Assignment No 3

Shivanshu Shekhar

February 18, 2022

1 Abstract

This week's Python assignment will focus on the following topics:

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

2 Introduction

For this weeks assignment we are going to use the **Bessel Function** and **Gaussian noise** to study the variation of parameters of a model with changing **Standard Deviation**.

$$f(t) = A * J_2(t) + B * t + n(t)$$

Where, A = 1.05, B = -0.105, J_2 = Bessel function, n(t) = Noise function

We seek to find the relation between A,B and the standard deviation of the Gaussian noise.

3 Problems and Solutions

3.1 Generation Data

Data is generated using the script given with the assignment, **scipy** library is used for getting the **Bessel function**, and t contains 101 equally spaced numbers between 0 and 10, which are fed into the Bassel function and added with noise to generate the data used further.

3.2 Loading data

Using numpy's loadtxt funtion we loaded the data into a numpy matrix, the data had 10 columns and 101 rows, the first column had the values of time in them and the remaing columns had data produced using standard deviation for the noise.

```
data = np.loadtxt(FILENAME) #Loading data
#Initializing variables
yy = data[:,1:]
t = data[:,0]
k = data.shape[1] - 1
sigma = logspace(-1,-3,k)
```

3.3 Plotting Data

The plotting of data in this assignment is done using **pylab's** plotting functions, which is nothing but the **Matplotlib' pyplot module**

3.4 Plotting the original function

We plot $f(t)$ without the noise using A = 1.05 and B = -0.105

```
def g(t, A = 1.05, B = -0.105): #Noise less function
    return A * sp.jn(2, t) + B*t


y_t = list(map(g, t)) #Noiseless values
#Plotting
figure(figsize=(10,8))
plot(t,yy, label = sigma)
plot(t, y_t, label = "True Value", color = "k")
legend([f"$\sigma_{i + 1}$ = {sig:.3f}" for i, sig in enumerate(sigma)] + ["True value"])
xlabel(r'$t\rightarrow$',size=20)
ylabel(r'$f(t)+noise\rightarrow$',size=20)
title("Q4: Data to be fitted to theory", size=20)
grid(True)
show()
```

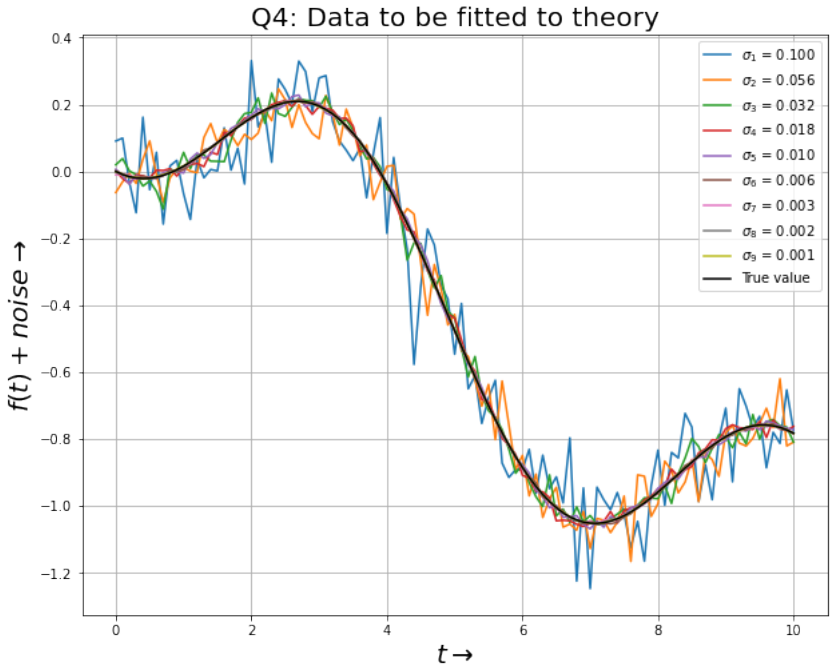

Figure 1: Question 3&4

3.5 Plotting Error Bars

We plot the first column of data with error bars using the **errorbar** function with error equal to standard deviation.

We plotted every 5^th data item to make the plot readable, and we also plotted the exact curve to see how much the data diverges.

```
#plotting
figure(figsize=(10,8))
plot(t, y_t, color = "k")
errorbar(t[::5],yy[::5,0],sigma[0],fmt="ro")#yerr = sigma
legend(["True Value", "Error Bars"])
xlabel(r'$t\rightarrow$',size=20)
ylabel(r'$f(t)+noise\rightarrow$',size=20)
title("Q5: Data points for $\sigma = 0.10$ along with exact function", size=20)
grid(True)
show()
```

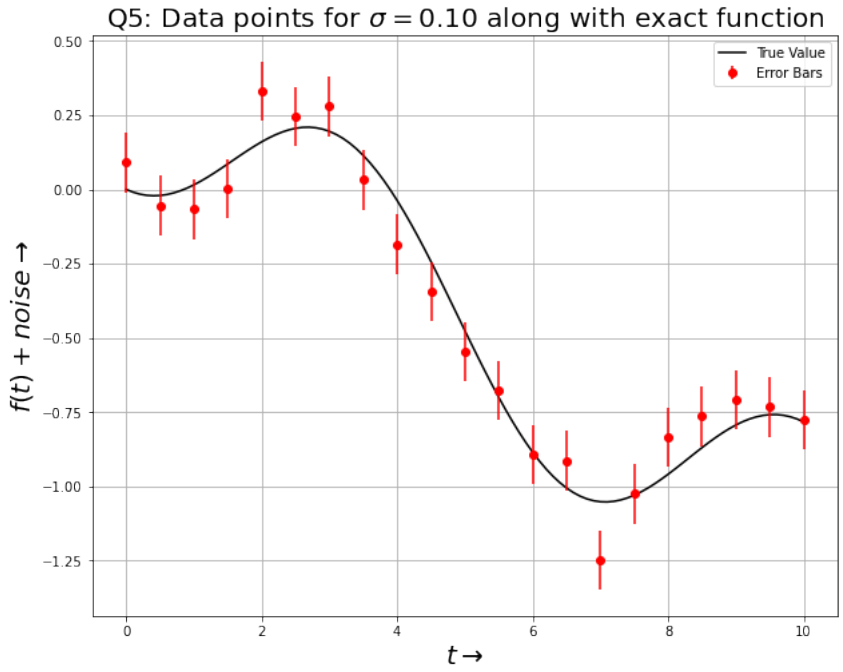

Figure 2: Question 5

3.6 Generation and Verification of M matrix

We generate the **M** matrix with accordance to the following equation:

$$g(t, A, B) = \begin{bmatrix} J_2(t_1) & t_1 \\ \dots & \dots \\ J_2(t_m) & t_m \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix}$$

We also show the $g(t, A_0, B_0)$ and $M \cdot \begin{bmatrix} A \\ B \end{bmatrix}$ are equal.

```
A, B = 1.05, -0.105 #Best Fit for no noise
Js = np.apply_along_axis(lambda x: sp.jn(2, x), axis = 0, arr = t)
M = c_[Js, t] #Getting M matrix
Gs = np.apply_along_axis(g, axis = 0, arr = t).reshape(-1,1)
#Checking if matrixes are equal
if np.allclose(np.matmul(M, np.array([A, B]).reshape(2,1)), Gs):
    print("The vectors are same")
else:
    assert False, "The vectors are not equal"
```

3.7 Calculation MSE for various combination of A and B

We calculate the MSE for every possible combination of A and B, where A and B are chosen from a set of 21 equally spaced numbers between 0 and 2

and between -0.2 and 0 respectively.

The error was calculated using the formula:

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

```
#Calculating MSE for 1st and 2nd column
A = np.linspace(0, 2, 21).tolist()
B = np.linspace(-0.2, 0, 21).tolist()
e = np.zeros((2,21,21))
print(t.shape)
for _ in range(2):
    for i in range(21):
        for j in range(21):
            Gs = np.apply_along_axis(lambda x: g(x,A[i],B[j]), axis = 0, arr = t)
            e[_][i][j] += 1/101*np.sum((yy[:,_] - Gs)**2, axis = 0)
```

3.8 Plotting Contour of Error

We plot the coutour using the contour funciton and we also used in inside the clavel function for labeling it.

```
x_a, y_a = linalg.lstsq(M, yy[:,0])[0] #Best fit for column 1
#Plptting
figure(figsize=(10,8))
clabel(contour(A,B,e[0], 10))
plot(x_a, y_a, marker = "o", markerfacecolor="red")
annotate("Exact Loaction", (x_a, y_a))
xlabel(r'$A\rightarrow$',size=20)
ylabel(r'$B\rightarrow$',size=20)
title("Q8: Contour plot of $\epsilon_{ij}$", size=20)
grid(True)
show()
```

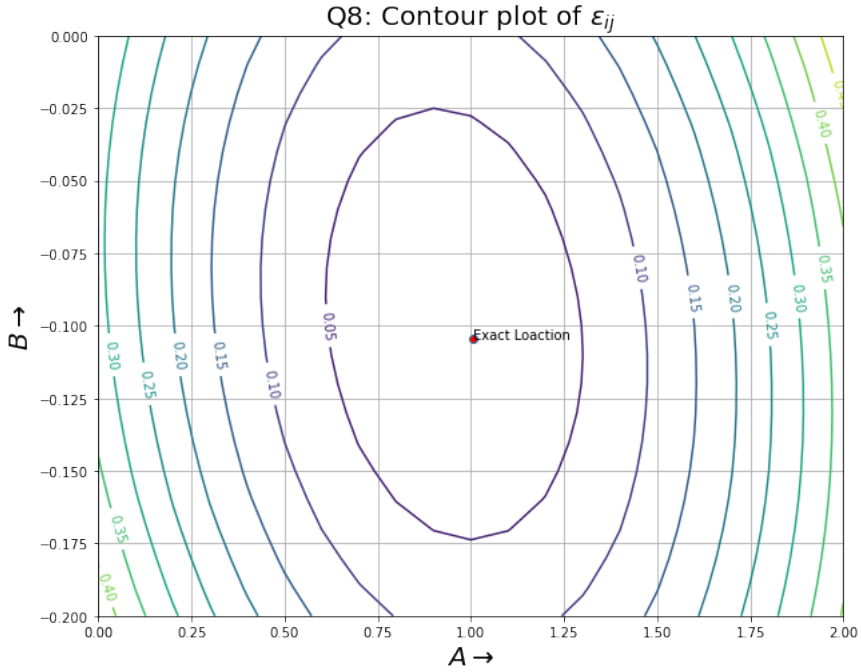

Figure 3: Question 8

3.9 Estimating A and B

A and B are chosed such that they minimize the following expression:

$$M \cdot \begin{bmatrix} A \\ B \end{bmatrix} - b$$

This is done using lstsq function in **scipy linalg module**

```
Gs = np.apply_along_axis(g, axis = 0, arr = t).reshape(-1,1)
GroundTruth = linalg.lstsq(M, Gs)[0]#No noise best fit
x = np.zeros((yy.shape[1], 2))
for i in range(yy.shape[1]):
    x[i] = linalg.lstsq(M, yy[:,i])[0] #Storing best fit for every column
print(f"The best estimate of A and B for 1st column is {x[0][0]:.4f} & {x[0][1]:.4f} respectively.")
```

3.10 Plotting Error in A and B vs σ in Linear scale

We use **plot** function for this and plot a slightly faded dashed line connecting the values.

```
err_a = np.zeros((yy.shape[1])) #Storing A errors
err_b = np.zeros((yy.shape[1])) #Storing B errors
#Euclidean distance
for i in range(yy.shape[1]):
    err_a[i] += ((x[i][0] - GroundTruth[0])**2)**0.5
    err_b[i] += ((x[i][1] - GroundTruth[1])**2)**0.5
#Plotting in linear scale
figure(figsize=(10,8))
plot(sigma,err_a, label = "Aerr", marker = "o", linestyle="dashed", alpha = 0.7)
plot(sigma, err_b, label = "Berr", marker = "o", linestyle="dashed", alpha = 0.7)
xlabel(r'$\sigma\rightarrow$',size=20)
```

```
ylabel(r'$Error(Euclidean)\rightarrow$',size=20)
legend()
title("Q10: Variation of error with noise", size=20)
grid(True)
show()
```

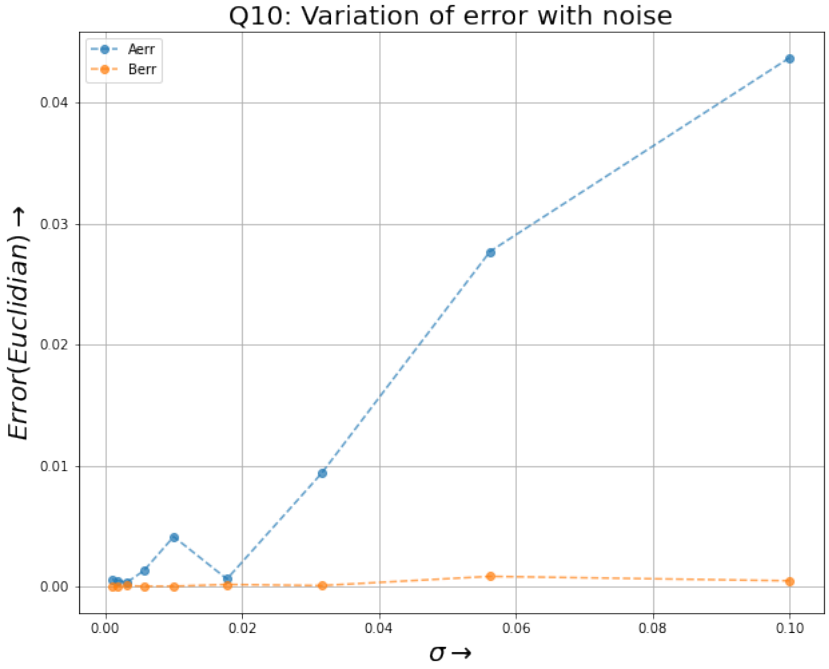

Figure 4: Question 10

3.11 Plotting Error in A and B vs σ in Log scale

We use **errorbar** function for this and plot error bars for the values

```
#Plotting in log scale
figure(figsize=(10,8))
errorbar(sigma, err_a, sigma,fmt="ro", label = "Aerr")
errorbar(sigma, err_b, sigma, fmt = "go", label = "Berr")
xlabel(r'$\sigma\rightarrow$',size=20)
ylabel(r'$Error(Euclidean)\rightarrow$',size=20)
yscale("log")
xscale("log")
legend()
title("Q11: Variation of error with noise", size=20)
grid(True)
show()
```

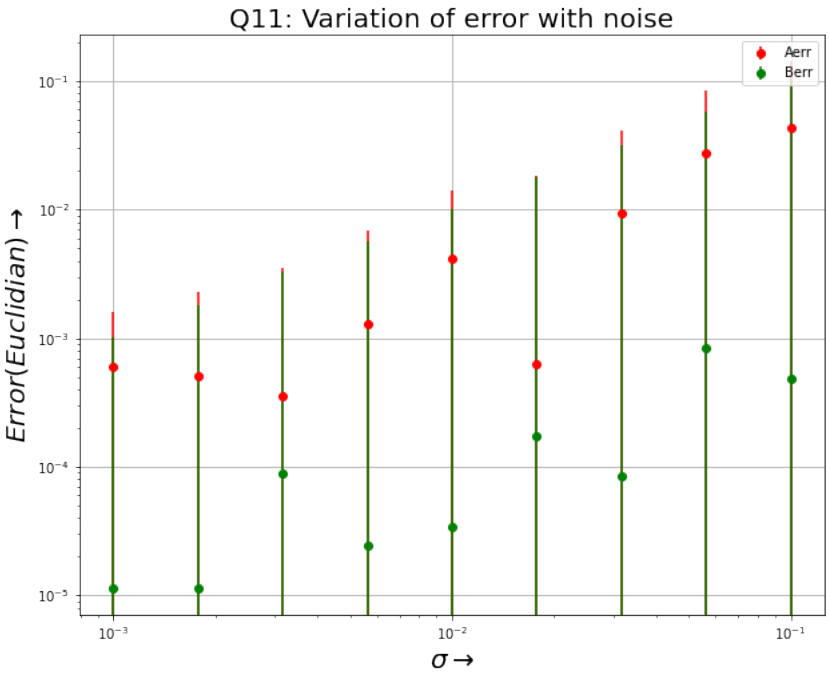

Figure 5: Question 11

4 Conclusion

From the plots we can see that the errors in estimated A and B increases with the standard deviation of the Gaussian noise in the data. We also see that the increase in error in log-log plot is somewhat linear.