EE2703: Applied Programming Lab
Assignment 3
Fitting Data to Models

Kaushik Ravibaskar
EE20B057

February 18, 2022

Contents

1 Aim **2**

2 Introduction **2**

3 Generating Random Data **2**

4 Parsing Data and Plotting Given Data **3**

5 Using Error Bar Plot **4**

6 Matrix Generation **4**

7 Calculating Mean Squared Error and Plotting Contours **5**

8 Plotting Error in A and B **7**

9 Conclusion **9**

1 Aim

The assignment mainly deals with the following points:

- Reading `fitting.dat` file and to parse the data from it.
- Analyze the data and to extract essential information from it.
- Generate a **linear model** of the given data and to analyze the effect of noise on it.
- Plot graphs to get better insights about the data and to make sound comments upon it.

2 Introduction

The learning objective of this assignment is to understand the **linear regression** process in an elementary stage. We have been given a randomized noise filled data, and we are told to find the best possible way to make a linear model of it. In the process, we are introduced to a famous mathematical method, **least square method**, which helps us to find the right linear relation

3 Generating Random Data

The assignment starts with generating randomized data as a linear combination of `Bessel` function and the input variable by running the `generate_data.py` file provided to us. To encapsulate real life signals, a normally distributed noise with standard deviation sampled uniformly on a logarithmic scale has also been added into the data. After running the `generate_data.py` file, we get `fitting.dat` file which contains the data that we have to work with.

The `fitting.dat` contains 101 rows and 10 colomns, the first colomn having time t values, and rest 9 colomns having the function:

$$f(t) = 1.05J_2(t) - 0.105J(t) + n(t)$$

where $J_2(t)$ is the `Bessel function` and $n(t)$ is the randomized noise:

$$P(n(t)|\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-n(t)^2}{2\sigma^2}\right)$$

generated over $\sigma = logspace(-1, -3, 9)$

4 Parsing Data and Plotting Given Data

In python, with the help of `numpy` module, we have used `np.loadtxt()` function to store the data from the `fitting.dat` file as a list of elements. Then we have separated the time and data components into respective matrices as follows:

```
data_lines = np.loadtxt("fitting.dat", dtype=float)
given_data = np.array(data_lines[:, 1:])
t_array = np.array(data_lines[:, 0])
```

After storing the data, we have used the `pylab` module to make plot of the data colomns along with the true plot. True plot is obtained by putting time values directly into the $f(t)$ function. The plot will look as follows:

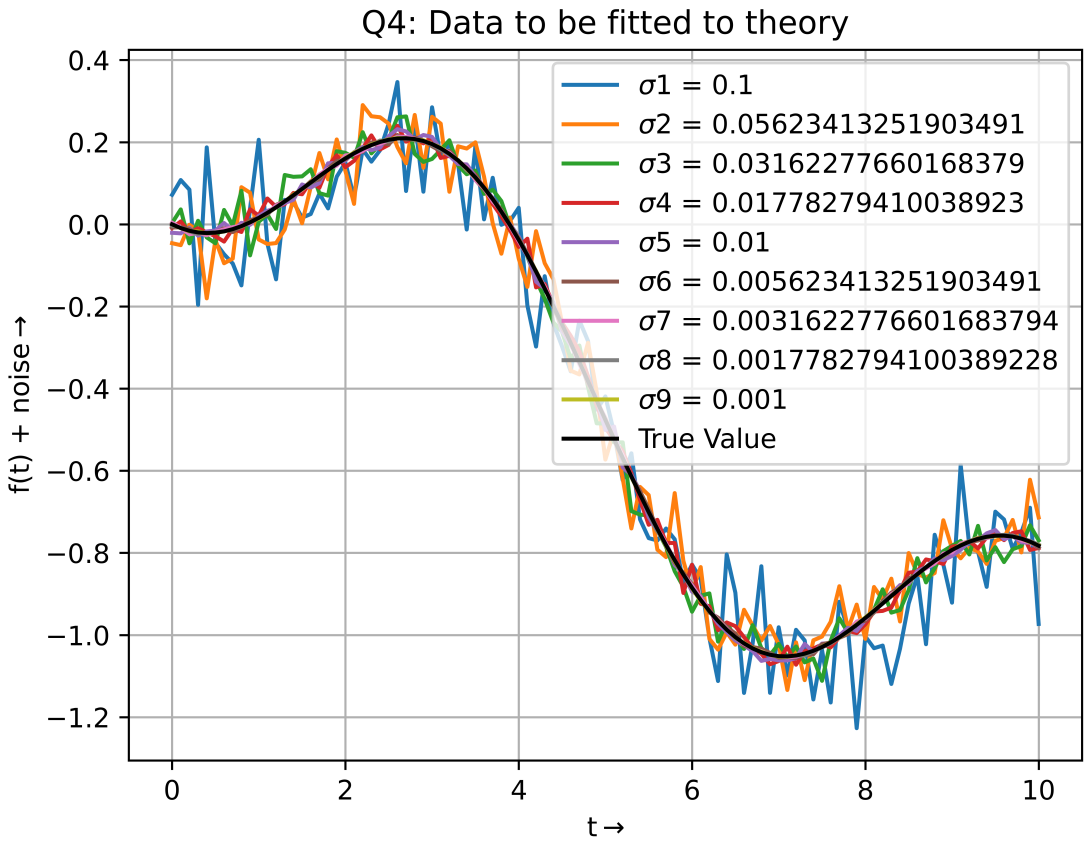

Figure 1: Plot of all data colomns

5 Using Error Bar Plot

Error bars are a very powerful graphical method to visualize errors in a plot. In here, we have plotted the error bar plot of the 1^{st} colomn along with the **true function** plot to give a sense of the error in the former data.

The main aim of the assignment is to find the values of A and B in the function $f(t) = AJ_2(t) + B(t) + n(t)$ that best fits the data given in different colomns. The error bar plot is as follows:

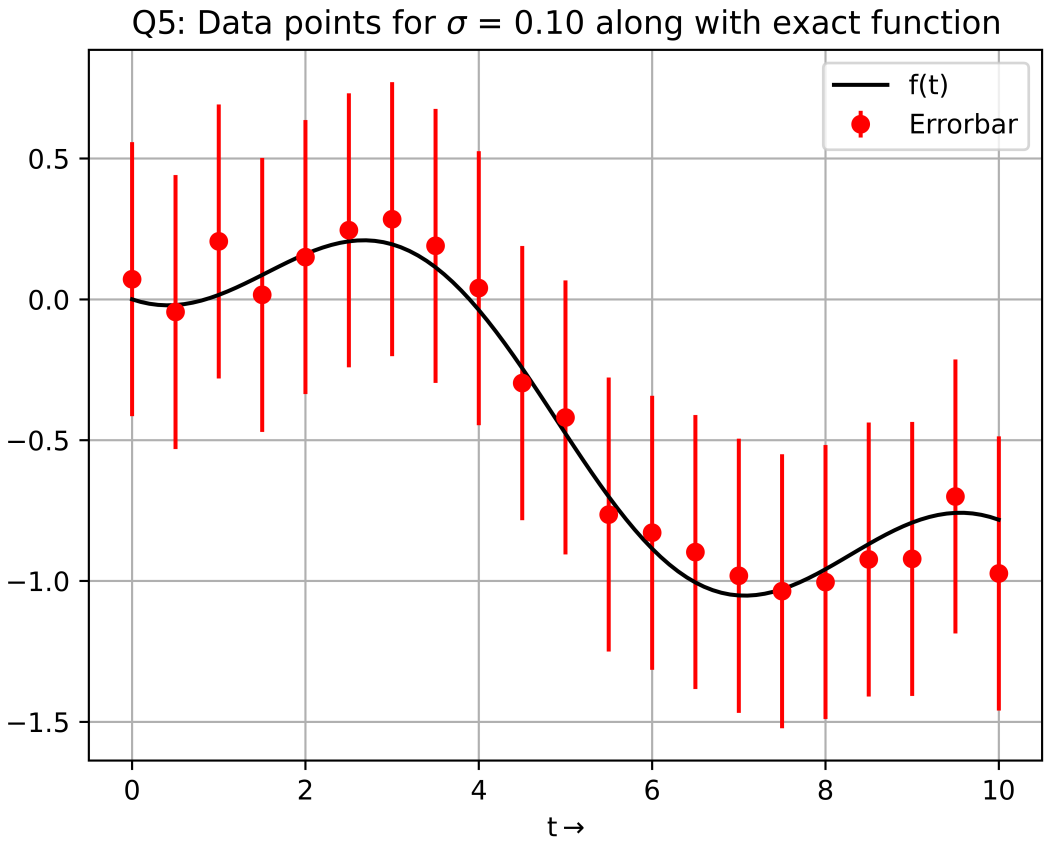

Figure 2: Errorbar plot for the 1^{st} colomn of the data array

6 Matrix Generation

We will now generate the matrix M, which will comprise of $J_2(t)$ values in the 1^{st} colomn and t values in the 2^{nd} colomn as follows:

$$M = \begin{bmatrix} \mathrm{J}_2(t_1) & \mathrm{t}_1 \\ ... & ... \\ \mathrm{J}_2(t_n) & \mathrm{t}_n \end{bmatrix} \tag{1}$$

We will also create a vector p which will contain the values A and B, which we want to find out.

$$p = \begin{bmatrix} \mathrm{A} \\ \mathrm{B} \end{bmatrix} \tag{2}$$

We are interested in the product $M * p$, which we will equate with each colomn in the given data, so as to obtain A and B that best fits the **linear** model. Let us define $g(t, A, B)$ as follows:

$$g(t, A, B) = M * p$$

7 Calculating Mean Squared Error and Plotting Contours

Mean square method is a very widely used method in the domain of statistics and data analysis. It is used to get qualitative insights into error and deviation of **actual data** from the *required data*.

We will consider the 1^{st} colomn of the data array to compute the mean squared error and plot the contour. The main aim is to find A and B which will give the least mean square error upon computation.

Hence, we will define ranges as, $A = 0, 1, \ldots, 2$ and $B = -0.2, -0.19, \ldots, 0$. We can compute the error function matrix as follows:

$$\epsilon_{ij} = \frac{1}{101}\Sigma((f_k - g(t, A, B))^2$$

where $k = 0, 1, 2, \ldots, 100$ and f is the 1^{st} colomn of the data matrix. The corresponding code in python will be:

```
A_array = py.arange(0, 2.1, 0.1)
B_array = py.arange(-0.2, 0.01, 0.01)
e_list = []

for a in A_array:
for b in B_array:
e_list.append(mn_sum_sq_diff(given_data[:, 0], t_array, a, b))
e_list = py.array(e_list)
e_list = e_list.reshape((py.size(A_array), py.size(B_array)))
```

where `mn_sum_sq_diff` function is as follows:

```
def mn_sum_sq_diff(in_arr, t_arr, val1, val2):
    temp_val_1 = (in_arr - g(t_arr, val1, val2))*(in_arr - g(t_arr, val1, val2))
    temp_val_sum = py.sum(temp_val_1)
    return (temp_val_sum/py.size(in_arr))
```

The **mean squared error** contour plot between A, B and ϵ is as follows:

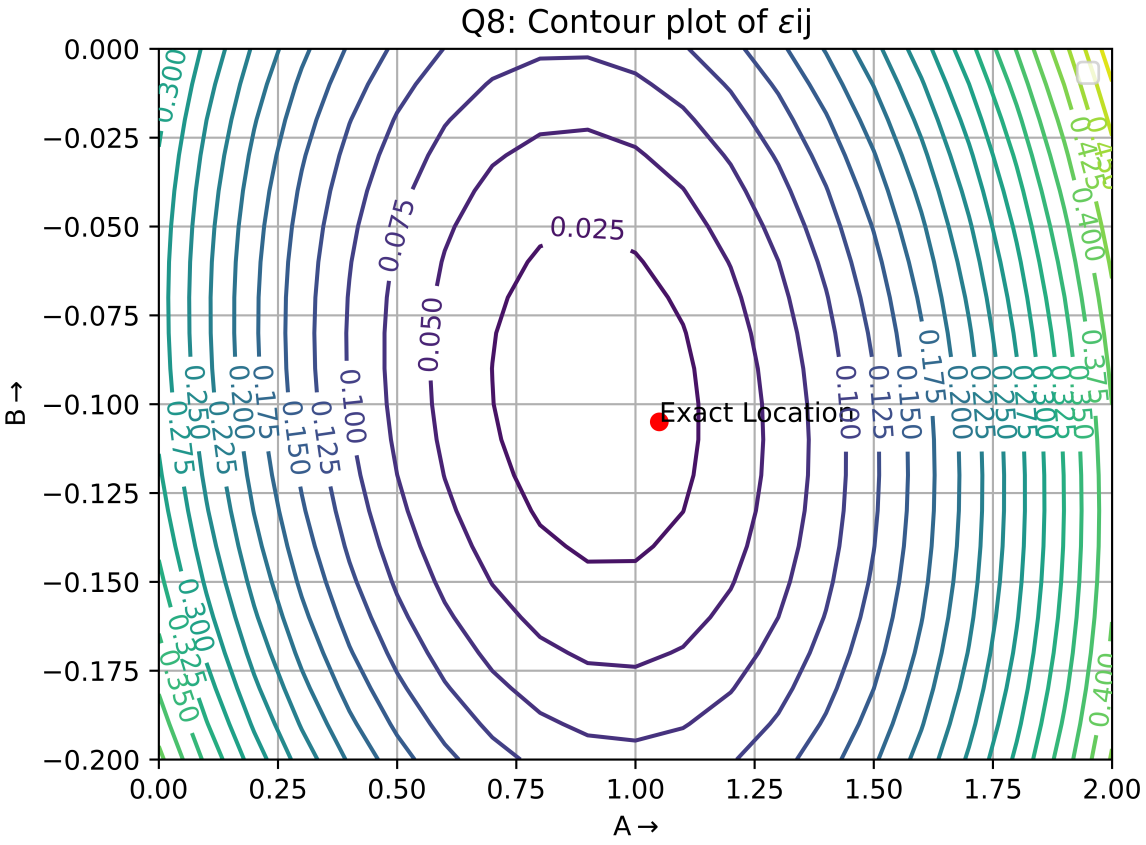

Figure 3: Contour plot of MS error

As can be seen from the contour plot, we get a minima in ϵ near around the point $(1.05, -0.105)$. Hence mean squared error analysis has proved out to be a useful method to give best fit values of A and B for the linear model.

8 Plotting Error in A and B

We can carry forward our process that we did above to all the other colomns of data matrix to obtain 9 values of A and B, one for each colomn. Python provides a very powerful function that gives us the value of A and B by least square method, given a matrix M and data colomn f as arguments to it. The code is as follows:

```
app_A = []
app_B = []
sigma_noise = py.logspace(-1, -3, 9)
for i in range(len(data_lines[0])-1):
    temp_app = sc.linalg.lstsq(M, given_data[:, i])
    app_A.append(temp_app[0][0])
    app_B.append(temp_app[0][1])
app_A = py.array(app_A)
app_B = py.array(app_B)
err_A = abs(app_A - 1.05)
err_B = abs(app_B + 0.105)
```

The above code also computes the absolute error between the **obtained value** and the **true value** of A and B. The error plot of A and B as a function of σ is as shown below:

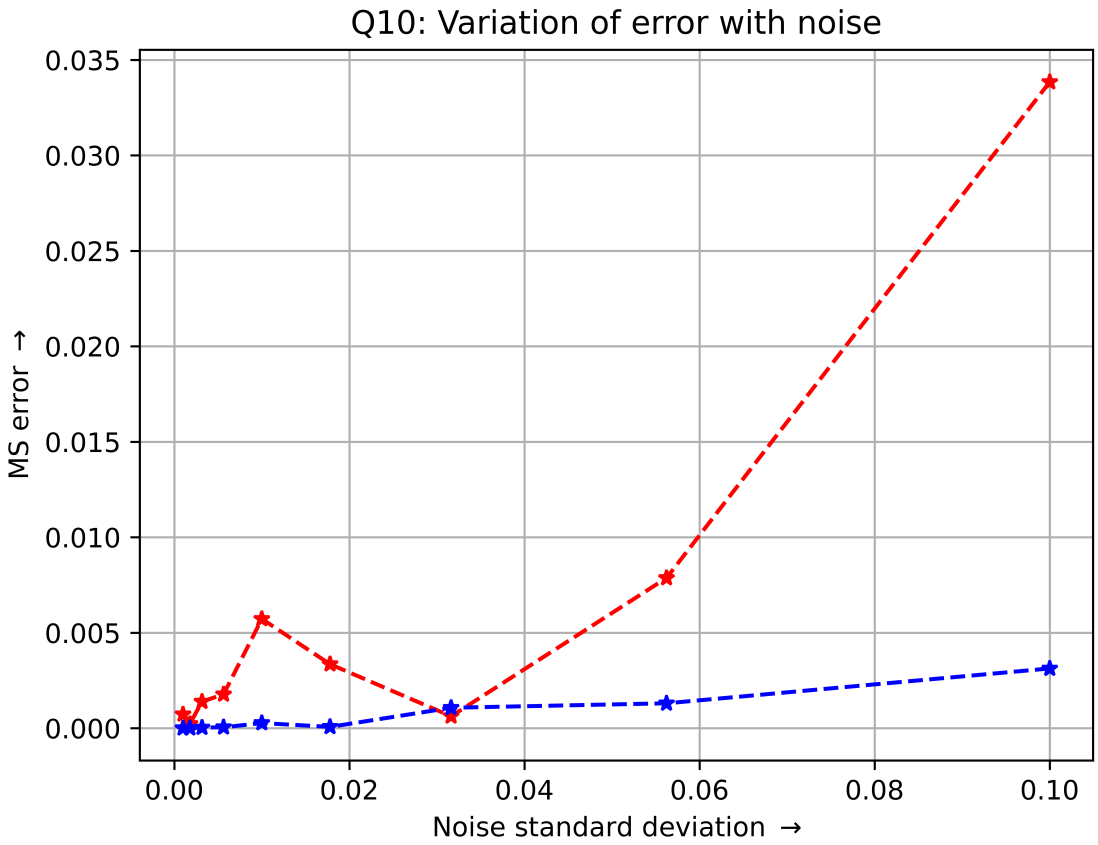

Figure 4: Error in A and B vs σ

On plotting on a **log-log** scale, as shown below, we can see that the plot is almost linear!

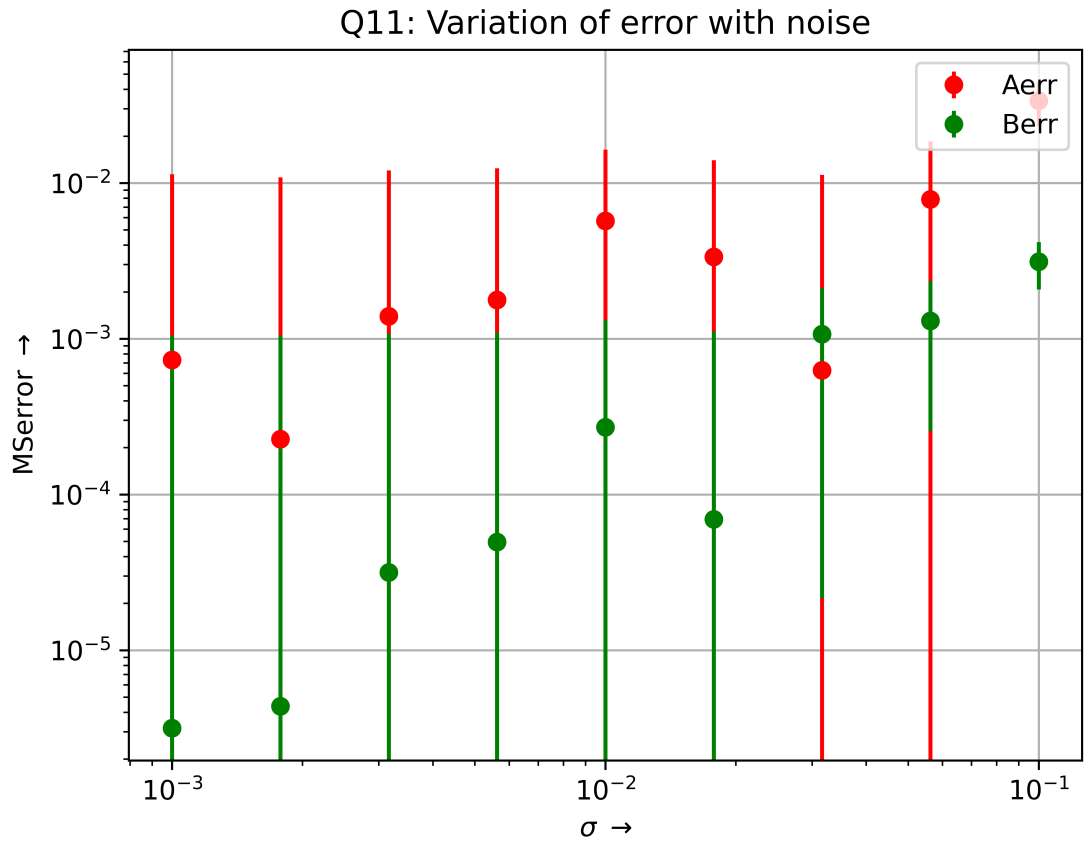

Figure 5: Error in A and B vs σ in **log-log** scale.

9 Conclusion

Hence, we can conclude that:

- The best possible **linear** fit for the data filled with randomized noises would be to minimize the least mean squared error.
- We can observe that error in A and B increases linearly with increase in σ on a log scale.