Assignment 3: Fitting Data To Models

Allu Yaswanth , EE18B007

February 20, 2022

Aim

- Reading data from the files and parsing them
- Analysing data to extract information.
- Study the effects of noise on the fitting process.
- Plot the graphs.

1 Extracting the data using generate-data.py code

Run the python code *generate_ data. py*, to create file *fitting. dat*.

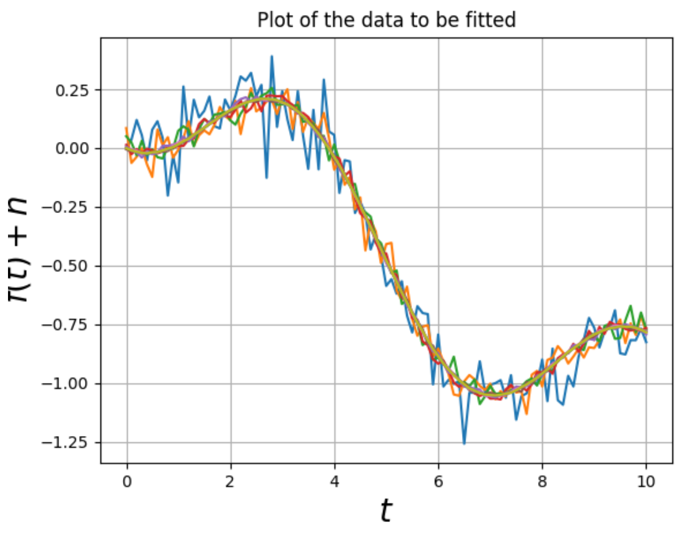

Figure 1: Data plot

The file *fitting. dat* contains data spread over 10 columns with 101 rows. The first column is the time and other nine are the noise values. The noise values are given by following function

```
sigma = logspace(-1,-3,9)
```

2 Plotting the true and noise added plots

First coloum is assigned to t as they are time values. Remaining coloums define the the 9 noisy data plots. second column is assigned to d. The function is defined here.

- f(t) =1.05J2(t)-0.105t

```
def g(t,A,B):
    return A*sp.jn(2,t) + B*t
```

By taking all 9 noisy f(t) values from data and plotting against time. from the function the true value is known and plotted.

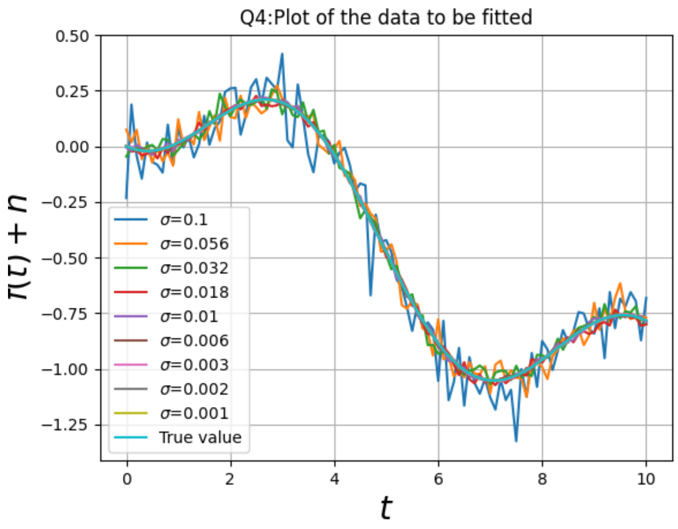

Figure 2: True and noise added plots

3 The Errorbar plot

Error bars are graphical representations of the variability of data and used on graphs to indicate the error or uncertainty in a reported measurement.They give idea about precision and shows variation from reported value to true value.In this question, the error bars are to be shown for every 5th element of first noisy data and true value.

The graph obtained by plotting every 5th data point with errorbars and the original data is as follows:

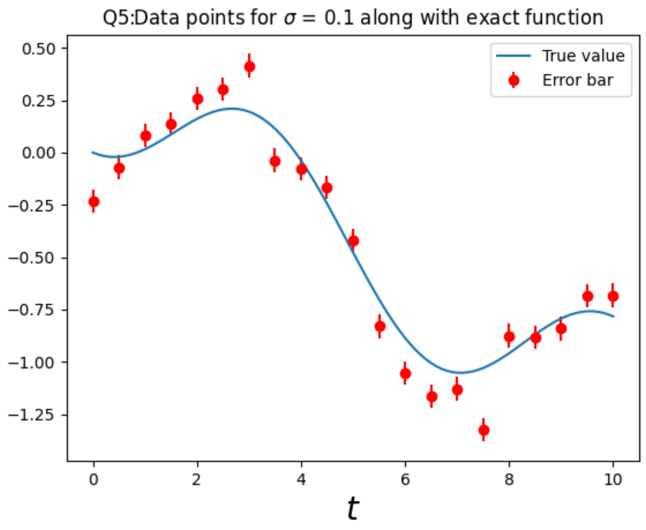

Figure 3: Error bar plot

4 The Matrix equation

The matrix M is created and multiplied with (A,B) matrix will give rise to the true value function. This can also be verified by substituting A = 1.05 and B = 0.105. I defined my own function whether the check both the arrays are equal. The is as follows

```
def equal_or_not(Q,fk):
count = 0
for i in range(N):
if Q[i] == fk[i]:
count += 1

if(count == 101):
print("The two vectors are equal.") #print as equal.
else: #else print as not equal.
print("The two vectors are not equal.")
```

5 The Mean Squared Error

The mean squared error is the error between the noisy data and the true functional data. It is calculated as follows:

$$\varepsilon_{ij} = (\frac{1}{101})\sum_{k=0}^{101}(f_k - g(t_k, A_i, B_j))^2$$

I defined a function to calculate the epsilon value, it is as follows:

```
def UpdateEpsilonValues(A,B,mean_sq_err):
    for i in range(len(A)):
        for j in range((len(B))):
            mean_sq_err[i,j] = np.mean(np.square(fk - g(t,A[i],B[j])))
```

The contour plot for ε for various values of A and B is:

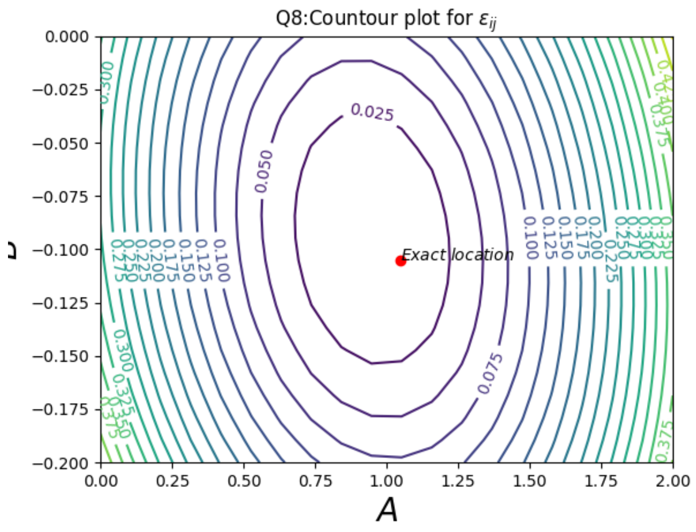

Figure 4: Contour plot

Conclusion

Only one minimum for ε.

6 Computing the Error

It is possible to try and compute the best measure for A and B from the matrix M by using the lstsq() function form scipiy.linalg. Using this we can calculate the error in the values of A and B. The python code snippet is as follows:

```
pred=[]
Aerr=[]
Berr=[]
y_true = g(t,1.05,-0.105)
for i in range(k-1):
    p,resid,rank,sig=lstsq(M,data[:,i+1])
```

```
a_temp = np.square(p[0]-1.05)
b_temp = np.square(p[1]+0.105)
Aerr.append(a_temp)
Berr.append(b_temp)
```

The plot of the error in A and B against the noise standard deviation is:

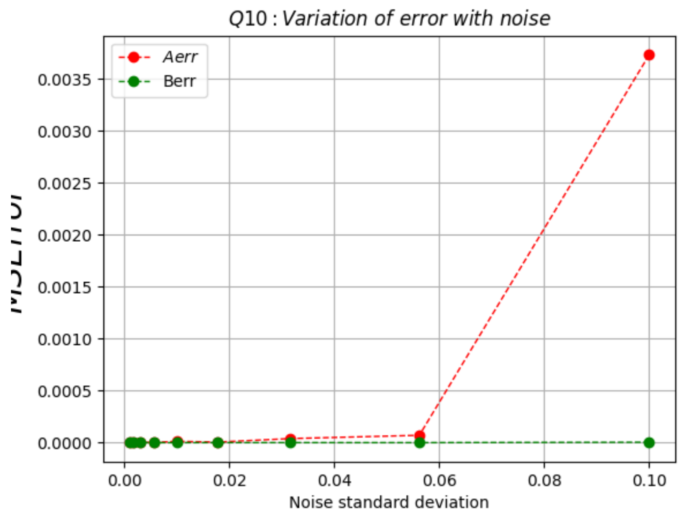

Figure 5: Error vs Standard deviation

We can also plot the same graph in log scale. This plot is shown below:

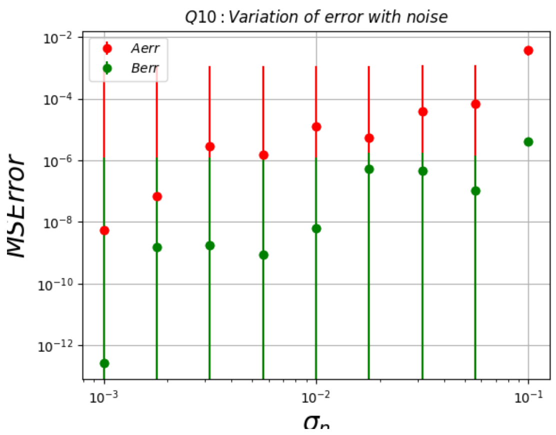

Figure 6: Error vs Standard deviation: log scale

Conclusion

From Figure 5, it is evident that the plot is not at all linear. Also from Figure 6, it is made sure that the plot is not linear in the logscale case too. Hence, in both the cases, the graph is non-linear.

Inference

The given noisy data was extracted and the best possible estimate for the underlying model parameters were found by minimizing the mean squared 6 error. This is one of the most general engineering use of a computer, modelling of real data. The method of least squares assumes that the best fit curve of a given type is the curve that has the minimal sum of deviations, i.e., least square error from a given set of data. It reduces the error and gives the best fitting data.