Assignment No 3

Sumanth R Hegde, EE17B032

February 12, 2019

Extracting and Visualizing the data

The given data was obtained by running the *generate_data.py* script. The data contained 10 columns : the first column was time, and the other 9 columns were each riddled with different amounts of noise, with standard deviation uniformly sampled from a logarithmic scale. On plotting all the 9 columns, the following graph was observed :

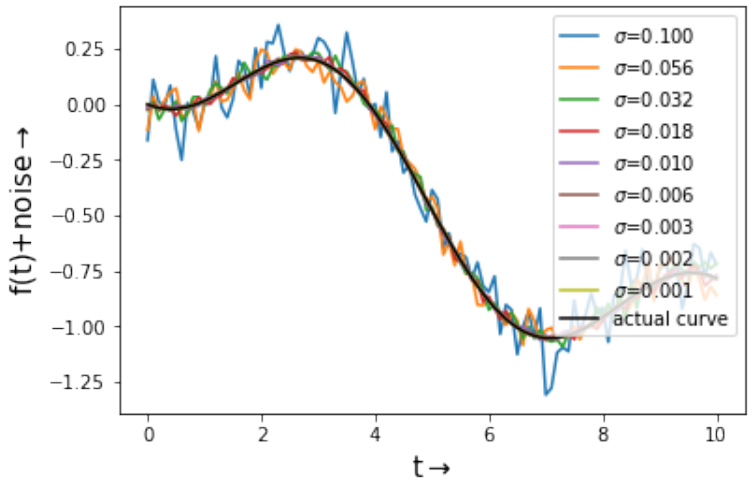

Figure 1: Plot of all data columns

The Function

Since we are aware of the actual function to be extracted from the data columns, the general shape of the function to fit the given data is known. A simple implementation does the required computation :

```
def func(t,A,B):
    return A*sp.jn(2,t) + B*t
```

Visualising noise - The Errorbar plot

An errorbar is a convenient way of visualising the uncertainty in the reported measurement. The errorbars for the first data column are plotted using the `errorbar()` function. The graph obtained by plotting every 5th data point with errorbars and the original data is as follows:

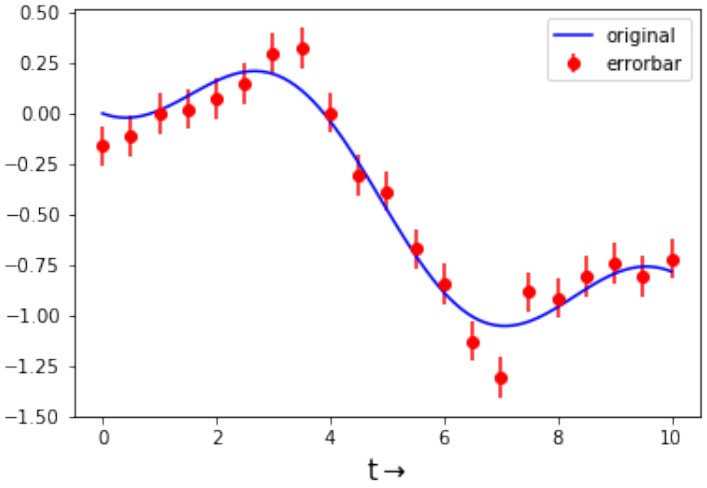

Figure 2: The Errorbar plot

Each point in the data column can be seen to be varying mostly within a σ width of the true value, except in a few cases. These are the points that most affect the prediction of the function parameters A and B .

The matrix equation

We can compare the results obtained using matrix multiplication and the user-defined function using the np.array_equal() function. If this returns a *True* value, then they are equivalent.

Error computation

The following code snippet is used to compute the error e in each data column:

```
for k in range(9):
    f = y_columns[:,k]
    for i in range(21):
        for j in range(21):
            e[i][j][k] = np.sum((f -np.array(func(time,A[i],B[j])))**2)/101
```

`y_columns` represents the data columns. The mean squared error for the first data column can now be plotted and the minima can be found to obtain the best estimate for A and B. The following code block does the needful:

```
pylab.contour(A,B,e[:,:,0])
a = np.unravel_index(np.argmin(e[:,:,0]),e[:,:,0].shape)
pylab.plot(A[a[0]],B[a[1]],'o',markersize=3)
```

The `np.argmin()` function returns the index of minima for the flattened array, and thus the `np.unravel_index()` function is used to get the location of the minimum in the original array.

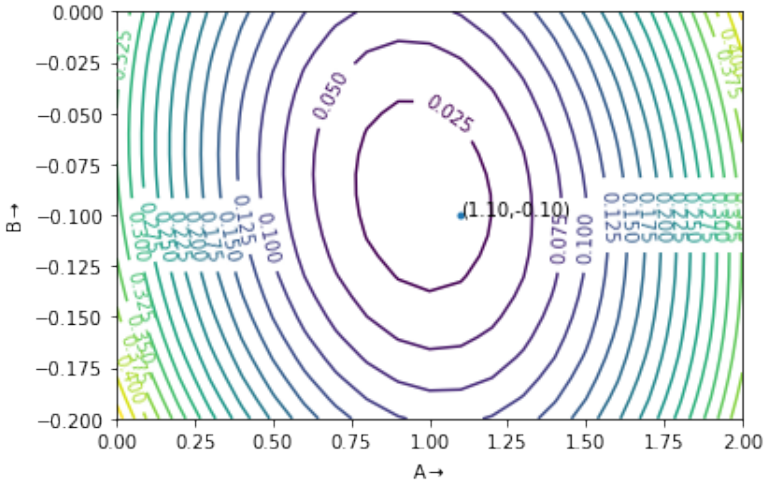

Figure 3: Contour plot

From the graph, the error function has one and only one minimum and it occurs at $A = 1.10$ and $B = -0.10$.

Parameter Estimation

The variation in error in estimates for A and B is as follows :

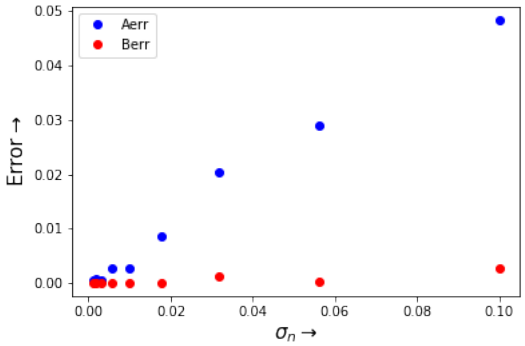

Figure 4: Error vs σ

The error estimate is non-linear with respect to the noise.

On plotting both the axes in the *log* scale, the graph becomes approximately linear, as the σ values are equally spaced in the *log* scale

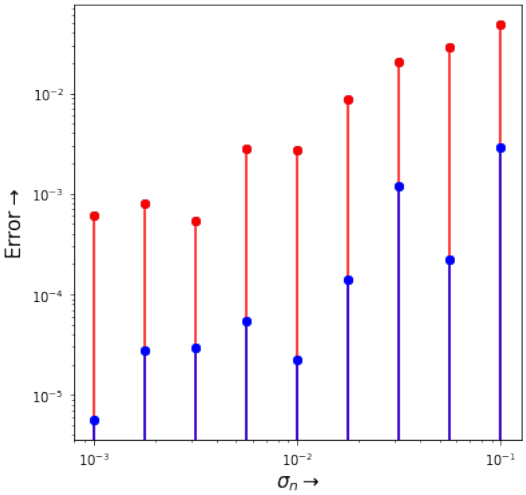

Figure 5: Error vs σ : The log scale

Conclusion

The given noisy data was extracted and the best possible estimate for the underlying model parameters were found by minimizing the mean squared error. It was observed that the error was approximately linear in σ_n in the *log* scale.