Applied Programming Lab: Assignment 3

Harish R EE20B044

February 21, 2022

Contents

1 Abstract

The primary aim of the exercise is to study how the **amount of noise** affects the quality of estimation in the process of fitting models (linear here) to data. Further we also learn how to **extract useful information** by analyzing the data and present the same through various kinds of plots. In the of doing so, we learn the usage of various libraries in python useful for data analysis and visualization.

2 Introduction

Our interest here is the special case of model parameter estimation, where the model is *linear in parameters*, i.e,

$$f(t; p_1, p_2, .., p_N) = \sum_{i=0}^{N} p_i F_i(t) \tag{1}$$

Say, we have M data observations and N parameters to estimate, this problem clearly reduces to the inversion of a matrix problem:

$$\begin{pmatrix} F_1(t_1) & F_2(t_1) & ... & F_n(t_1) \\ F_1(t_2) & F_2(t_2) & ... & F_n(t_2) \\ ... & ... & ... & ... \\ F_1(t_M) & F_2(t_M) & ... & F_n(t_M) \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ ... \\ p_N \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ ... \\ a_M \end{pmatrix} \tag{2}$$

Since our real world measurements are always noisy, Accounting for the noise, we get the following matrix equation in vector form:

$$F.\vec{p} = \vec{x_0} + \vec{n} = \vec{x} \tag{3}$$

Since a generalized solution to this problem is extremely non-trivial, We wish to simplify things by making specific assumptions about the noise present in the data:

1. The noise added to each observation x_i, namely n_i is *independent* of the noise added to any other observation.

2. Further the noise is uniform and normally distributed, i.e, it has zero mean and a standard deviation of σ.

Proceeding further, we try to get the best guess for , by minimizing the L_2 norm of the error that is given by:

$$\epsilon = F.\vec{p} - \vec{x} \tag{4}$$

The norm of the error is:

$$\|\epsilon\|^2 = \epsilon^T \epsilon = ((F.\vec{p} - \vec{x})^T (F.\vec{p} - \vec{x})) = \sum_i \epsilon_i^2 \tag{5}$$

On plotting the error term, we would expect the surface plot to have a minimum at some point . When we proceed to find it by taking the gradient and setting it to zero, We get the following result:

$$\vec{p_0} = (F^T F)^{-1} F^T \vec{x} \tag{6}$$

This is called the *Least Squares Estimates* method.

3 Procedure

Let us take the following function to be the true data which we are supposed to fit linearly.

$$f(t) = 1.05 J_2(t) - 0.105t \tag{7}$$

where $J_2(t)$ is the *Bessel function of order 2.*

3.1 Adding noise to the data

Based on our previous assumption, We add a noise series that is normally distributed, whose probability distribution is given by

$$P(n(t)|\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{n(t)^2}{2\sigma^2}} \tag{8}$$

The below code snippet generates 9 columns of noisy data corresponding to 9 linearly spaced values of σ in logspace (from 10^{-3} to 10^{-1}):

```
def generate_data():
    N=101                           # no of data points
    k=9                             # no of sets of data with varying noise

    # generate the data points and add noise
    t= np.linspace(0,10,N)              # t vector
    y=1.05*sp.jn(2,t)-0.105*t       # f(t) vector
    Y=np.meshgrid(y,np.ones(k),indexing='ij')[0] # make k copies
    scl=np.logspace(-1,-3,k)           # noise stdev
    n=np.dot(np.random.randn(N,k),np.diag(scl))     # generate k vectors
    yy=Y+n                          # add noise to signal

return t,yy
```

3.2 The plot function

We extensively use the **pyplot** module from matplotlib library for various kinds of visualization. The following code snippet is a generalized plot function, when evoked with appropriate inputs, can return the plot figures:

```
def plot_data(fn,x,y,xl,yl,title,legends=None,stdev=None,z=None,loglog=False):

    parameters = {'axes.labelsize': 12,
       'axes.titlesize': 15,
       'legend.fontsize': 10,
       'mathtext.fontset':'cm'}

    plt.rcParams.update(parameters)
    fig, ax = plt.subplots(figsize =(8,8))
    if fn=='line':
     ax.plot(x,y)
    elif fn=='err':
     ax.errorbar(x,y,stdev,fmt='ro')
    elif fn=='contour':
     cont = contour(x,y,z,20)
     clabel(cont, np.linspace(0.025,0.100,4), inline=True)
    elif fn=='dashed':
     ax.plot(x,y,linestyle='dashed',marker='o', markersize=5)

    if loglog:
     ax.set_xscale('log')
     ax.set_yscale('log')

    if legends:
     ax.legend(tuple(legends), loc='upper right')
```

```
        ax.set(xlabel=xl, ylabel=yl, title=title)
        ax.grid()

    return fig
```

3.3 Visualizing the noisy data

Using the above described plot function, we now visualize the noisy data generated. The variation in levels of noise with respect to σ_i is apparent in the below graph.

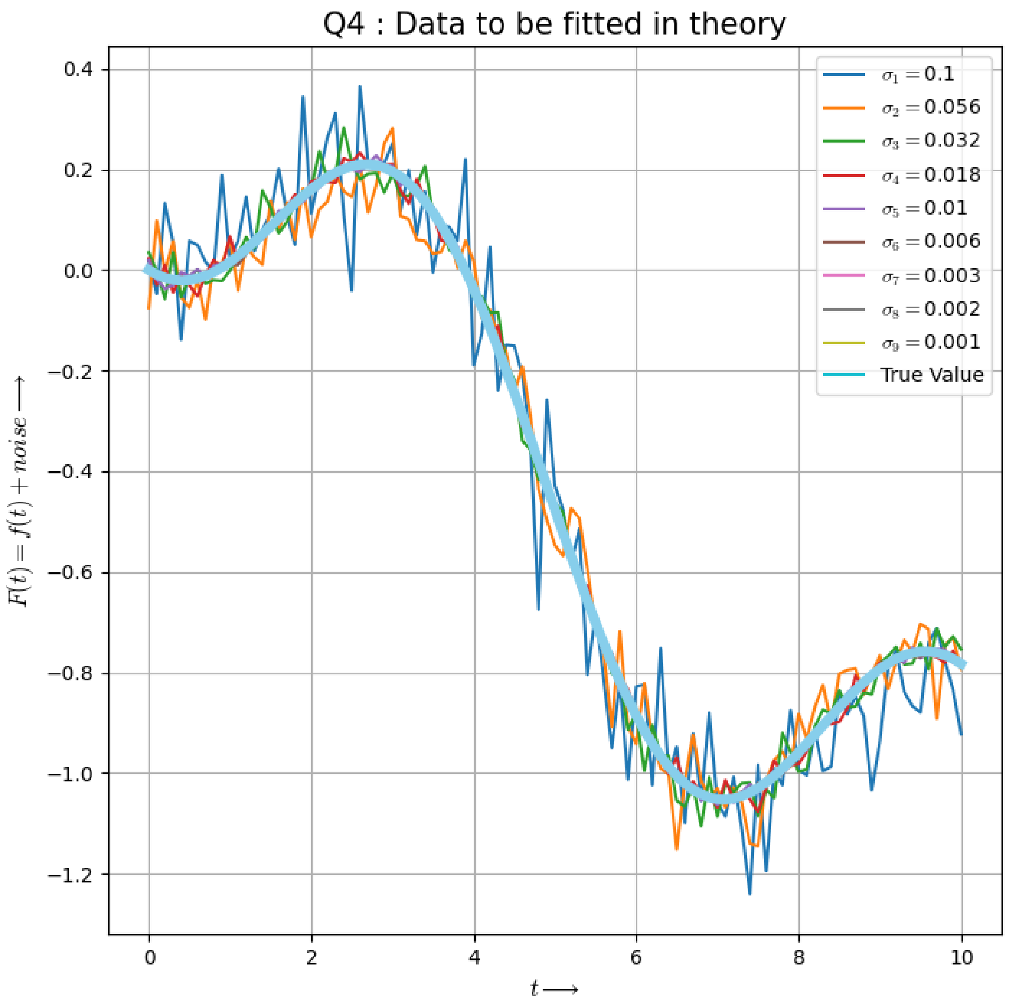

Figure 1: Noisy Data with True Data

A plot of the first column of data with error bars is also generated alongside the exact curve, by plotting every 5th data item to make the plot readable. This gives us an idea on how much the data diverges.

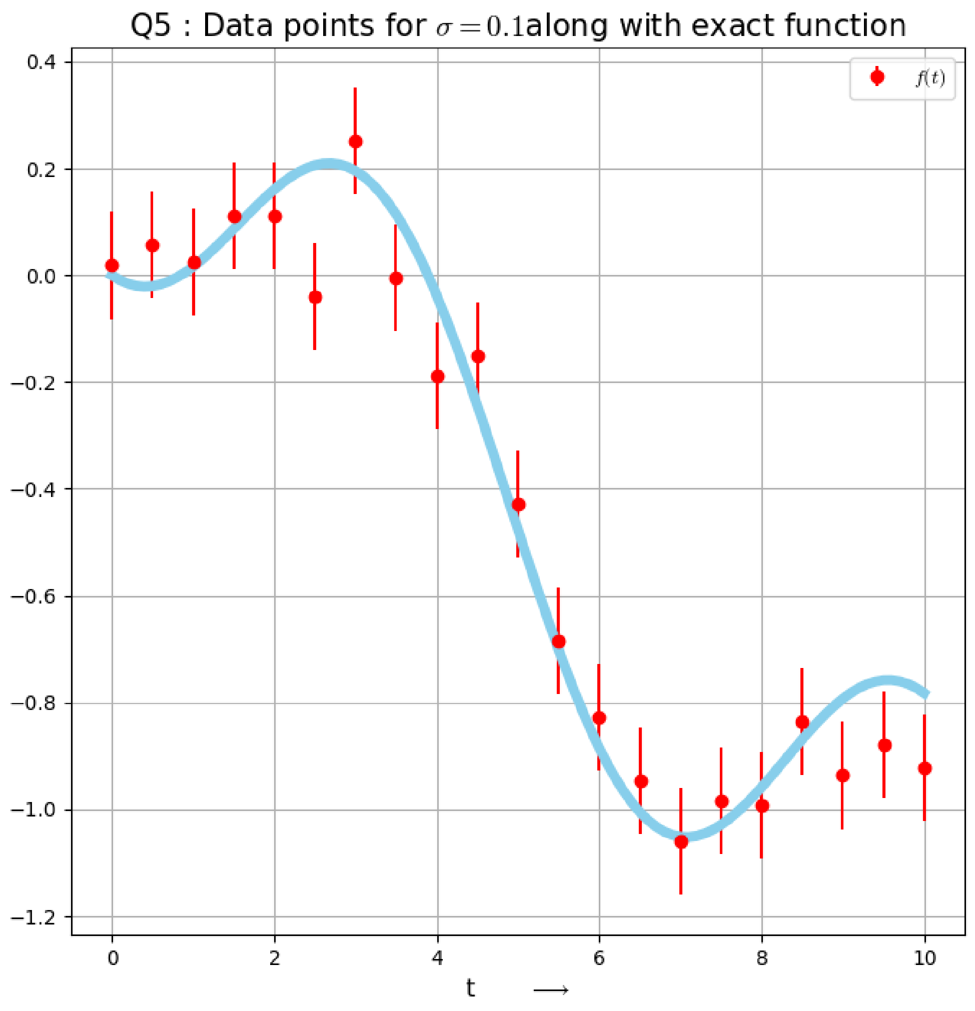

Figure 2: Noisy Data with Errorbar

3.4 Linear fit the data

The true function can be modelled as

$$g(t, A, B) = AJ_2(t) + Bt \tag{9}$$

where A and B are constants that we need to find.

We begin by guessing values for A and B in the ranges of $(0, 2)$ and $(-2, 0)$ respectively. The error in the estimate is computed as shown below

$$\epsilon_{ij} = \frac{1}{101}\sum_{k=0}^{101}(f(t_k) - g(t_k, A_i, B_j))^2 \tag{10}$$

where ϵ_{ij} is the error for (A_i, B_j).
When we plot the variation of error function in the meshgrid of A and B, we find the below contour plot with a **Minima at** $(A, B) = (1.1, -0.11)$ which is very close to the exact solution.

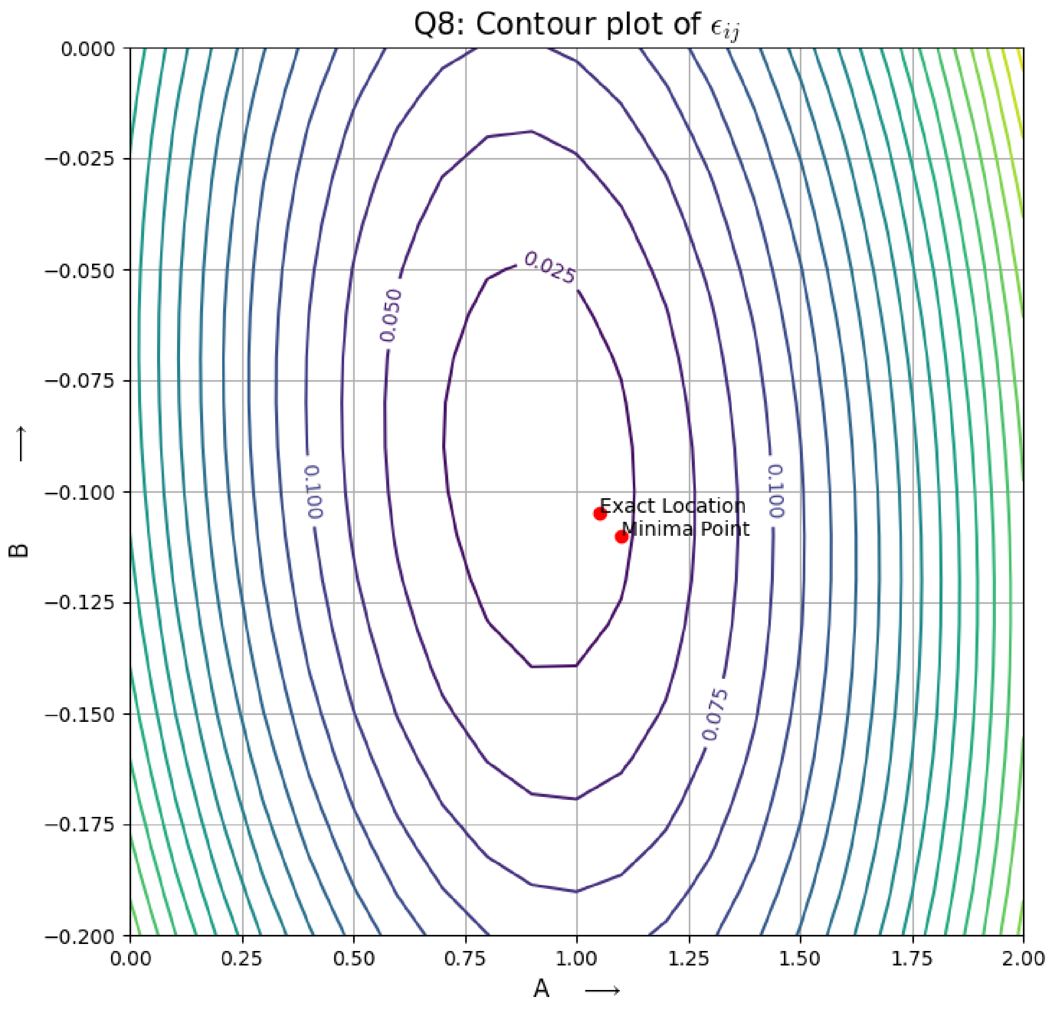

Figure 3: Contour Plot of ϵ_{ij}

As discussed earlier, we perform the least squares fit to the function using `linalg.lstsq` function available in scipy module. This solved for A_{fit} and B_{fit} as described by the below equation.

$$M.p = D \tag{11}$$

where

$$M = \begin{bmatrix} J_2(t_1) & t_1 \\ \dots & \dots \\ J_2(t_m) & t_m \end{bmatrix},\ p = \begin{bmatrix} A_{fit} \\ B_{fit} \end{bmatrix} \text{ and } D = \begin{bmatrix} f(t_1) \\ \dots \\ f(t_m) \end{bmatrix} \tag{12}$$

3.5 Finding out the variation of ϵ with σ_n

We proceed to generate 100 different files of the `fitting.dat` data with different noises and get the estimates A_{fit} and B_{fit} using the procedure described in *Section 3.3*. Then we get the mean squared error of A_fit and B_fit across the datafiles for a given σ_i. This MSE variation with σ_n is captured in the below graph

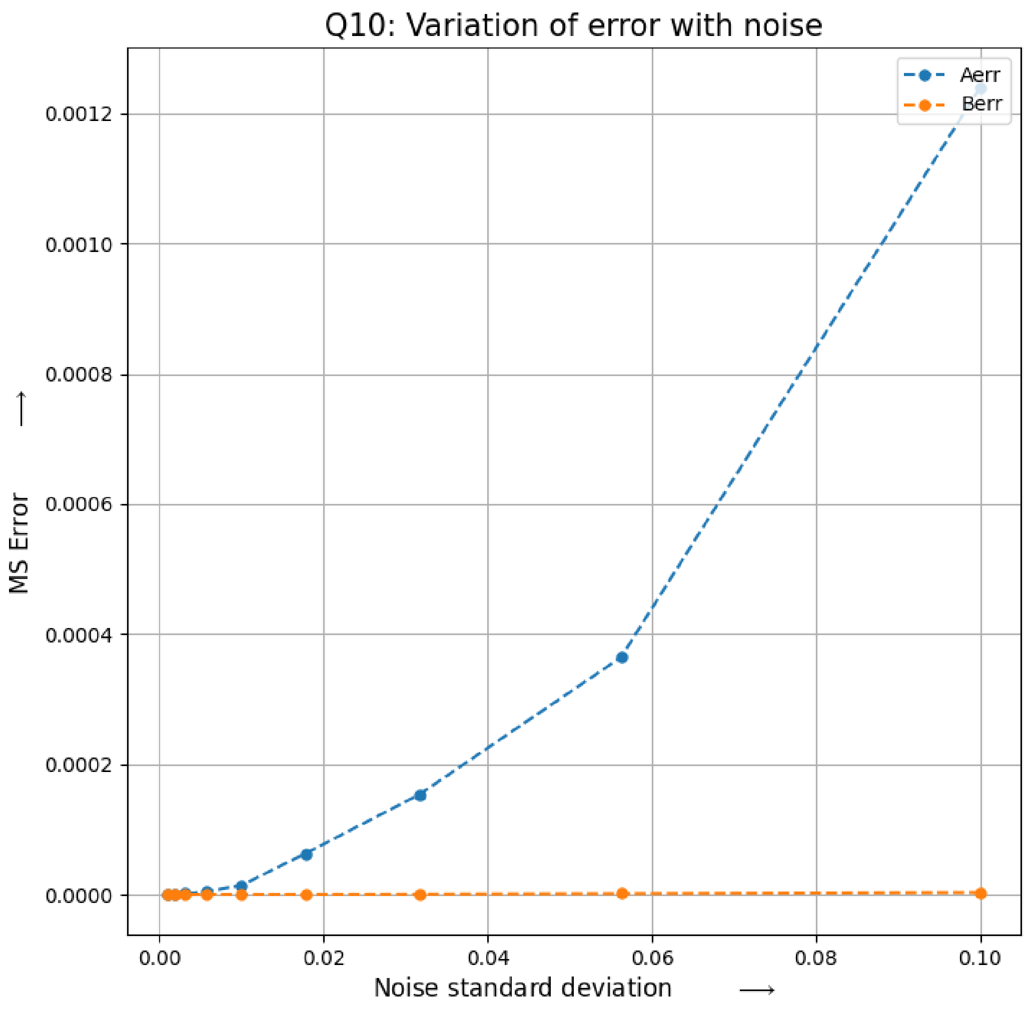

Figure 4: Mean Squared Error vs Standard Deviation

Since the above plot is not directly inferential in nature, We plot the same in `loglog` scale as below:

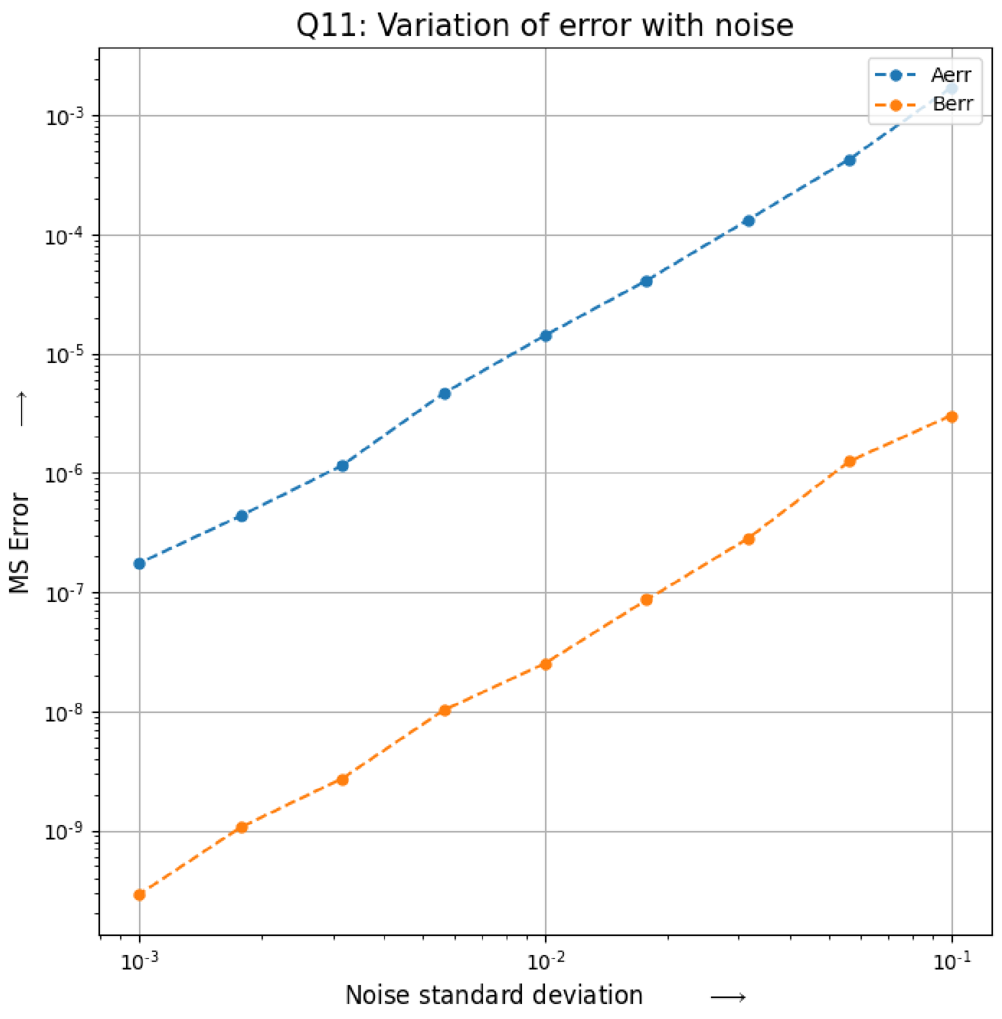

Figure 5: Error vs Standard Deviation `loglog` Plot

It is evident from the above graph that the relationship between $log(\epsilon_{std})$ and $log(\sigma_{data})$ is approximately linear.

4 Conclusion

We can conclude from the exercise that, the logarithm of error in the least squares fit estimate is **linearly** proportional to the standard deviation of noise in the data.

5 Acknowledgement

The following sources were referred while preparing the report:

1. *Section: Linear Fitting to Data*, Assignment Handout, EE2703.
2. *MathTex : LaTeXfor Matplotlib*,(MathTex Documentation)