Homework 1

Question 2 (1.)

Description of Dataset:

X ranges from -5 to 5 in a linear spacing of 1000, and Y is obtained using a function f(x)=sinx, to which we add the random noise later to make the data realistic.

Splitting the Dataset

We then split our dataset with the test_size of 0.2, meaning we use 80% of the data for training the model. We make use of sklearn in doing so i.e. train_test_split.

Weights Obtained using closed-form solution:

```
The weights as obtained from one-step solution:
 [[1.69890033e-02]
 [3.97267712e-01]
 [-1.72700240e-04]
 [-3.02314285e-02]
 [-9.82909863e-05]]
```

Fig 1: Test Data (Blue) Vs. Predicted data on Test Set (Green)

Fig 2: Training Data (Blue) Vs. Predicted Data on Train Set (Red)

```
Text(0, 0.5, "outputs 'y'")
```

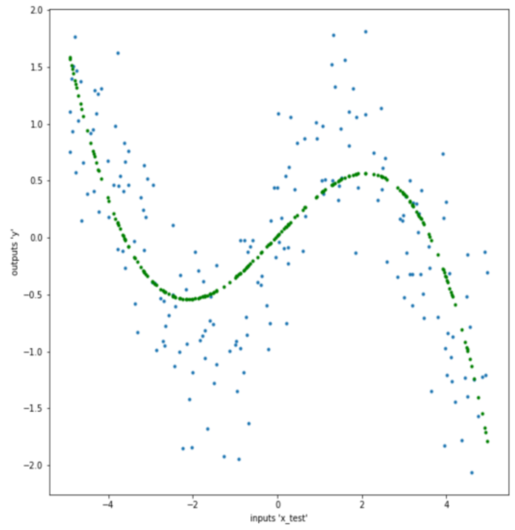


Fig 1.

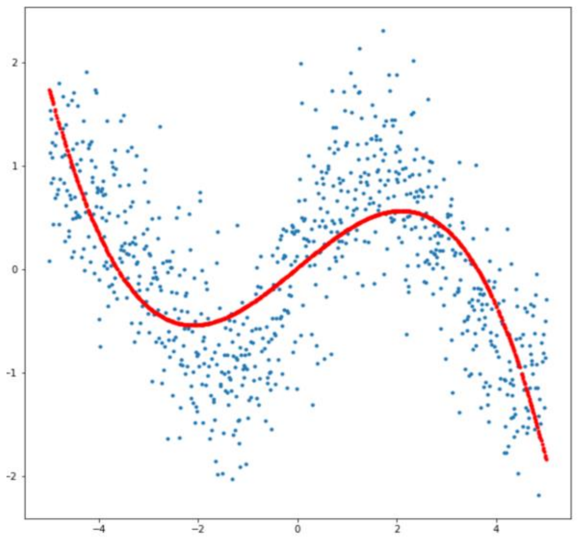


Fig 2.

Root Mean Squares:

```
The root mean square on the test set is 0.6130722933403482
The root mean square on the train set is 0.6295192071311908
```

Question 2(2.)

Implementation of Batch Gradient Descent:

First, we normalize the dataset to make sure our data falls within a specified range. We use the normal normalization technique i.e.

$$(X - \mu) / \text{sigma}$$

Tuning the algorithm with Learning rate and Number of Iterations:

Though we had an option to run the iteration until the convergence, we chose to run it for a given value.

Using **lr = 0.001** and **number of iterations as 10000**, we obtain:

Weights:

```
([[0.00447455],
  [0.57042038],
  [-0.00477323],
  [-0.84987292],
  [-0.01882188]])
```

Though the weights do not perfectly align with those obtained using closed-form solution, we believe some tuning in the parameters will help in doing so.

Fig 3: Cost Function vs. Number of Iterations

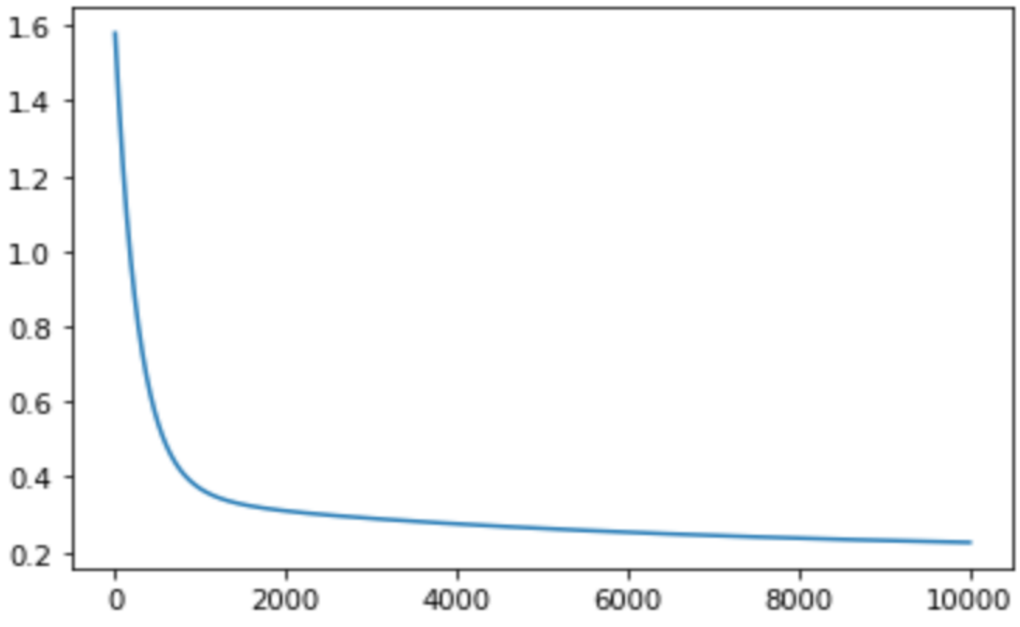
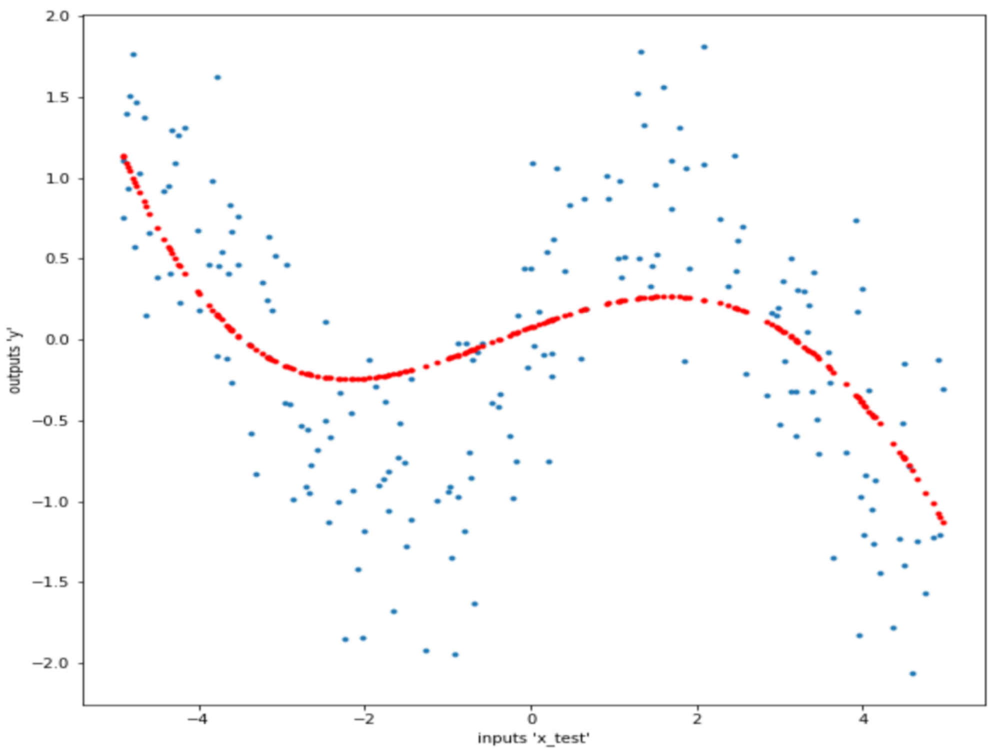


Fig 3.

Fig 4: Test Data (Blue) Vs. Predicted data on test set (Red)



Root Mean Square on the Test Set:

```
The root mean square on the test set is 0.6683942601682994
```

Question 2(3.)

Overfitting Issue:

We played around with different numbers for the sample i.e. N, and the number of polynomials i.e. polynomial (in code).

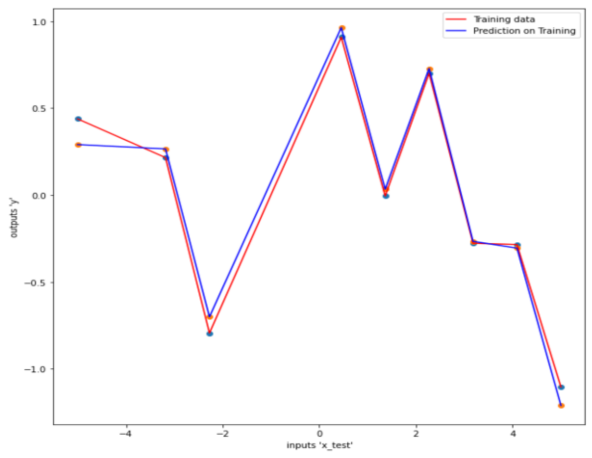
We use N= 12, and polynomial = 10 to introduce the concept of overfitting.

Weights when overfitting happens:

```
[[-2.24318183e-02]
 [-1.89160156e+01]
 [1.33339844e+01]
 [1.20714844e+02]
 [-1.82187500e+02]
 [-1.91175781e+02]
 [4.76531250e+02]
 [-7.89062500e+01]
 [-3.09750000e+02]
 [1.67031250e+02]
```

Fig 5: Training Data (Red) Vs. Prediction on Train Set (Blue)

Fig 6: Test Data (Red) Vs. Prediction on Test Set (Blue)



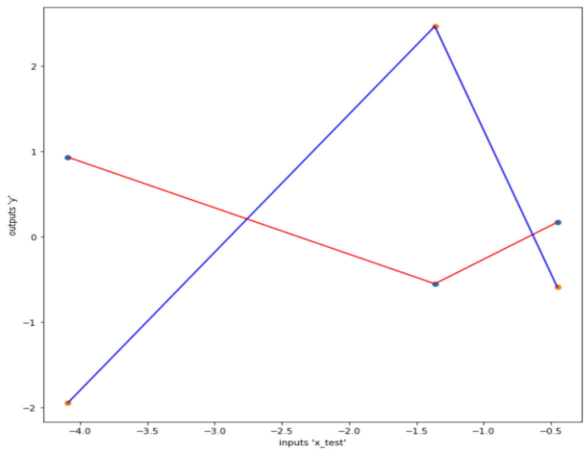
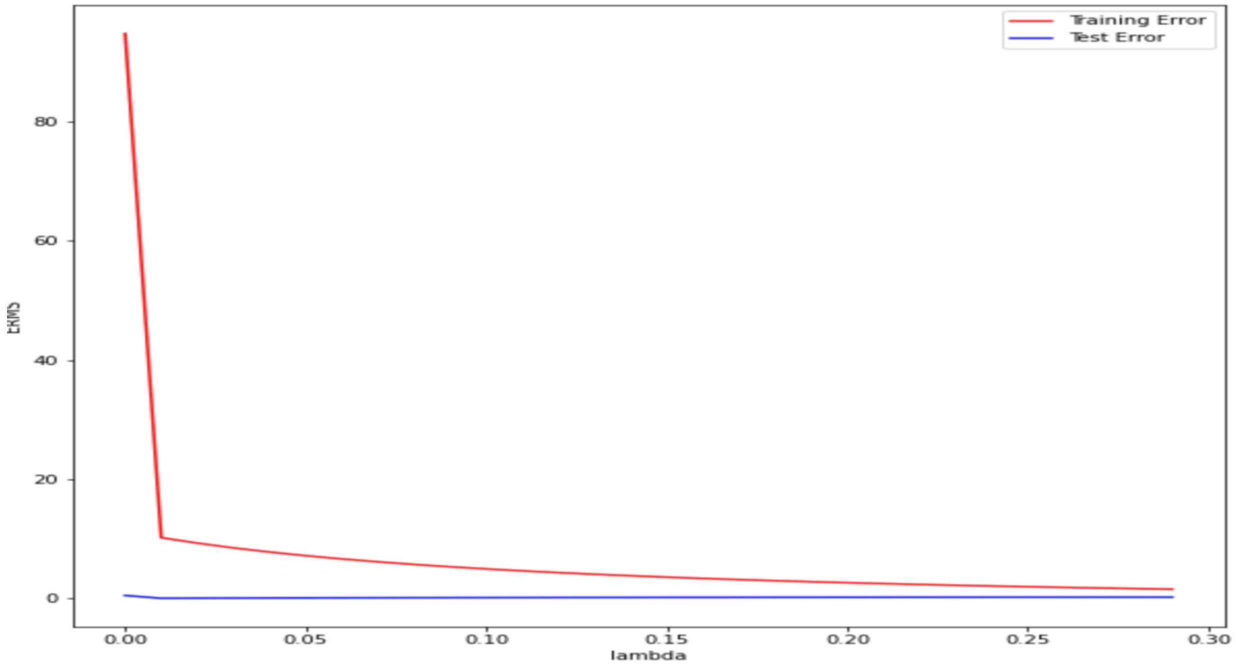


Fig 5. Fig 6.

We use regularization technique to fix the overfitting issue. We use lambda from the range of 0 to 0.3 with a step of 0.01

Fig 7: ERMS vs. Lambda



With the weights thus obtained after introducing regularization, we obtain the following graph:

Fig 8: Training Data (Red) Vs. Prediction on Train Set (Blue)

Fig 9: Test Data (Red) Vs. Prediction on Test Set (Blue)

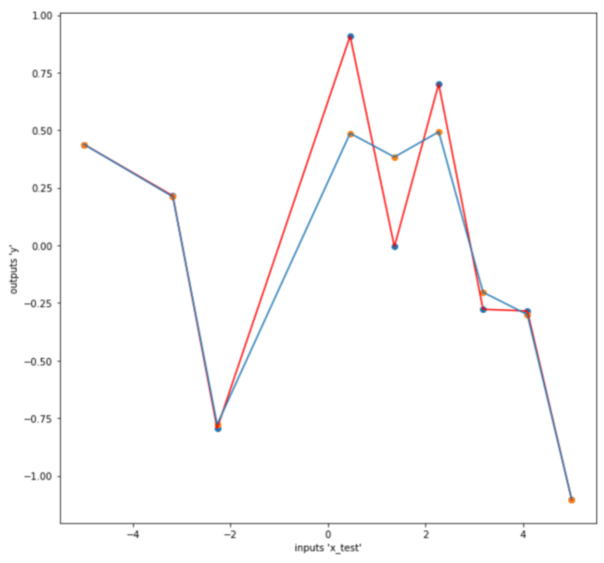


Fig 8.

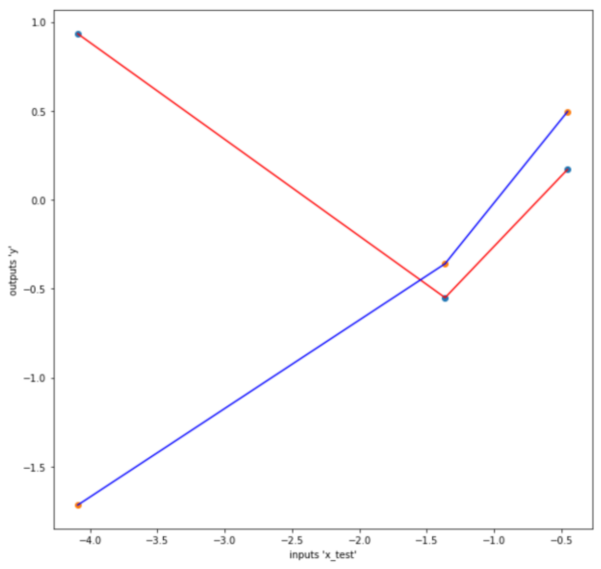


Fig 9.

Root Mean Squares:

```
The root mean square error on the test set is 1.5452452073382066
The root mean square error on the train set is 0.20505288088758705
```

Introducing Lambda helped us penalizing the large weights, and therefore in achieving the better results by reducing the ERMS.

Q.1

$$\tilde{E}(\omega) = \frac{1}{2} \sum_{i=1}^{N} \{y(x_n, \omega) - t_n\}^2 + \frac{\lambda}{2} \|\omega\|^2$$

We can write,

$$\tilde{E}(\omega) = \frac{1}{2} (t - \phi\omega)^T (t - \phi\omega) + \frac{\lambda}{2} \omega^T\omega$$

$$= (t - \phi\omega)^T (t - \phi\omega) + \omega^T\omega$$

$$= t^Tt - t^T\phi\omega - (\phi\omega)^T t + (\phi\omega)^T\phi\omega + \lambda\omega^T\omega$$

$$= t^Tt - 2\phi^T\omega^T t + \phi^T\omega^T\phi\omega + \lambda\omega^T\omega$$

Differentiationg w.r.t ω^T, we get;

$$-2\omega^T t + 2\phi^T\phi\omega + 2\lambda\omega$$

$$\therefore \quad 0 = -\omega^T t + \phi^T\phi\omega + \lambda\omega$$

$$\omega^T t = \phi^T\phi\omega + \lambda\omega$$

$$\omega^T t = \omega(\phi^T\phi + \lambda I)$$

$$\therefore \quad \omega = (\phi^T\phi + \lambda I)^{-1} \omega^T t$$

Hence,

closed-form sol^n for ω :

$$\boxed{(\phi^T\phi + \lambda I)^{-1} \omega^T t}$$