.2.12 with  $M = 3$ ,  $N = 20$ ,  $E_{rms} = 10.6091$ , In Fig 2.2.14 with  $M = 3$ ,  $N = 100$ ,  $E_{rms} = 6.031032$ , In Fig 2.2.13 with  $M = 3$ ,  $N = 1000$ ,  $E_{rms} = 5.439468838$ .
4. The optimal value of  $M$  is the one in which model gives the minimum root mean square error thus it best fits the data. Here from In Fig 2.2.16 with  $N = 1000$   $M = 3$ ,  $\lambda = 1$ ,  $E_{rms}$  is minimum ( $E_{rms} = 5.439594874$ ) hence the optimal value of  $M = 3$ .
5. On further increasing the  $M$  beyond the optimal value, the error on training data decreases For example In Fig 2.2.14 with  $M = 4$ ,  $E_{rms}(\text{training}) = 3.97792$  which is smaller than Minimum value of  $E_{rms} = 4.314521$  for the same figure.  
It is because of the increase in degree of freedom of the model( $M$ ) results in model fitting the training more accurately but as a consequence it fits the “noise” as well which results in poor prediction on test. This causes Error on test data to increase. For example In Fig 2.2.14(test) with  $M = 4$ ,  $E_{rms} = 6.227477$  which is higher than the  $E_{rms}$  in the same Figure for  $M = 3$  ( $E_{rms} = 6.031032$ ). This implies low bias and high variance and is termed as overfitting.

6. On decreasing the  $M$  below its optimal value, the error on training data increases. For example, In Fig 2.2.11 with  $M = 1$ ,  $E_{rms} = 28.9874$  which is greater than Minimum value of  $E_{rms} = 4.314521$ .  
It is because of the decrease in degree of freedom of the model( $M$ ) which decreases the model flexibility. The model becomes too rigid to fit the data leading to high bias but low variance. This is termed as underfitting.
7. In Fig 2.2.12 for  $N = 20$ , quadratic model with  $M = 2$  is the best fit and gives the minimum error ( $E_{rms} = 6.508403$ ).
8. In Fig 2.2.14 for  $N = 100$ , model with  $M = 3$  is the best fit and gives the minimum error ( $E_{rms} = 6.031032$ ).
9. In Fig 2.2.16 for  $N = 1000$ , model with  $M = 3$  is the best fit and gives the minimum error ( $E_{rms} = 5.43946883$ ).
10. On increasing the value of  $\lambda$  for a particular Model with some model complexity,  $M'$ , the value of  $E_{rms}$  is decreasing which is because the regularization increases the smoothness of the curve by making sure that the value of parameters do not get very high and prevents overfitting to some extent. For example, In Fig 2.2.13,  $M = 3$ , for  $\lambda = 0$ ,  $E_{rms} = 4.313521$ , for  $\lambda = 0.05$ ,  $E_{rms} = 4.314513$ , for  $\lambda = 1$ ,  $E_{rms} = 4.312926$ .

### 3) Linear model for regression using Gaussian basis function (Dataset-3 and Dataset-4)

#### 2.3) Method

In this section, the following notations are used

$\bar{x}$  - Vector x, the training data

$\bar{w}$  - Model parameters, coefficient

$\Phi$  - Design matrix

N - Dataset Size

$X^T$  - Transpose of the matrix X or vector x.

M - Number of clusters

d - dimension of the data

$\bar{c}$  = All centroids as vector

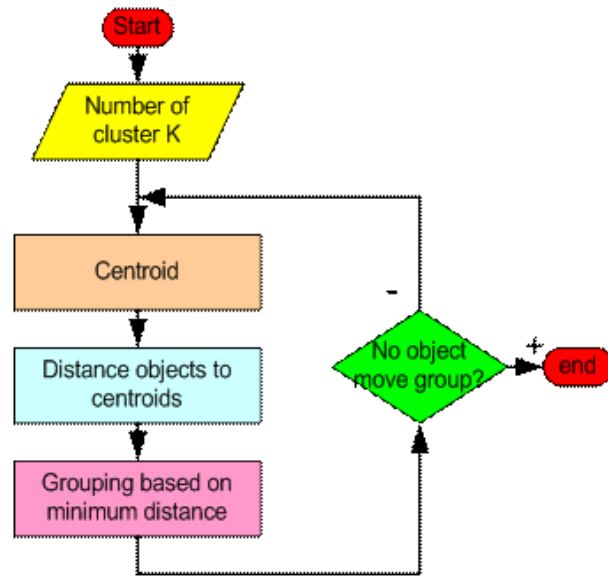
$\bar{p}$  = predicted data

t = System output

y= Model output

- a. Calculate the k-means of the dataset
  - i. Randomly select M points from the data and declare them as the centroids. Each cluster will have one centroid.
  - ii. Calculate distance between each point and the selected M centroid's. This would be a N X M matrix. Euclidean distance is calculated.
  - iii. Assign clusters to each data point depending on the minimum distance from the centroid. Basically grouping the data near to the centroid.
  - iv. Build new centroids by taking the mean of the points selected in step (iii) and repeat the above step till the cluster points remain the same.

Below is the flowchart of the algorithm



**K-Means algorithm flowchart**

- b. Calculate the Design matrix  $\Phi$  where  $i : [1, N]$  and  $j : [1, M]$

$$\Phi(i, j) = e^{(\bar{x} - \bar{c})^2 / 2\sigma^2}$$

- c. Calculate the pseudo-inverse of the design matrix and add the regularization term to the Design matrix.

$$\bar{w} = ((\Phi^T \Phi) + \lambda I) \Phi^T \bar{t}$$

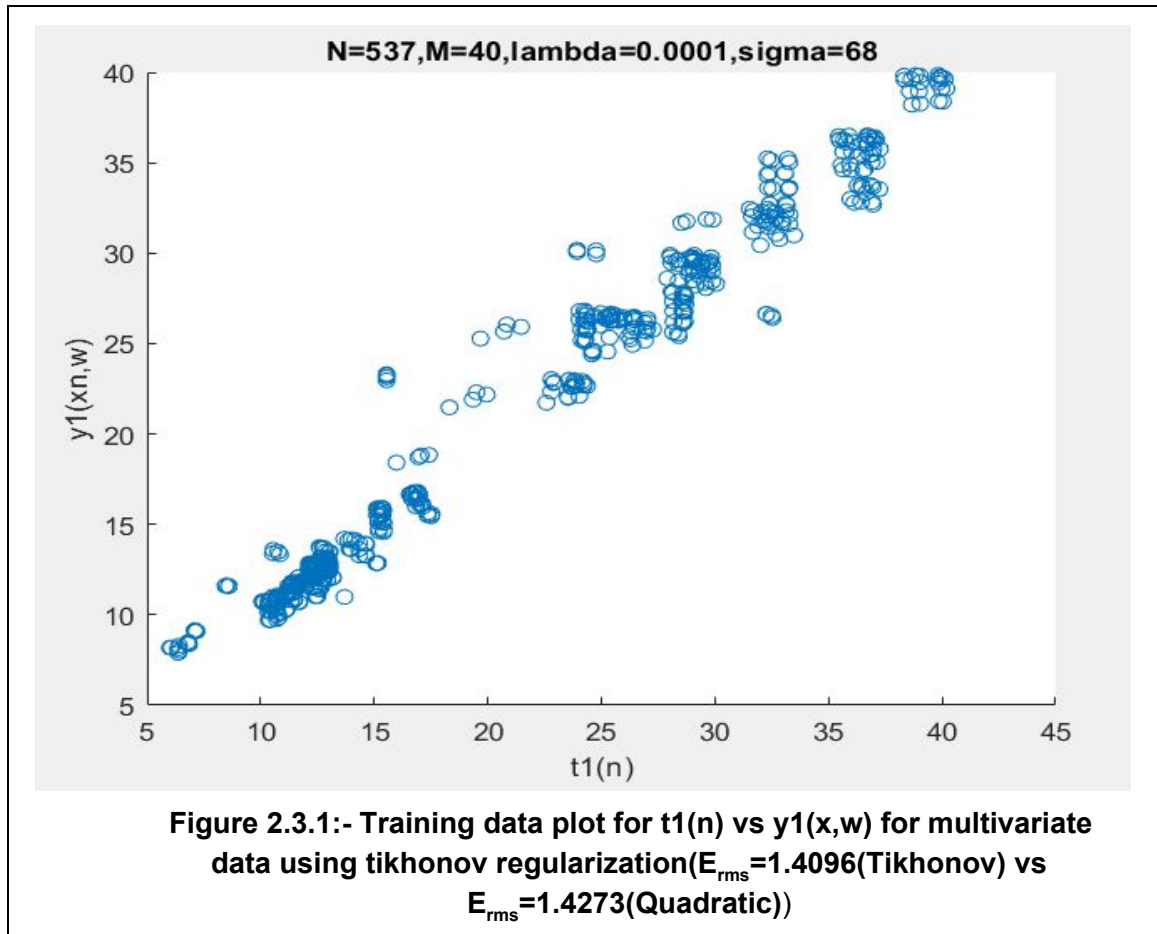
- d. Calculate the predicted data

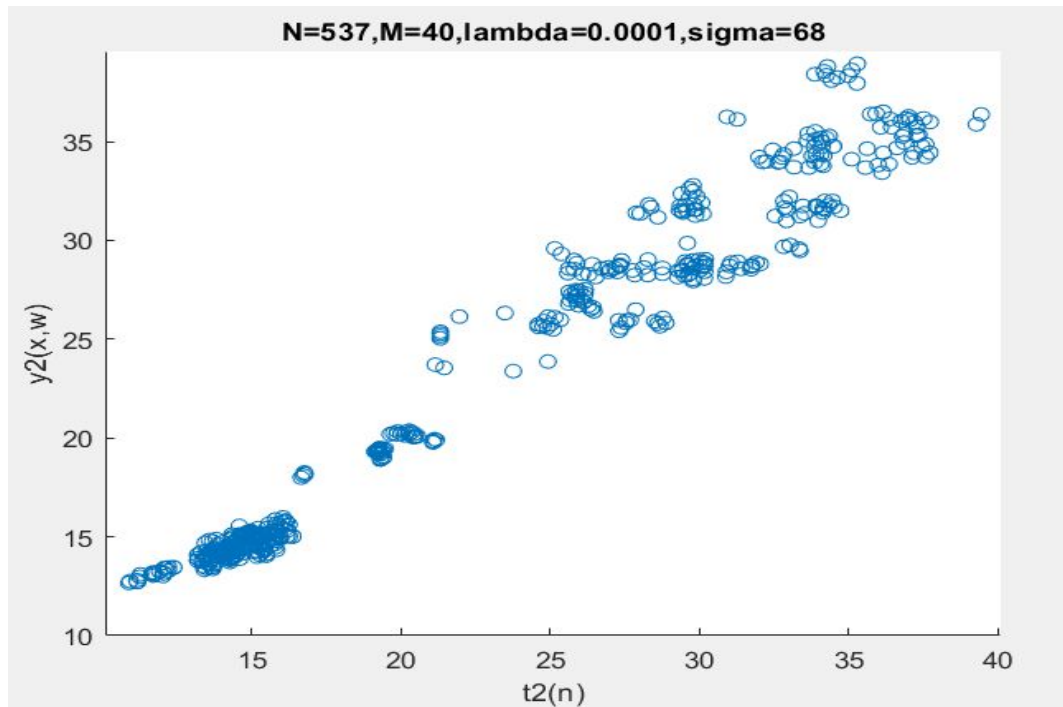
$$\bar{p} = \bar{w}^T * \exp((\bar{a} - \bar{c})^2 / 2\sigma^2)$$

- e. Plot the test and the predicted data.

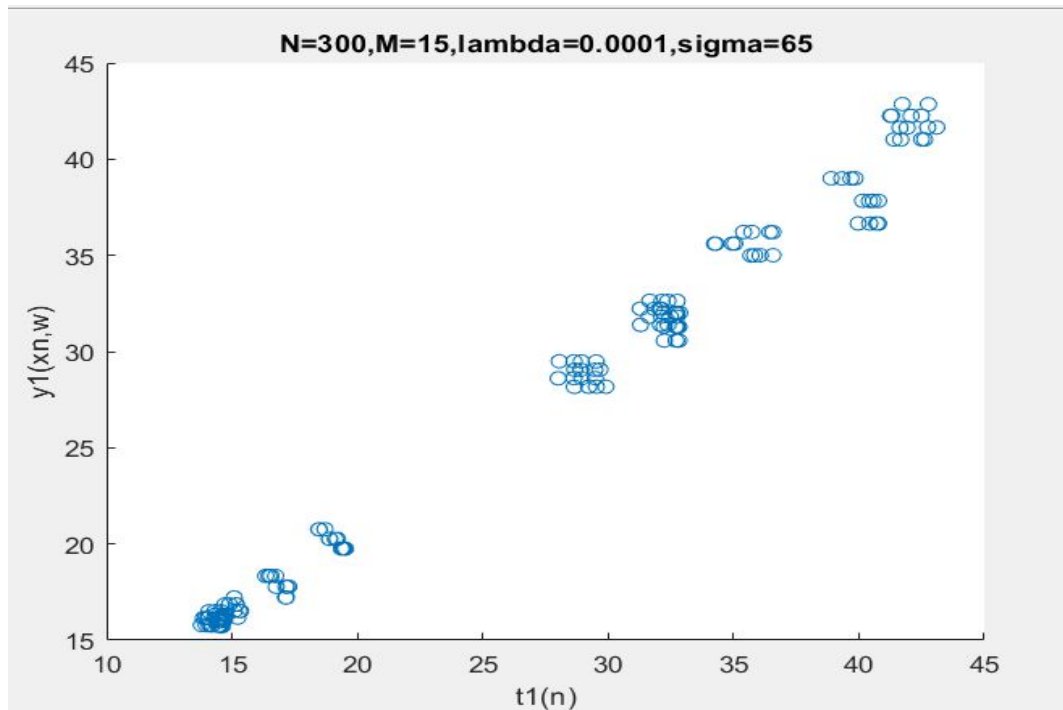


## 2.3) Plots

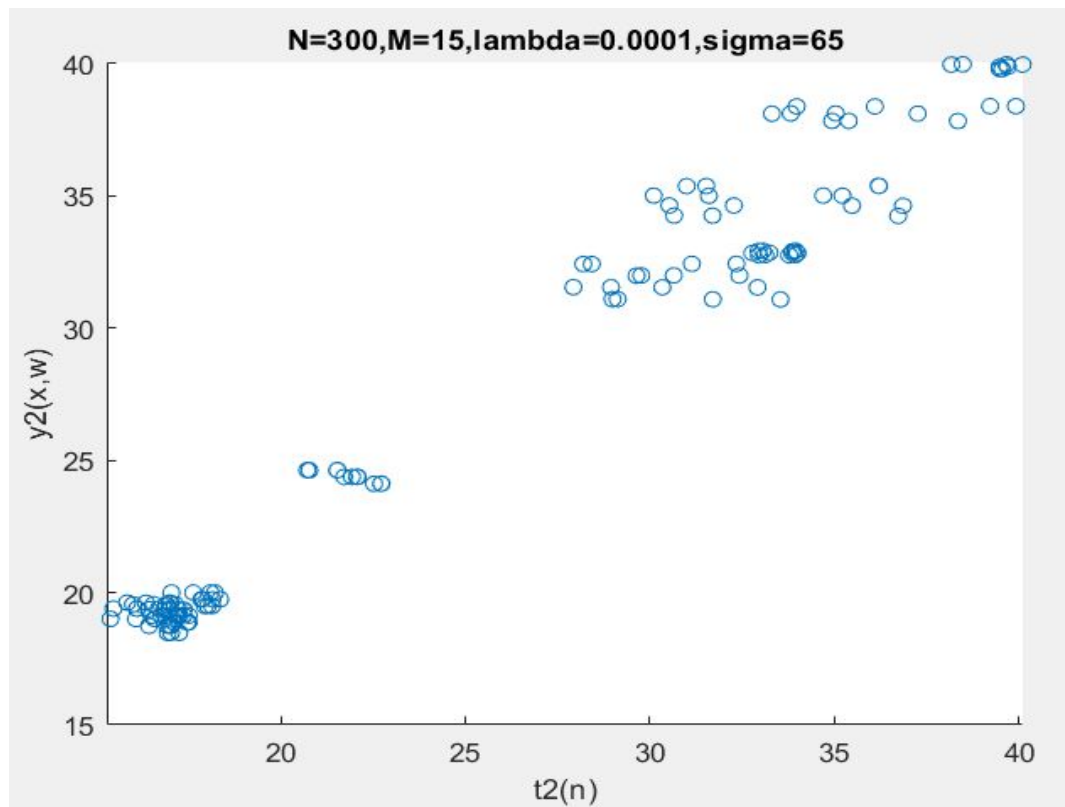




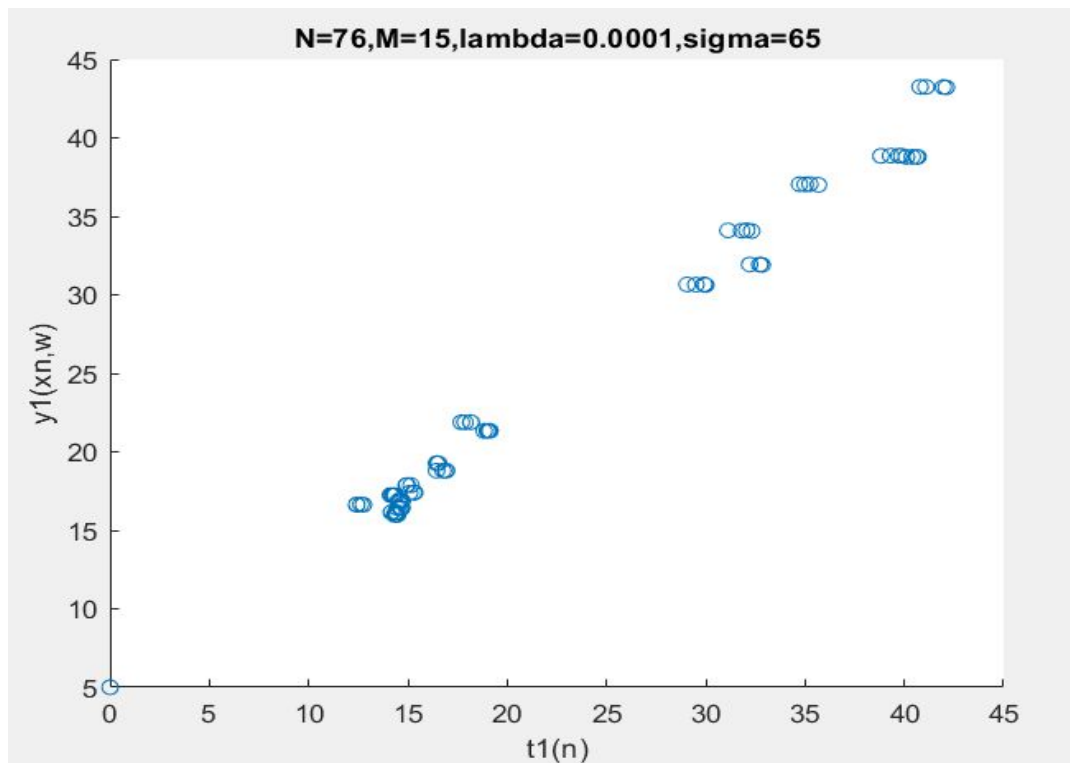
**Figure 2.3.2:- Training Data plot of  $t2(n)$  versus  $y2(x,w)$  for multivariate data using tikhonov regularization. ( $E_{rms}=1.3864$ (Tikhonov) versus  $E_{rms}=1.4201$ )**



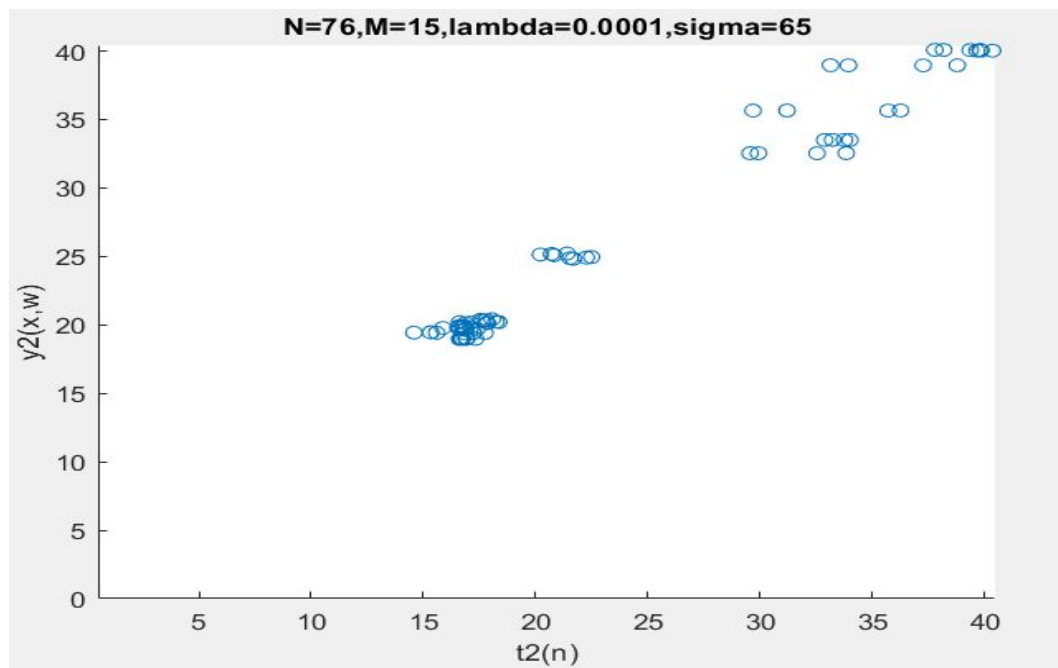
**Fig 2.3.3:- Validation data plot of  $t1(n)$  versus  $y1(x,w)$  for multivariate data using tikhonov regularization.**



**Fig 2.3.4:- Validation data plot of  $t2(n)$  versus  $y2(x,w)$  for multivariate data using tikhonov regularization.**

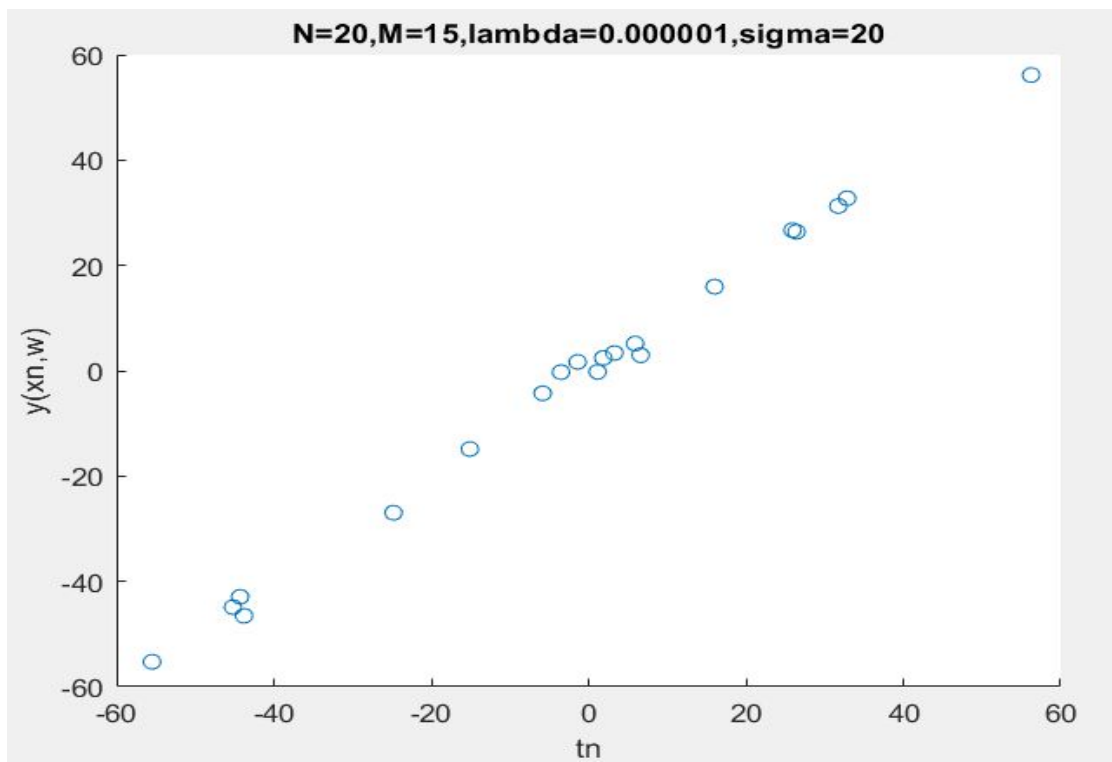


**Figure 2.3.5:- Test data plot for  $t1(n)$  versus  $y1(x,w)$  for multivariate data using tikhonov regularization**

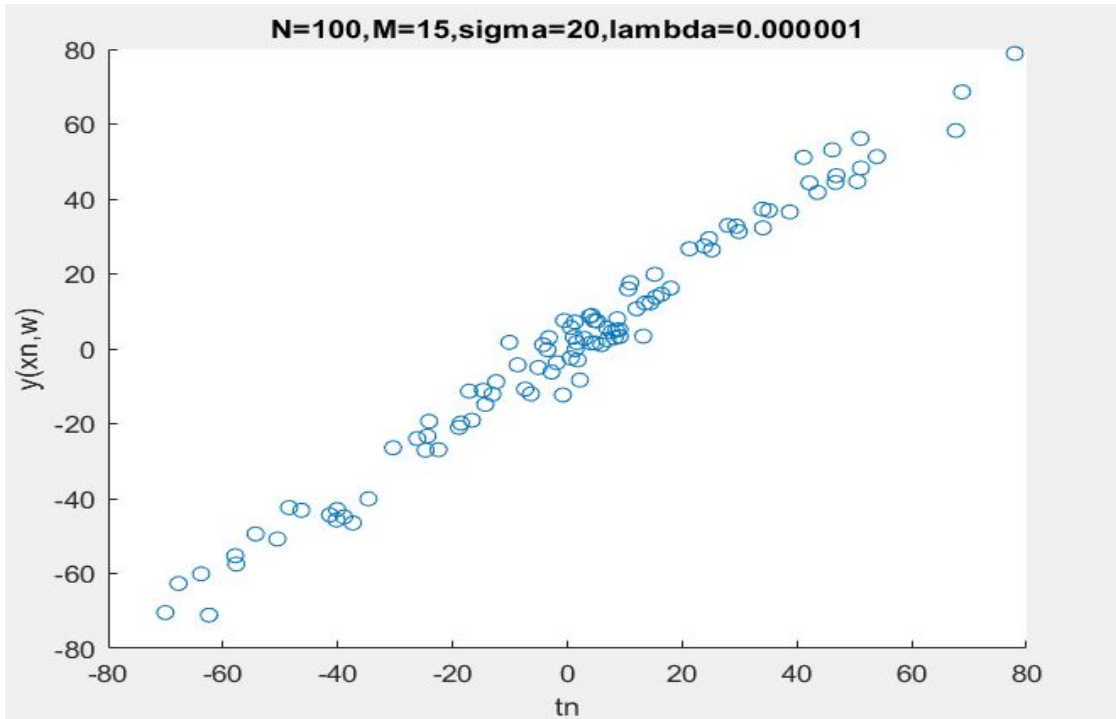


**Figure 2.3.6:- Test data for  $t2(n)$  versus  $y2(x,2)$  for multivariate data using tikhonov regularization.**

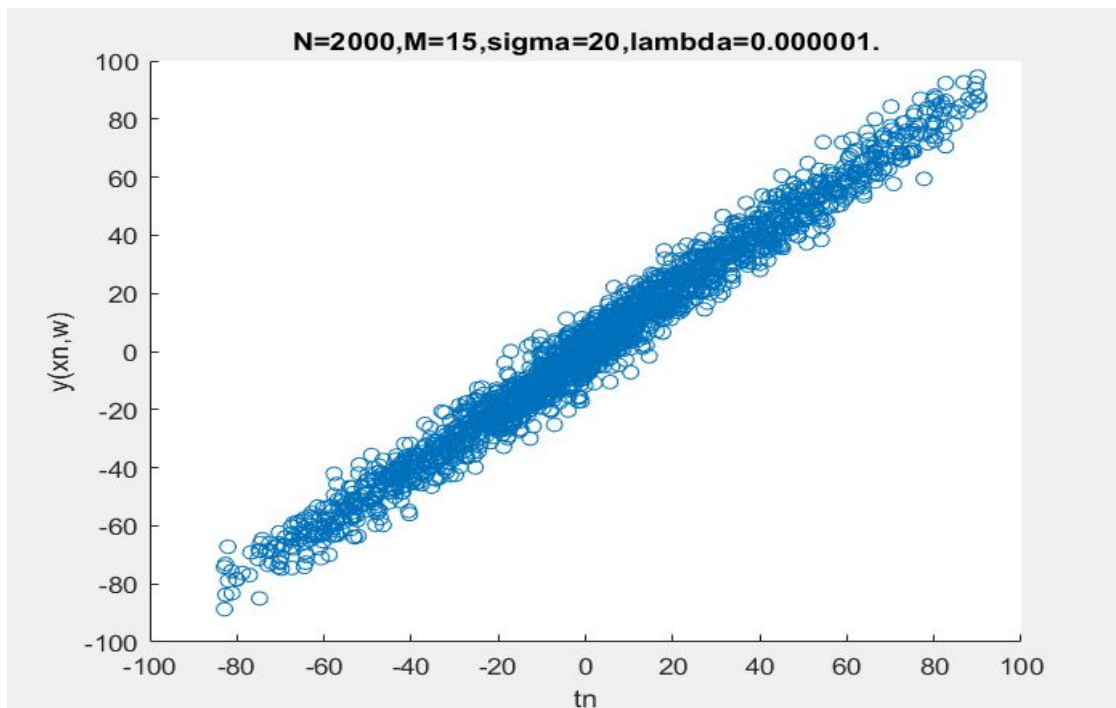
### Bivariate Data plots using gaussian basis functions.



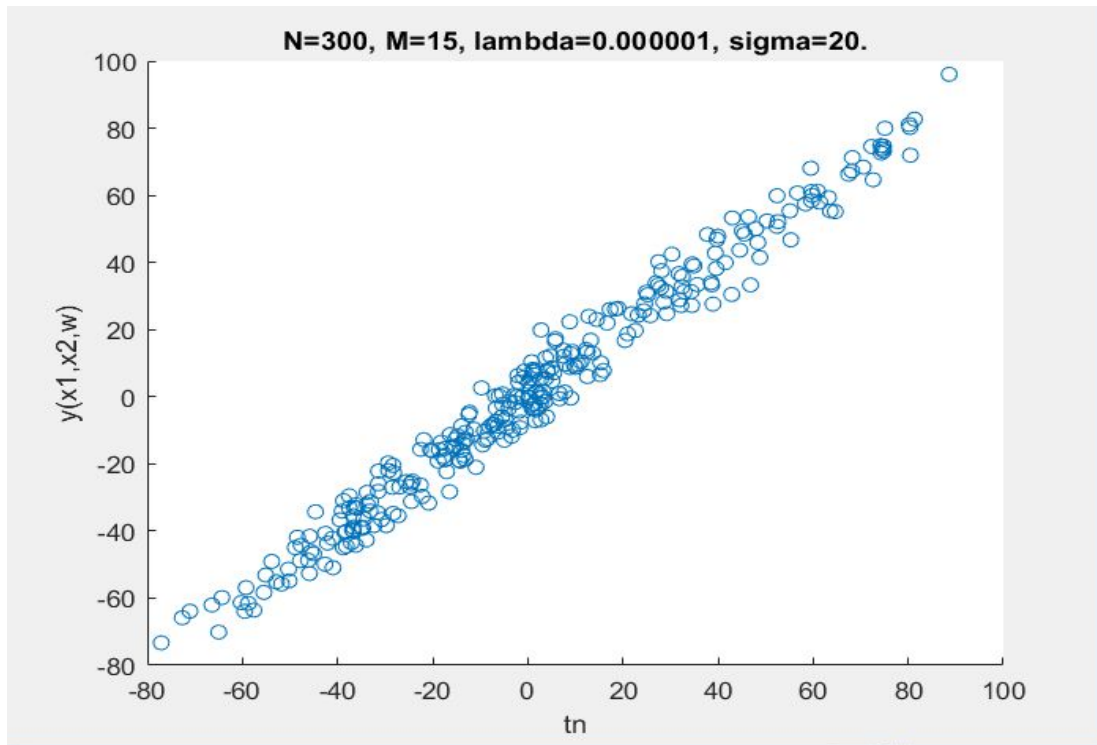
**Fig 2.3.7:-Training Data plot of  $t(n)$  vs  $y(x1,x2,w)$  for bivariate data using tikhonov regularization ( $E_{rms}=1.6394$ (Tikhonov)).**



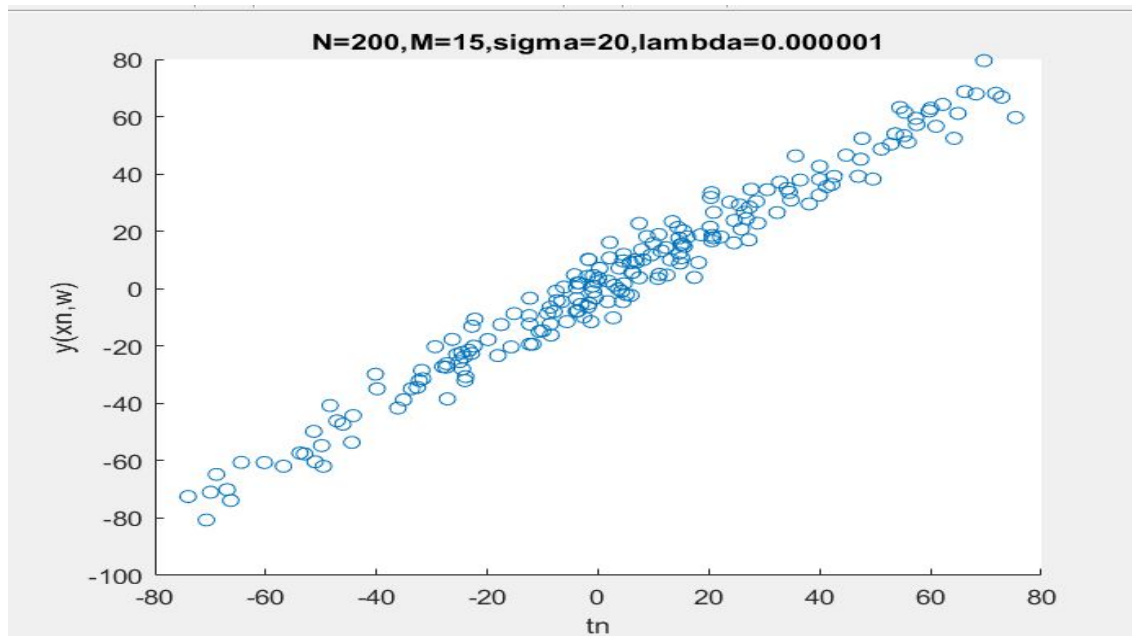
**Fig 2.3.8:-Training Data plot of  $t(n)$  vs  $y(x_1,x_2,w)$  for bivariate data using tikhonov regularization ( $E_{rms}=1.6733$ (Tikhonov)).**



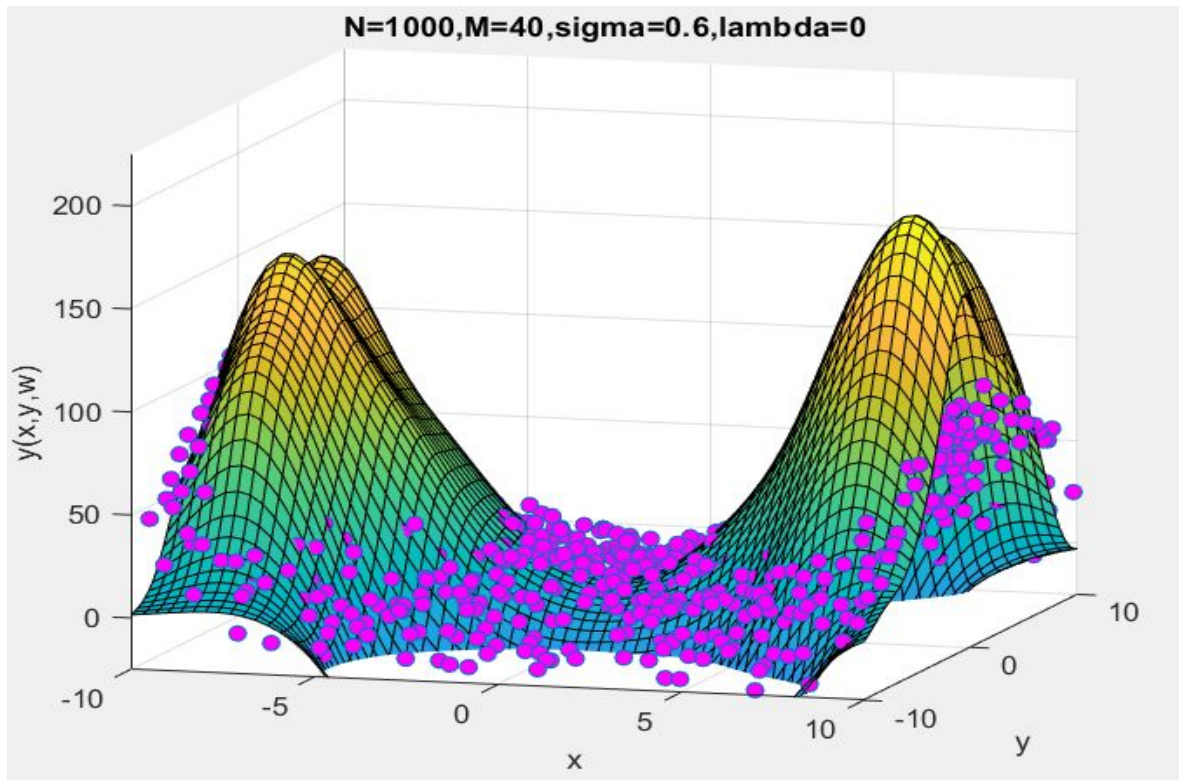
**Figure 2.3.9:- Training Data plot of  $t(n)$  vs  $y(x_1,x_2,w)$  for bivariate data using tikhonov regularization ( $E_{rms}=3.4437$ (Tikhonov) versus  $E_{rms}=3.7526$ (Quadratic)).**



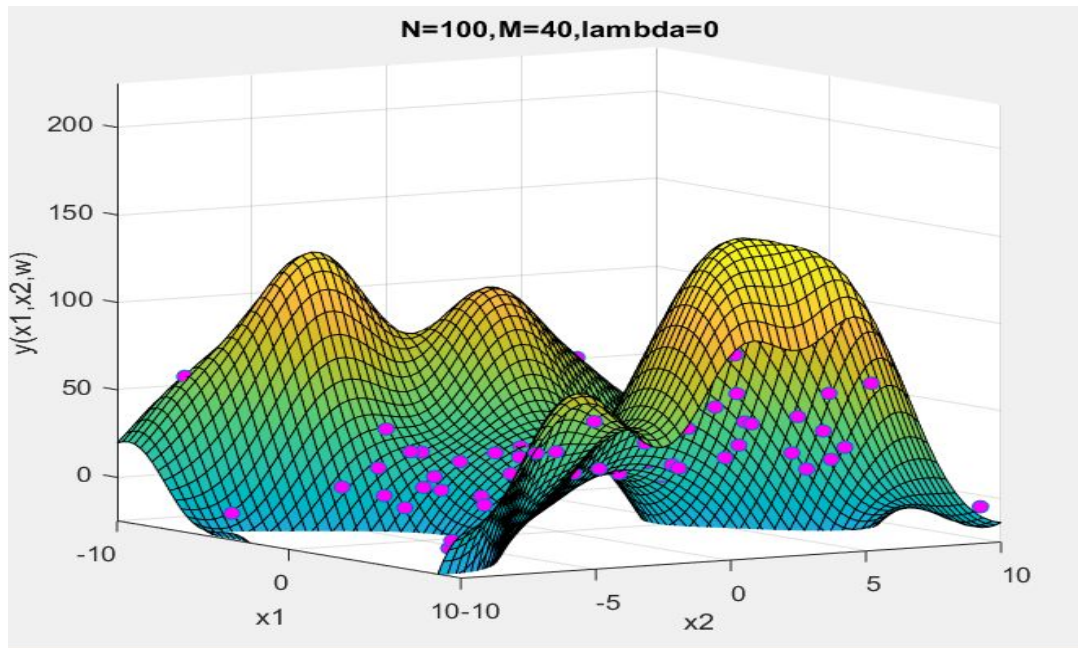
**Figure 2.3.10:- Validation data plot for  $t(n)$  versus  $y(x,w)$  for bivariate data using Tikhonov regularization. ( $E_{rms}=2.1181$ (Tikhonov) vs  $E_{rms}=1.9199$ (Quadratic)).**



**Figure 2.3.11:- Test data plot for  $t(n)$  versus  $y(x,w)$  for bivariate data using tikhonov regularization. ( $E_{rms}=1.8265$ (Tikhonov) versus  $E_{rms}=1.9225$ (Quadratic)).**

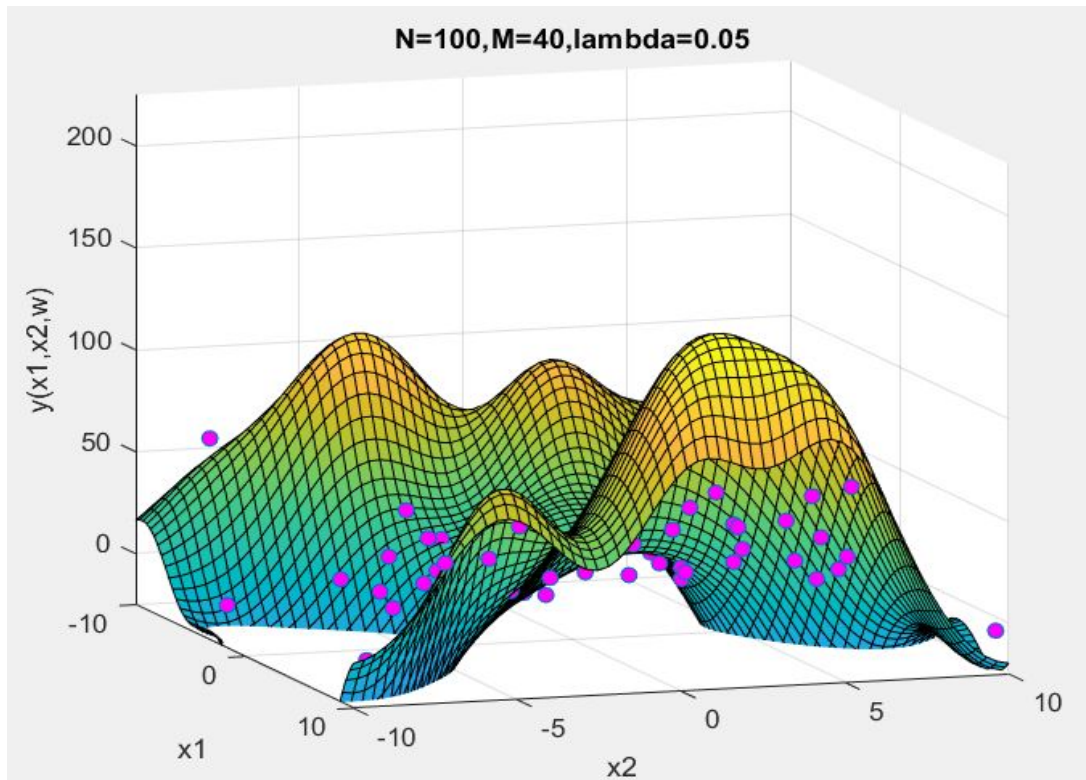


**Fig 2.3.12:- Training data plot of  $(x_1, x_2)$  versus  $y(x_1, x_2, w)$  with optimum values.**

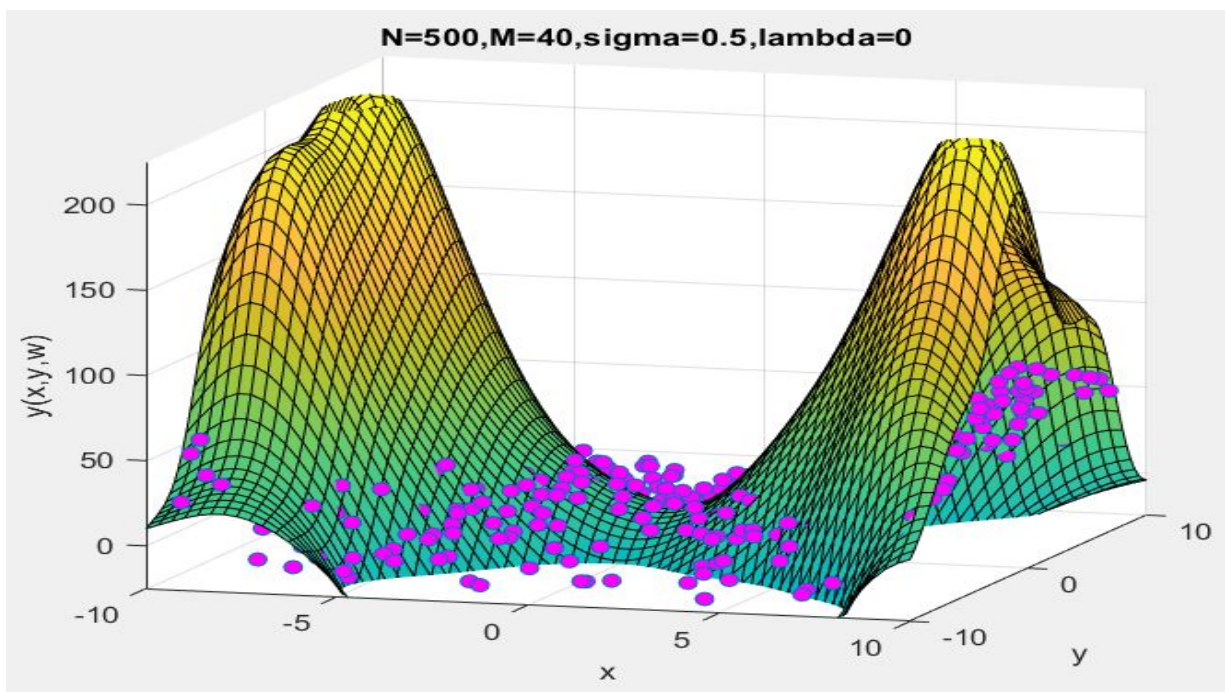


**Fig 2.3.13:- Training data plot of  $(x_1, x_2)$  versus  $y(x_1, x_2, w)$  (Overfitting) with  $\sigma=0.5$**





**Fig 2.3.14:- Training data plot of  $(x_1, x_2)$  versus  $y(x_1, x_2, w)$  (Overfitting reduced with  $\lambda=0.05$  and  $\sigma=0.5$ ).**



**Fig 2.3.15:- Training data plot of  $(x_1, x_2)$  versus  $y(x_1, x_2, w)$  (Overfitting reduced with  $N=300$  and  $\sigma=0.5$ ).**

### 2.2.3) Results

**Fig 2.3.16:-  $E_{rms}$  for Dataset 3 (N = 100) for various values of M and  $\lambda$**

		Dataset 3 ERMS(Gaussian)		
		Training Data		
		y1 (Output)		
	Sigma	Without reg ( $\lambda=0$ )	With reg ( $\lambda=0.000001$ )	
M=6	87.6108	4.9379	5.7908	
M=7	38.8865	4.2249	5.2679	
M=8	30.4519	3.6212	4.4954	
M=9	20.6996	2.5908	3.9867	
M=15	20	1.6733	2.6030	
M=20	11	1.0850	2.1429	
		Test Data		
		y1 (Output)		
	Sigma	Without reg ( $\lambda=0$ )	With reg ( $\lambda=0.000001$ )	
M=6	87.6108	5.0012	5.6643	
M=7	38.8865	4.3555	5.2143	
M=8	30.4519	3.5858	4.9671	
M=9	20.6996	2.4749	3.3564	
M=15	20	1.4565	2.7123	
M=20	11	3.3316	3.9998	

**Fig 2.3.17:-  $E_{rms}$  for Dataset 3 (N = 20) for various values of M and  $\lambda$**

		Dataset 3 ERMS(Gaussian)		
		Training Data		
		y1 (Output)		
	Sigma	Without reg ( $\lambda=0$ )	With reg ( $\lambda=0.1$ )	
M=2	15	28.7287	29.6484	
M=6	11	5.4118	5.6063	
M=9	13	3.3983	3.8277	
M=10	8	2.6570	2.2462	
M=20	0.01	0	2.6479	
		Test Data		
		y1 (Output)		
	Sigma	Without reg ( $\lambda=0$ )	With reg ( $\lambda=0.1$ )	
M=2	15	34.3300	34.2600	
M=6	11	7.4525	7.9924	
M=9	13	3.4766	4.2534	
M=10	8	7.0240	6.8921	
M=20	0.01	15.6789	10.6811	

**Fig 2.3.18:-  $E_{rms}$  for Dataset 4 for various values of M and  $\lambda$**

				Dataset 4 ERMS				
				Training Data				
			y1 (Output)				y2(Output)	
	Sigma	$\lambda=0$	$\lambda=25$	$\lambda=100$		$\lambda=0$	$\lambda=25$	$\lambda=100$
M=1	110	11.4061	11.5243	11.3115		11.1498	11.6008	11.2323
M=5	100	2.5333	4.9811	7.9046		3.1190	4.1543	7.2883
M=8	70	2.3314	4.2158	7.8054		2.1256	4.0976	7.3754
M=15	20	2.2851	6.2900	13.1234		2.0096	6.4412	13.3280
M=40	20	1.4096	5.9008	12.7370		1.3864	6.0646	12.5725
				Test Data				
	Sigma	$\lambda=0$	$\lambda=25$	$\lambda=100$		$\lambda=0$	$\lambda=25$	$\lambda=100$
M=1	110	11.78	11.7758	11.7759		11.4150	11.2559	11.2800
M=5	100	2.6037	4.5757	7.9295		3.2970	4.2870	7.4092
M=8	70	2.5293	4.4186	7.9193		2.3851	4.2265	7.4000
M=15	20	1.9309	4.3777	7.4300		2.2877	4.3123	7.4199
M=40	20	2.9106	8.7762	14.7111		2.5576	8.2051	14.6506

## 2.2.4) Inferences

1. Fig 2.3.1 and Fig 2.3.2 shows the output of  $t_1(n)$  versus  $y(x,w)$  and  $t_2(n)$  versus  $y(x,w)$  for multivariate data.

We can see that the target output ( $t_1(n)$ ) and model output  $y(x,w)$  is almost same for all target outputs. So the model correctly predicts the output.

Similarly, Fig 2.3.3 and Fig 2.3.4 shows the output of  $t_2(n)$  versus  $y(x,w)$  and  $t_2(n)$  versus  $y(x,w)$  for test data.

We can see that the target output ( $t_1(n)$ ) and model output  $y(x,w)$  is almost same for all target outputs. So the model correctly predicts the output.

2. Fig 2.3.7, 2.3.8, 2.3.9, 2.3.10 and 2.3.11 shows the output of  $t(n)$  vs  $y(x,w)$  for bivariate data using gaussian distribution. We can see that the target output  $t(n)$  vs  $y(x,w)$  is almost same for all target outputs. So the model correctly predicts the output.

3. Figure 2.3.12 shows 3-D plot of  $(x,y)$  versus  $y(x,y,w)$  for bivariate data using gaussian basis functions.

We can see that most of the target outputs lie on the gaussian surface.

This implies that the model correctly predicts the target outputs for given input vector  $(x,y)$ .

4. For bivariate data, with No. of clusters=15,  $\lambda=0.000001$  we get an  $E_{rms}=1.8265$  using gaussian basis functions.

Using degree=3,  $\lambda=1$ , we get an  $E_{rms}=5.4396$  using polynomial basis functions.

Hence, polynomial basis functions were more suitable than gaussian basis functions for given bivariate data set.