Assignment 3: Fitting Data To Models

Shekhar Diwakar [EE20B123]

February 18, 2022

Abstract

This assignment is about fitting data to models.
The main content of the assignment is:

- Analysing data to extract information out of it.
- To study the effects of noise on the fitting process.
- To plot a number of different types of graphs.

1 Extracting the data

On running the python file *generate_ data. py*, a DAT file *fitting. dat* is created in the same directory in which the code was compiled. Also, the .py file generates a plot of the function with noises being included.

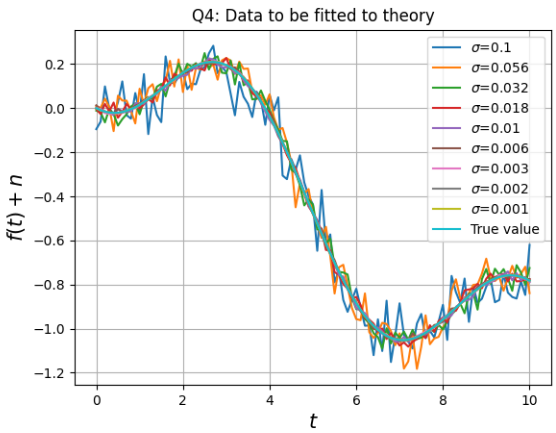

Figure 1: Data plot(Q-4)

This file(fitting.dat) contains 10 columns with 101 rows of data. The first column is the time values and then the next nine columns are the noisy values of the function as shown below. The standard deviation of each column of the data is given by the python command below:

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

2 Plotting the function

Since, the actual function is known, we can plot it's graph also. The function is defined in python *ee20b123_assignment_3.py* as the following code snippet:

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

On plotting the function's true value along with all the 9 noise added to the true function's values, the following plot was generated. This is the Figure 0 that was asked in Q-3 and Q-4. The python code snippet for plotting the follwing graph is as follows:

```
for i in range(col - 1):
    # Define legend ()
    plot(data[:,0],data[:,i+1],label='$\sigma$'+"="+str(np.around(deviation [i],3)))
    plt.legend()
```

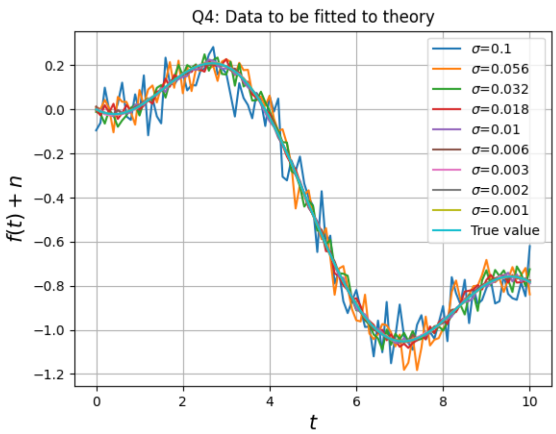

Figure 2: True and noise added plots

3 Error plot

Error bars is a good way of representing the uncertainty in the reported measurement. The errorbars for the first data column are plotted using the

errorbar() function. The python code snippet for plotting the errorbar plot is as follows:

```
plot(t, g(t, 1.05, -0.105), label = r"True value")
errorbar(t[::5], data[::5, 1], deviation[1], fmt='ro', label = r"Error bar")
```

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

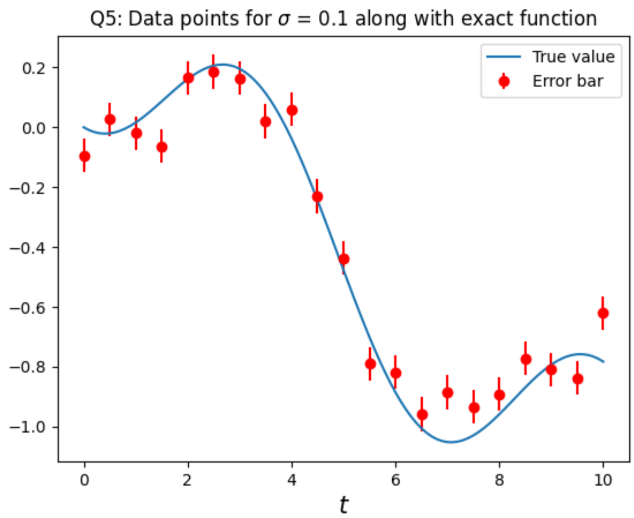

Figure 3: Errorbar plot

4 The Matrix equation

The true value function can be created using a matrix equation also. The matrix M when multiplied with (A,B) matrix will give rise to the actual function. In order to compare 2 matrices, we create a function `matrixequal(P, Q)` whose code snippet is given below:

```
def matrixequal(P, Q):
    count = 0

    for i in range(0, rn):
        if P[i] == Q[i]:
            count += 1

    if count == rn: # If all elements matched
        return True
    else: # Else, return false
```

```
        return False

if matrixequal(P, Q):
    print("Both the matrices(N = MP and fk) are equal.")
else:
    print("Both the matrices(Q = MP and fk) are not equal.")
```

5 Mean Squared Error

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

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

The python code snippet to calculate the mean squared error is as follows:

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

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

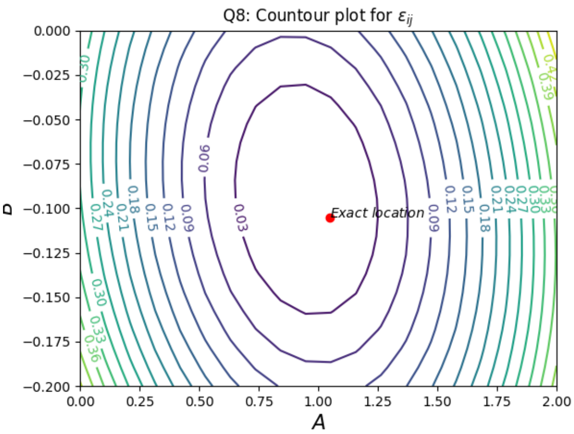

Figure 4: Contour plot

Conclusion

From the above plot, we can conclude that there exist one and only one minimum for ε.

6 Error calculations

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 = [] # Initialising the required variables
Aerror = []
Berror = []
y_true = g(t, 1.05, -0.105) # True graph
for i in range(col - 1):
    p, resid, rank, sig = lstsq(M, data[:, i + 1])
    aerr = np.square(p[0] - 1.05)
    ber = np.square(p[1] + 0.105)
    # Updating sthe errors
    Aerror.append(aerr)
    Berror.append(ber)
```

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

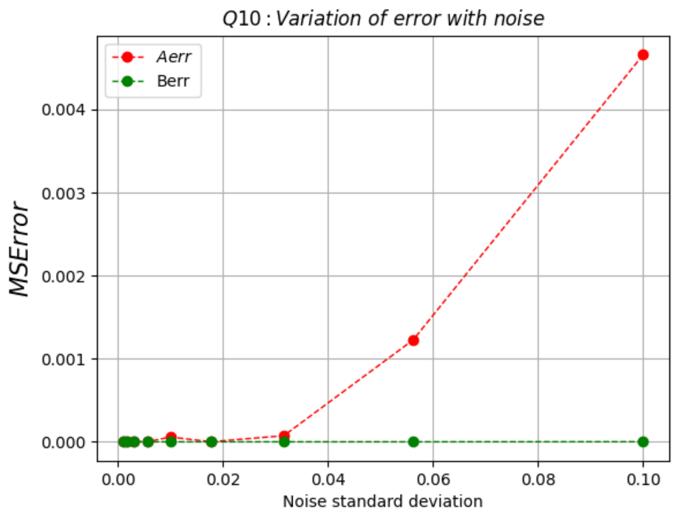

Figure 5: Error vs Standard deviation

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

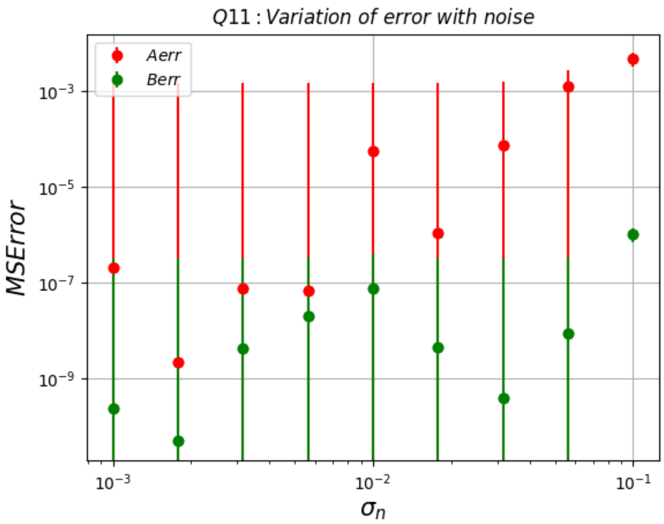

Figure 6: Error vs Standard deviation: log scale

Conclusion

From Figure 5, it is confirmed that the plot is not linear. Also from Figure 6, it is fixed that the plot is not linear in the logscale case too.

Inference

The given noisy data was extracted and the best possible estimate for the underlying model parameters were found by minimizing the mean squared error. This is one of the most general engineering use of a computer, modelling of real data.