Assignment 07: On Linear And Non Linear Regression

PH1050 Computational Physics

Pranav S Ramanujam
EP23B038

1^{st} year EP
Department of Physics, IIT Madras

Problem Statement

Part A: To Generate a linear data and add noise to get the "experimental data". Then fit the data to obtain the original linear function. First do it manually and then do it using LinearModelFit[] and FindFit[].

Part B: To Generate a multi-parameter (at least 3) data with Gaussian Noise. Then find the parameters of the original function. First do it using FindFit[] and plot the function obtained. Then it must be done by converting the data into matrix form and using LinearSolve[] to get the fit parameters. Plot the generated and the fitted data.

Part C: To generate a non Linear data of our own choice and add White noise to it to get the experimental data. Then obtain the original parameters using NonlinearModelFit[].

Aim

Part A: To generate linear data and then add disturbance to it to generate the experimental data. Then to obtain the original parameters by the 2 methods given in the problem statement. Then to plot the original data, Noisy data and the data obtained after the fitting is done.

PartB: To take a non linear function consisting of atleast 3 parameters and then generate experimental data by adding Gaussian noise to it. Then to obtain the original parameters of the function by using FindFit[] and by representing the data in the Matrix form and then using LinearSolve[]. Then to plot the original data, Noisy data and the data obtained after the fitting is done.

Part C: To generate non linear data and then add white noise to it to generate the experimental data. Then to obtain the original parameters of the function by using the function NonLinearModelFit[]. Then to plot the original data, Noisy data and the data obtained after the fitting is done.

Introduction

This code involves the usage of many different kinds of functions such as RandomReal[{0, 50}, n] and RandomVariate[NormalDistribution[0, 6], n] to generate white and Gaussian noise respectively. It also makes use of Plotting functions to plot graphs and the functions LinearModelFit[], FindFit[], LinearSolve[] and NonlinearModelFit[] to obtain the initial paramter values when fed with the noisy data.

Code Organization

Part A:
1)I defined a linear function ‘y’.
2)Generated a list of random values of ‘x’ which serve as the input values for ‘y’.
3)Made a list of the corresponding output values of ‘y’.
4)Added Gaussian noise to each of the elements of the list containing the output values pf ‘y’.
5)Plotted this NoisyData.
6)Then using the formulae given in class to calculate slope and intercept of the original function , calculated the two of them manually.
7)Calculated slope and intercept using LinearModelFit[] and FindFit[] also.
8)Plotted the Noisy data along with the data obtained from the fitted function.

Part B:
1)Defined a function ‘f’ which is non linear and comprised of 3 parameters.
2) Generated a list of Random values for ‘x’ which serve as inputs for the function and made a list of the corresponding output values. Then added Gaussian noise to the set of output values.
3)Then using FindFit[], found the values of the parameters of the original function by using the Noisy Data.
4) Then found the parameters (using only the noisy data) by representing the data in the form of matrices and using LinearSolve[].
5)Plotted the Original data, Noisy data and the data obtained from the fitted function.

Part C:
1)Defined a non linear function called ‘function’.
2)Generated a list of random values of input and a list of the corresponding values of ouput.
3)Added White noise to the output list by means of RandomReal[].
4)Then found the value of the initial parameters (using the noisy data), by using the function NonlinearModelFit[].

Code for computation

Part A:

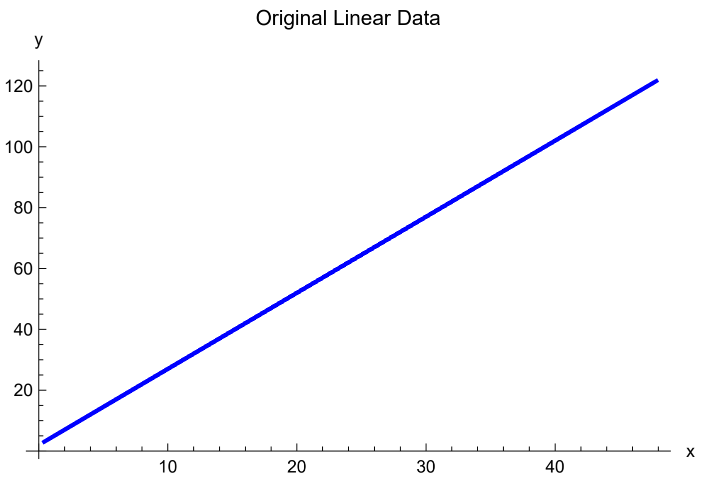
In[751]:=

```
y[m_, c_] = m * x + c
m = 5 / 2; (*Slope*)
c = 2; (*Intercept*)
n = 100;
(*Generate random x values within a certain range*)
RandomXValues = RandomReal[{0, 50}, n];

(*Calculate y values using the linear equation*)
yValues = m * RandomXValues + c;

(*Display the original linear data*)

Plot1 = ListLinePlot[Transpose[{RandomXValues, yValues}],
  PlotStyle → Blue, AxesLabel → {"x", "y"}, PlotLabel → "Original Linear Data"]
```

Out[751]=

c + m x

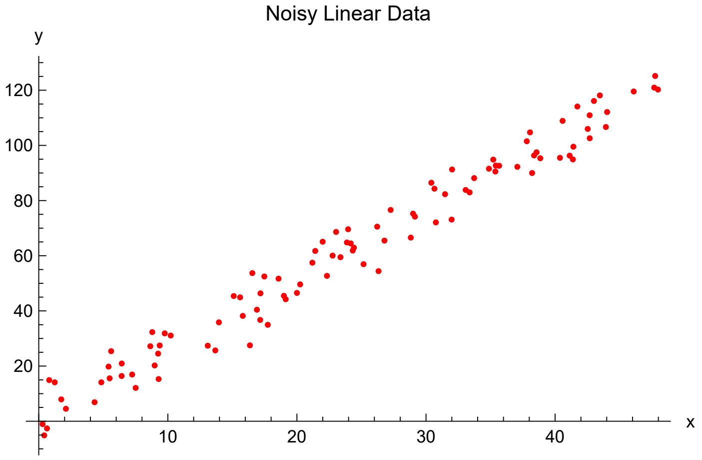
Out[757]=



In[758]:=

```
noise = RandomVariate[NormalDistribution[0, 6], n];
(*Generating a list of length=length of yValues,
of gaussian noise to add to each of the elements of yValues. *)
 NoisyY = yValues + noise; (*Adding the noise to each of the yValues*)
Plot2 = ListPlot[Transpose[{RandomXValues, NoisyY}],
  PlotStyle → Red, AxesLabel → {"x", "y"}, PlotLabel → "Noisy Linear Data"]
```

Out[760]=



In[761]:=

```
ProdXY = RandomXValues * NoisyY;
SumXY = Total[ProdXY];
SumX = Total[RandomXValues];
SumY = Total[NoisyY];
(*Manually calculating the value of the parameters of the Linear Function*)
SumXsq = Total[RandomXValues * RandomXValues];
SumXWholeSq = SumX * SumX;
Delta = n * SumXsq - SumXWholeSq;
m = (n * SumXY - SumX * SumY) / Delta
c = (SumXsq * SumY - SumX * SumXY) / Delta
```

Out[768]=

2.50518

Out[769]=

1.57788

In[770]:=

```
data2 = Transpose[{RandomXValues, NoisyY}];
func = LinearModelFit[data2, x, x]
(*Calculating the parameters using LinearModelFit[]*)
```

Out[771]=

FittedModel[1.57788 + 2.50518 x]

In[772]:=

```
f[x_] = func["BestFit"]
```

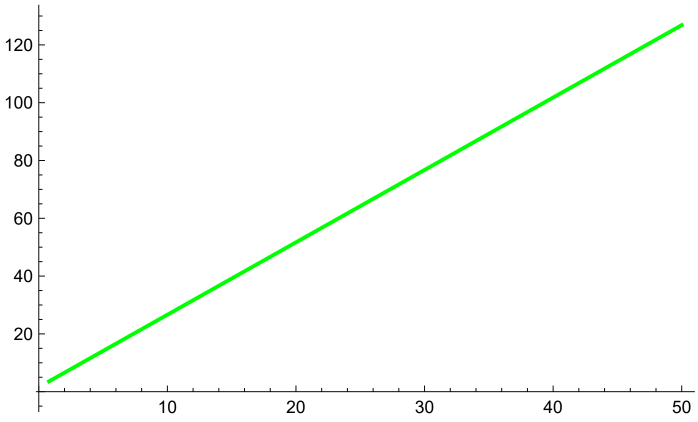
Out[772]=

1.57788 + 2.50518 x

In[773]:=

```
(*Plotting the f[x] we just obtained*)
Plot3 = Plot[f[x], {x, 4 / 5, 50}, PlotStyle → {Green}]
```

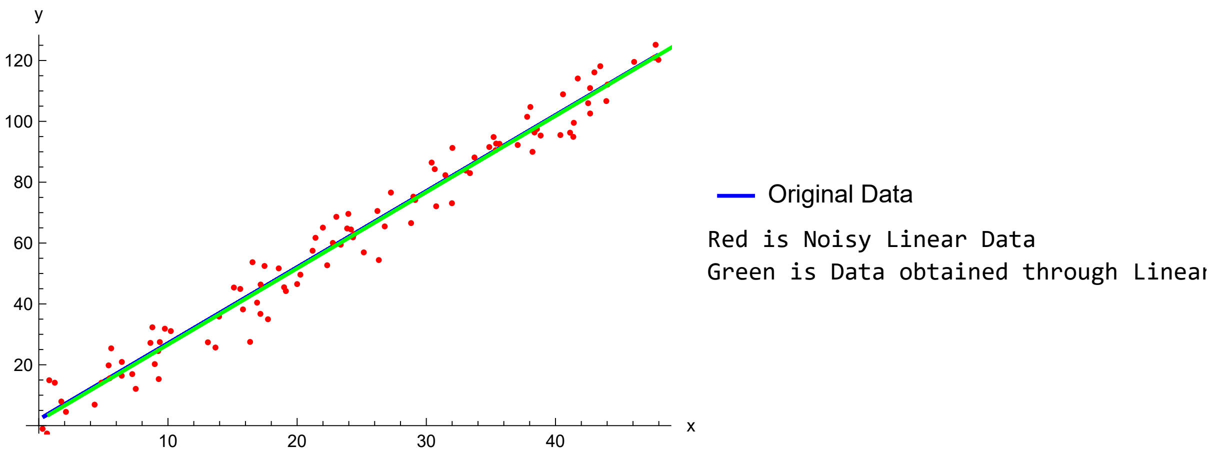
Out[773]=



In[774]:=

```
(*Plotting the original data, noisy data and the fitted function together*)
Show[ListLinePlot[Transpose[{RandomXValues, yValues}],
  PlotStyle → Blue, AxesLabel → {"x", "y"}, PlotLegends → {"Original Data"}],
 ListPlot[Transpose[{RandomXValues, NoisyY}], PlotStyle → Red,
  AxesLabel → {"x", "y"}, PlotLegends → "Red is Noisy Linear Data"],
 Plot[f[x], {x, 4 / 5, 50}, PlotStyle → {Green},
  PlotLegends → "Green is Data obtained through Linear Model Fit"]]
```

Out[774]=



In[775]:=

```
(*Using FindFit[] to get the value of the parameters*)

FindFit[data2, k * z + t, {k, t}, z]
```

Out[775]=

{k → 2.50518, t → 1.57788}

Part B:

In[776]:=

```
(*To Generate a multi-parameter (at least 3) data with Gaussian Noise.*)
f[a_, b_, c_, x_] = a * x^2 + b * x + c;
a = 2;
b = 4;
c = 5;
n2 = 50;
RandomXValues2 = RandomReal[{0, 10}, n2]; (*The list of the input values*)
yValues2 = a * (RandomXValues2) ^2 + b * (RandomXValues2) + c; (*Corresponding output*)
noise2 = RandomVariate[NormalDistribution[0, 3], n2];
(*List containing noise values*)
NoisyY2 = yValues2 + noise2;
data3 = Transpose[{RandomXValues2, NoisyY2}]; (*The noisy data*)
fit = FindFit[data3, q * r^2 + e * r + j, {q, e, j}, r]
(*Finding the fitted function using FindFit[]*)
QuadModel[q_, e_, j_] = q * r^2 + e * r + j
```

Out[786]=

{q → 1.93411, e → 4.6128, j → 3.97133}

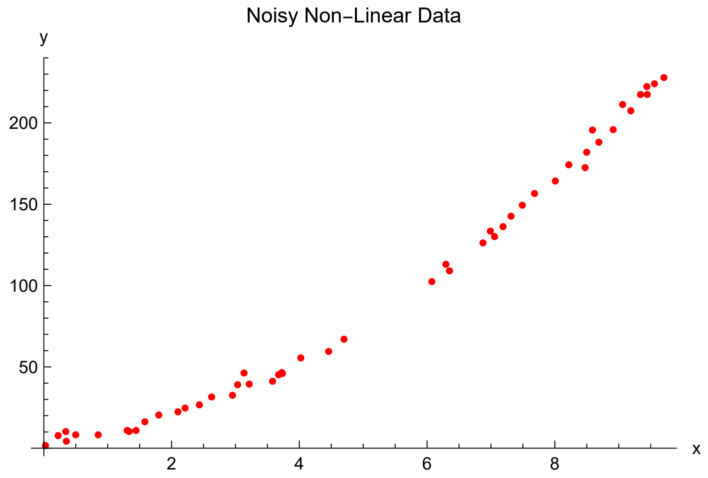
Out[787]=

$j + e\, r + q\, r^2$

In[788]:=

```
(*Plotting the noisy data*)
Plot4 = ListPlot[data3, PlotStyle → Red,
  AxesLabel → {"x", "y"}, PlotLabel → "Noisy Non-Linear Data"]
```

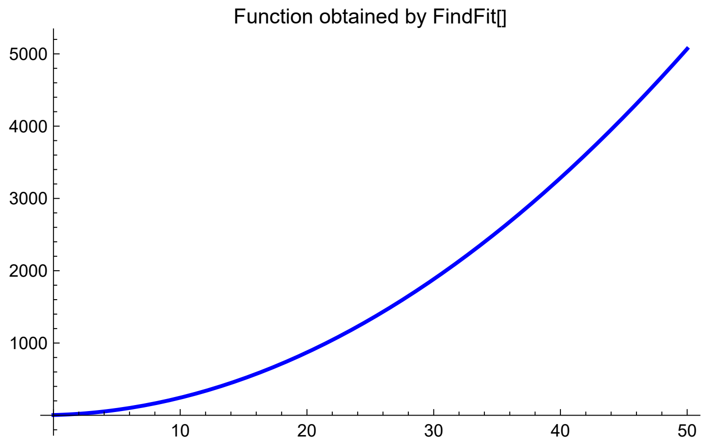
Out[788]=



In[789]:=

```
(*Plotting the fitted data we got from FindFit[]*)
Plot5 = Plot[{QuadModel[q, e, j] /. fit}, {r, 0, 50},
  PlotStyle → Blue, PlotLabel → "Function obtained by FindFit[]"]
```

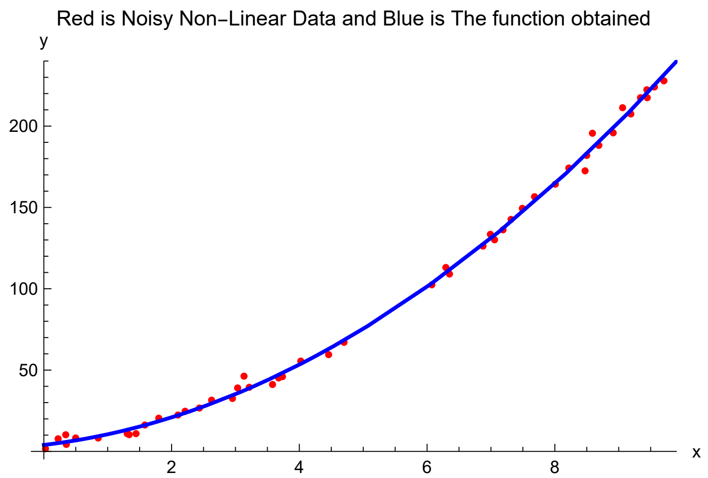
Out[789]=



In[790]:=

```
(*SHowing the Fitted data and the Noisy data together*)
Show[Plot4, Plot5,
 PlotLabel → "Red is Noisy Non-Linear Data and Blue is The function obtained"]
```

Out[790]=



In[791]:=

```
(*Using the Matrix method and LinearSolve[] to get the value of the parameters*)
Matrix1 = {};
For[i = 1, i ≤ n2, i++,
 AppendTo[Matrix1, {1, RandomXValues2[[i]], (RandomXValues2[[i]])^2}]]
Matrix2 = {{a1}, {a2}, {a3}};
Matrix3 = {};
For[i = 1, i ≤ n2, i++, AppendTo[Matrix3, {NoisyY2[[i]]}]]
Matrix1T = Transpose[Matrix1];
Matrix1 // MatrixForm;
Matrix1T // MatrixForm;
ProductMatrix = Matrix1T.Matrix1
```

Out[799]=

```
{{50., 253.822, 1790.36}, {253.822, 1790.36, 14125.5}, {1790.36, 14125.5, 117338.}}
```

In[800]:=

```
ProductMatrix2 = Matrix1T.Matrix3
```

Out[800]=

```
{{4832.15}, {36 586.8}, {299 213.}}
```

In[801]:=

```
solution = LinearSolve[ProductMatrix, ProductMatrix2]
 (*We have obtained the desired values of the coefficient of x square x and 1 *)
```

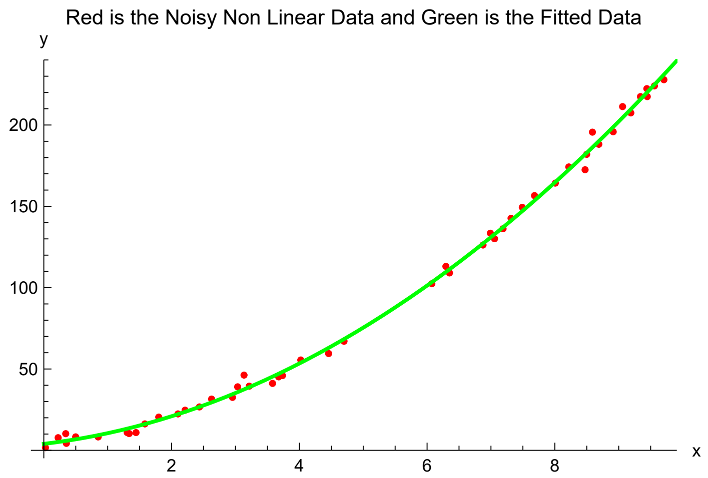
Out[801]=

```
{{3.97133}, {4.6128}, {1.93411}}
```

In[806]:=

```
Q[x_] = solution⟦1⟧ + solution⟦2⟧ * x + solution⟦3⟧ * x^2;
Plot6 = Plot[Q[x], {x, 0, 10}, PlotStyle → Green];
(*defining the function and generating the Plot of the Function
 we have obtained by doing the matrix method and LinearSolve[]*)
Show[Plot4, Plot6,
 PlotLabel → "Red is the Noisy Non Linear Data and Green is the Fitted Data"]
(*Displaying the Noisy data and the fitted data
 (Obtained from the function we have found) together*)
```

Out[808]=



Part C:

In[809]:=

```
Clear["Global`*"]
```

In[813]:=

```
function[a_, b_, x_] = a^2 * Sin[x] + b^2 / x^2 (*Declaring the Non Linear Function*)
a = 2;
b = 3;
```

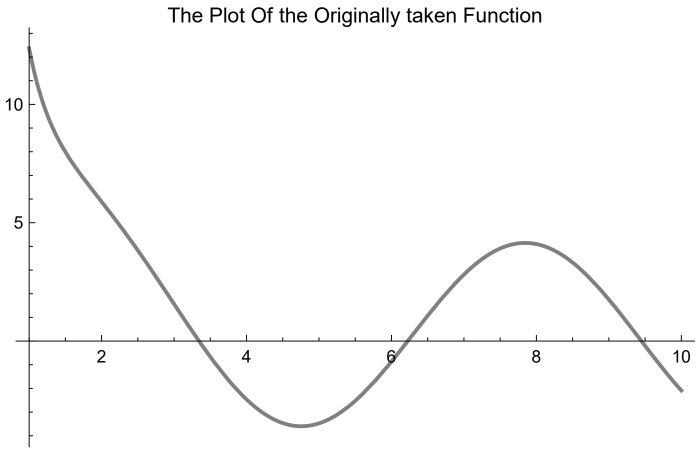
Out[813]=

$$\frac{9}{x^2} + 4\,\mathrm{Sin}[x]$$

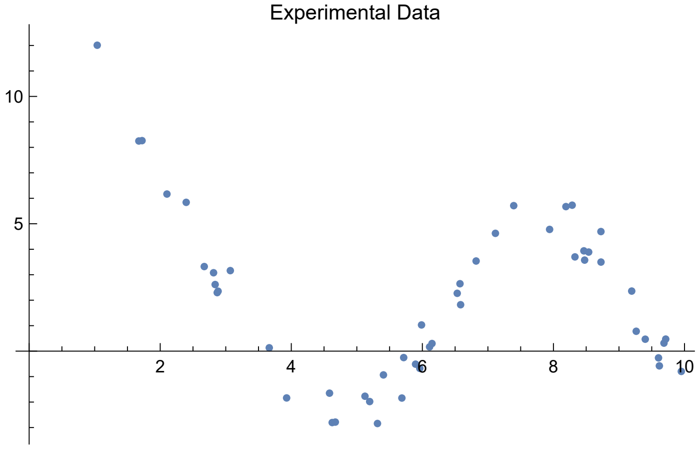
In[816]:=

```
Plot1 = Plot[function[a, b, x], {x, 1, 10}, PlotStyle → Gray,
  PlotLabel → "The Plot Of the Originally taken Function"]
(*This is the plot of the original function*)
n3 = 50;
RandomXValues3 = RandomReal[{1, 10}, n3];
YValues3 = {};
For[i = 1, i ≤ n3, i++,
 AppendTo[YValues3, a^2 * Sin[RandomXValues3[[i]]] + b^2 / (RandomXValues3[[i]]) ^2]]
NoisyY3 = {};
For[i = 1, i ≤ n3, i++, AppendTo[NoisyY3, YValues3[[i]] + RandomReal[{0, 2}]]]
(*Making a list of the Y values or the output values after adding noise to them*)

Plot2 = ListPlot[Transpose[{RandomXValues3, NoisyY3}], PlotLabel → "Experimental Data"]
(*Plotting the noisy data*)
```

Out[816]=



Out[823]=



In[824]:=

```
(*Finding the best fit function for the noisy data using NonLinearModelFit[]*)
nlm = NonlinearModelFit[
  Transpose[{RandomXValues3, NoisyY3}], c^2 * Sin[r] + d^2 / r^2, {c, d}, r]
```

Out[824]=

FittedModel[$\frac{11.1196}{r^2} + 4.0368 \, \text{Sin}[r]$]

In[825]:=

```
fittedfunction = nlm["BestFit"]
 (*Getting it in the form of a function that i can plot*)
```

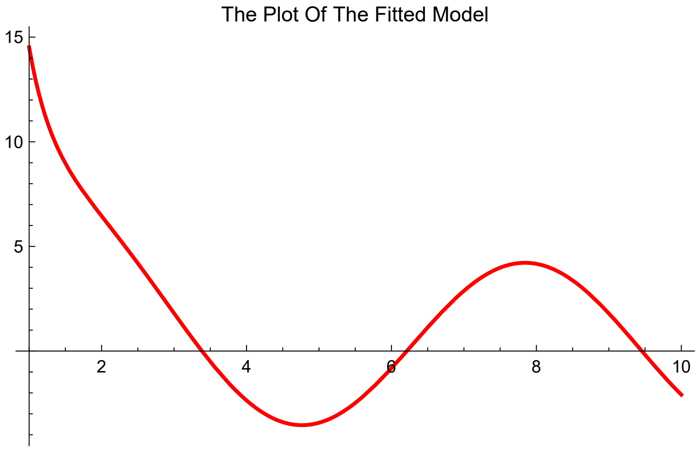
Out[825]=

$$\frac{11.1196}{r^2} + 4.0368 \, \text{Sin}[r]$$

In[826]:=

```
(*Plot of the fitted function we have obtained by doing NonlinearModelFit*)
Plot3 = Plot[fittedfunction, {r, 1, 10},
  PlotLabel → "The Plot Of The Fitted Model", PlotStyle → Red]
```

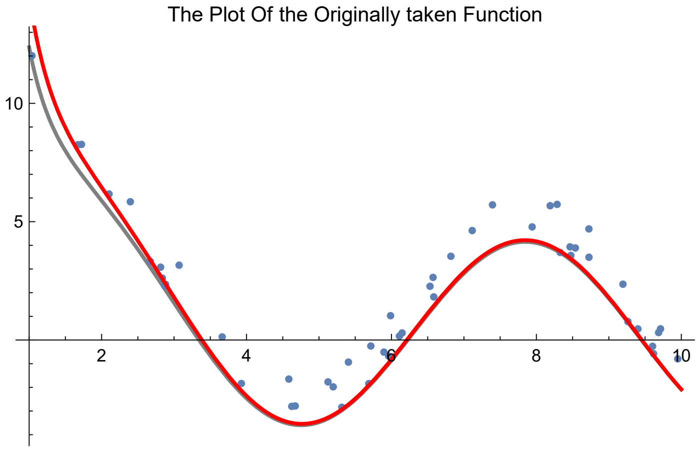
Out[826]=



In[827]:=

```
Show[Plot1, Plot2, Plot3] (*Plotting the original data,
noisy data and the fitted data (Obtained from the function we got)  all together*)
```

Out[827]=



Results

Part A:

kindly refer to output 757, 760, 773, 774, 775.

Part B:

kindly refer to output 786, 788, 789, 790, 801, 808.

Part c:

kindly refer to output 816, 824, 825, 826, 827.

Comments

This assignment was really enjoyable but lengthy. Getting the closest fit to the given data using Linear and Non Linear regression is really interesting and comes in very handy to make predictions for those input values for which we don’t know the ouput.

References

1) https://chat.openai.com/
2) https://reference.wolfram.com/language/