EE2703 : Applied Programming Lab
Assignment 3

Potta Muni Asheesh
EE19B048

March 3, 2021

Aim

The aim of the assignment is to fit the given data into a given model using the **least squares method**.
The given model is as follows

$$g(t; A, B) = AJ_2(t) + Bt$$

Generating and visualing data

The data required is generated using the *generate_data.py* python script provide. The script, when run, creates a file named *fitting.dat* which contains 10 columns. The first column is time and next 9 columns are the model function evaluated at the corresponding time value to which some noise $n(t)$ is added. Each of the 9 columns have different noise with standard deviation σ varying from 10^{-3} to 10^{-1} uniformly spaced in log scale. The A and B values used are 1.05 and -0.105.
So, the data given in the columns corresponds to the function

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

The data loaded using the `loadtxt()` function available in `numpy`. This gives a 2D array with contents of the file. This array is transposed and stored as `data` to make the columns accessable easily.
Here is a plot showing the given data which contains noise along with the true curve.

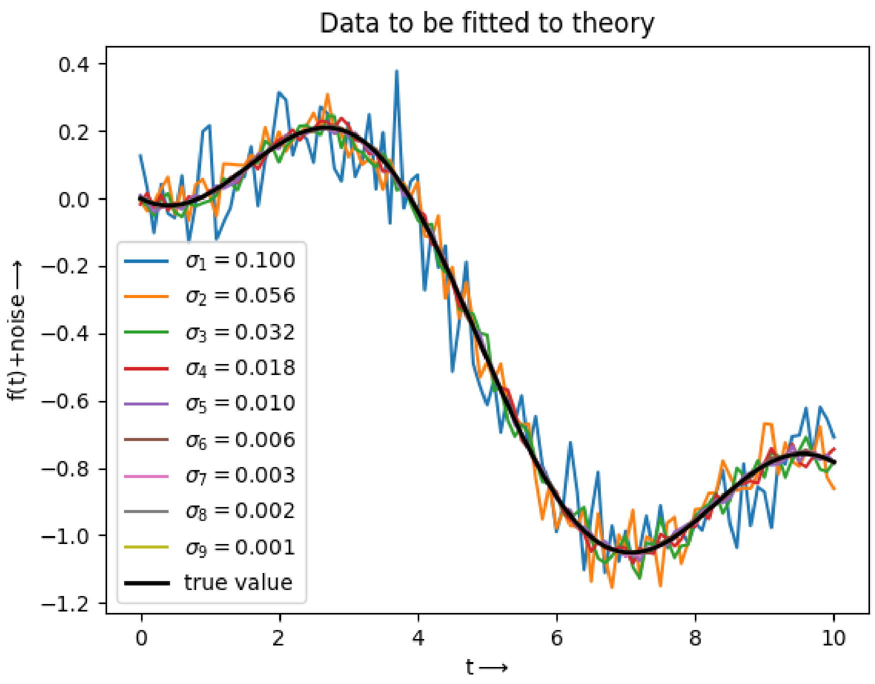

Figure 1: Plot of data from all columns

Visualing noise using errorbar

A convenient way of visualing nosie is to plot using errorbar. Python module `matplotlib.pyplot` provides a function `errorbar()` which can be used to do this. The second column data is used for this. The standard deviation for noise in this data is 0.1. This is how the plot looks like

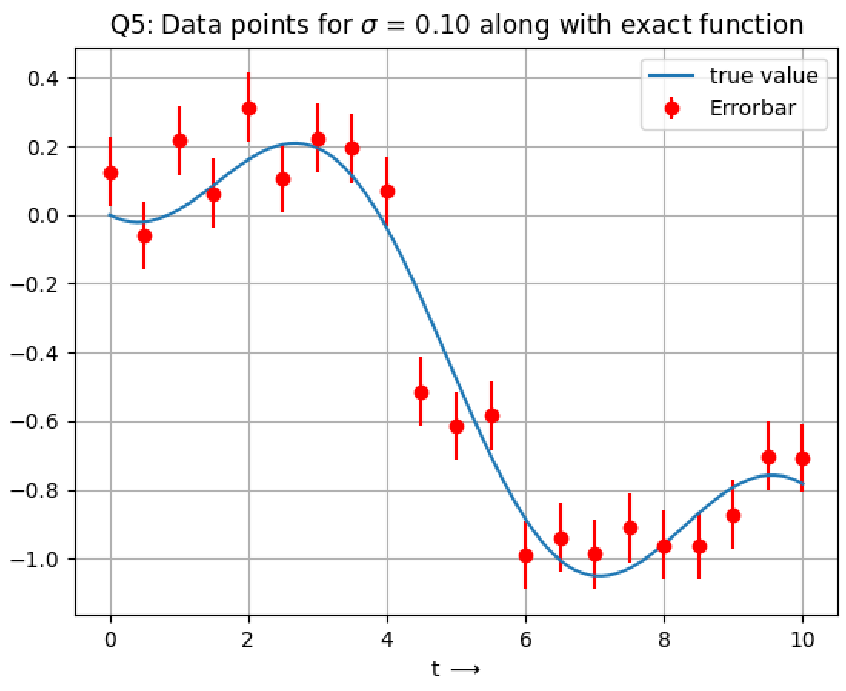

Figure 2: Errorbar plot

These data points are the most problematic in predicting A and B values as they deviate the most from the true value.

Deducing A and B values using least squares

Matrix representation

The model $g(t; A, B)$ can be represented as a matrix equation as

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

The dot product $M \cdot p$ and the function $g(t; A, B)$ for the same values of A and B. We can verify this plotting both the arrays on the same diagram.

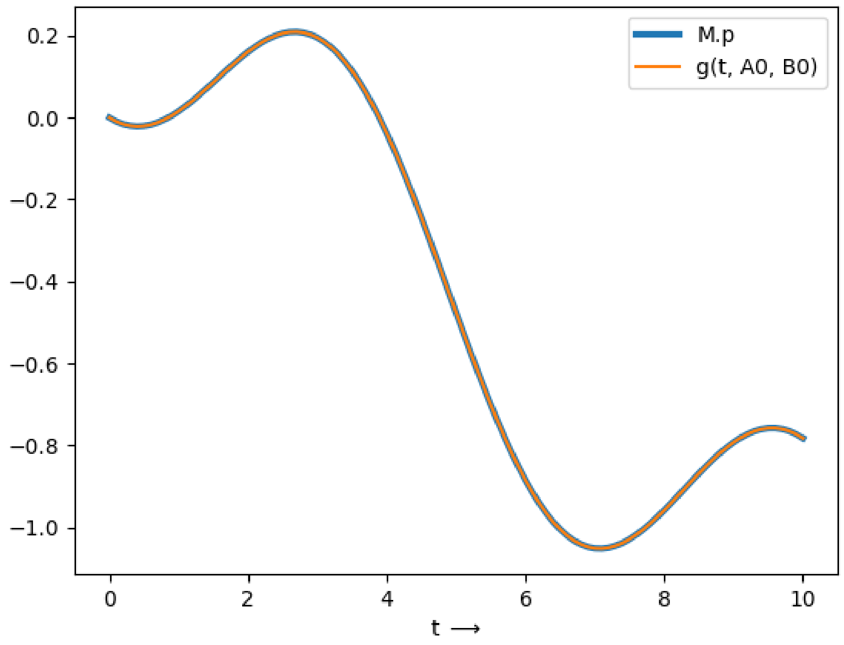

Figure 3: Verify the equivalence of $M \cdot p$ and $g(t; A, B)$

Mean squared error

For $A_i = 0, 0.1, .., 2$ and $B_j = -0.2, -0.19, .., 0$, we can use the data of columns 1 and 2 to compute the mean squared error ϵ_{ij} as

$$\epsilon_{ij} = \frac{1}{101} \sum_{k=0}^{100} (f(t_k) - g(t_k; A_i, B_j))$$

Here, ϵ_{ij} is the value of mean squared error for corresponding values of A_i and B_j. These values can be stored in a matrix. The matrix is created in python as follows

```
A = np.linspace(0, 2, 21)
B = np.linspace(-0.2, 0, 21)
mean_squared_error = np.zeros((A.size, B.size))
for i in range(A.size):
   for j in range(B.size):
       mean_squared_error[i,j]+=(1/101)*(np.sum(np.square(data
          ↪ [1]-g(time,A[i],B[j]))))
```

A contour plot of this matrix plotted using `contour()` function in `matplotlib.pyplot`. The A and B values arrays along with the matrix created are given as parameters to this function as shown below. The plot looks as shown below (See Figure 4)

```
f4, ax = plt.subplots()
CS = ax.contour(A, B, mean_squared_error, np.arange(0,1,0.025))
ax.clabel(CS, CS.levels[:5], inline=1, fontsize=10)
```

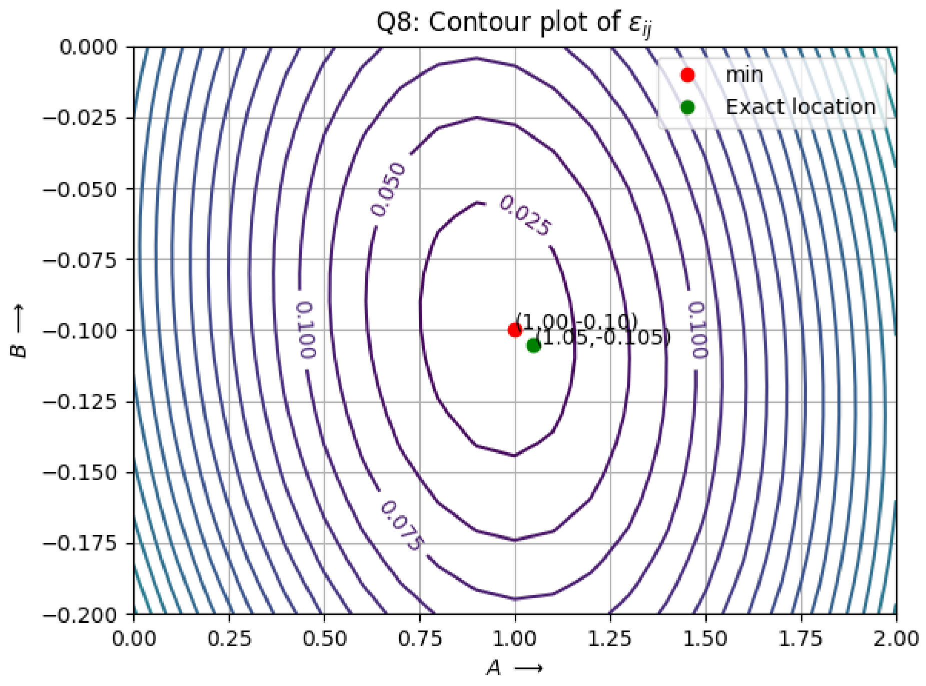

Figure 4: Contour plot

The index of minimum value in the matrix found using `numpy.argmin()` which finds the index of minimum for the flattened matrix and `numpy.unravel_index()` is used to find the index in original matrix. This can be done as follows

```
min_error_i, min_error_j = np.unravel_index(mean_squared_error.
    ↪ argmin(), mean_squared_error.shape)
```

There is only one minimum in the plot which is at $(1.00, -0.105)$ whereas the actual values of (A, B) are $(1.05, -0.105)$.

Least squares approximation

The matrix M created above and column data p can be passed to `lstsq()` function available in `scipy.linalg` to find the approximate values of A and B such that mean squared error is minimum. We can plot absolute difference of actual parameter value and approximated parameter value as Error against standard deviation of noise σ_n for both A and B.

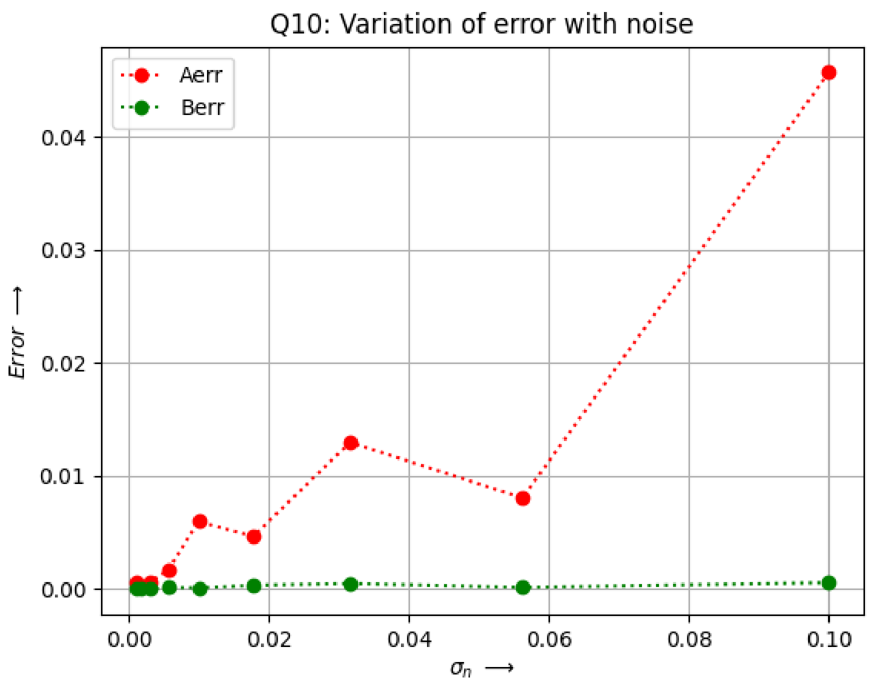

Figure 5: Error vs σ_n (linear scale)

The plot looks *non-linear* in nature. When we plot both x-axis and y-axis in log scale, the plot looks roughly *linear*. This tells us that error from actual value and σ_n of noise are directly proportional.

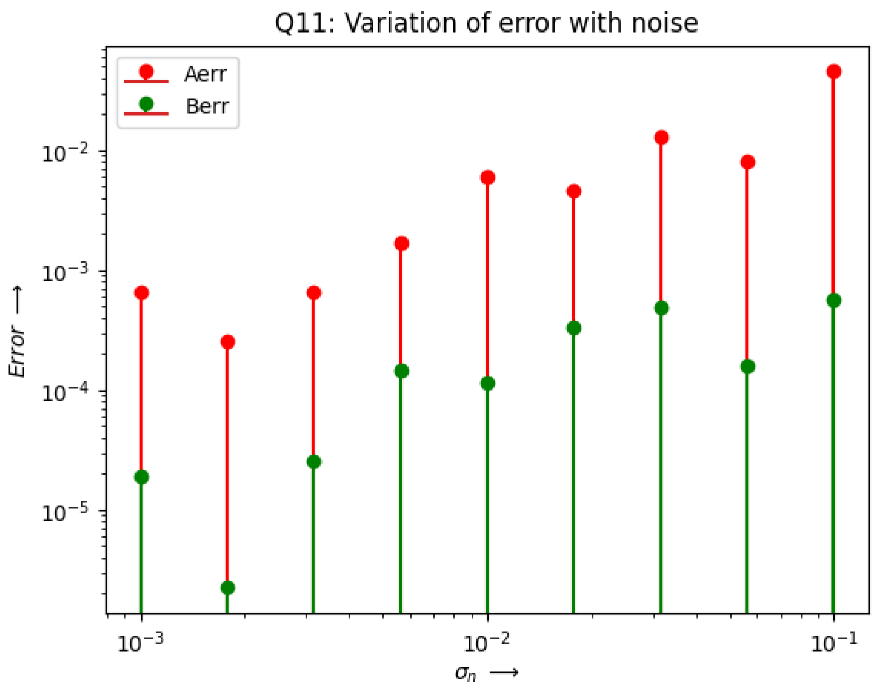

Figure 6: Error vs σ_n (log scale)

Conclusion

For a given noisy data and model with parameters, we can approximate the parameters such that the mean squared error from actual model data is minimum using **least squares approximation method**. It was found that the standard deviation of noise is linearly related to error in parameter approximation in log scale.