Assignment 3

Sagnik Ghosh, EE19B132

March 2021

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

This assignment is an attempt at getting acquainted with modelling of real data, which is an integral part of any engineering discipline. We start with generating data for a linear combination of the **Bessel Function** and the function $f(t) = t$ with varying amounts of Gaussian noise. This is done using a Python script provided to us.

$$A * J_2(t) + B * t$$

where $A = 1.05$ and $B = -0.105$ are the parameters to be estimated.

The packages used to achieve the above task are as follows:

- Numpy
- Matplotlib
- SciPy
- PyLab

The most important function used in this assignment is **scipy.linalg.lstsq()**.

3 Assignment

3.1 Plotting and Visualising the Data to be Fitted

The data to be fitted was generated using the *generate.py* script provided to us. This generated a file *fitting.dat* which contained 10 columns. The first column contained the values of the independent variable and the remaining columns contained the data to be fitted corrupted, using varying degrees of noise. The plot of all the noise corrupted data and the true function was generated using the following code:

```
legends = []
for i in range(len(data)):
    plt.plot(time, data[i])
    legends.append('Noise Std Dev = %1.3f'%(sigma[i]))
plt.plot(time, orig_func, color = 'black')
legends.append('Original Function')
plt.xlabel(r'$t$', size = 15)
plt.ylabel(r'$f(t)+n$', size = 15)
plt.title(r'Plot of Data to be Fitted')
plt.legend(legends)
plt.show()
```

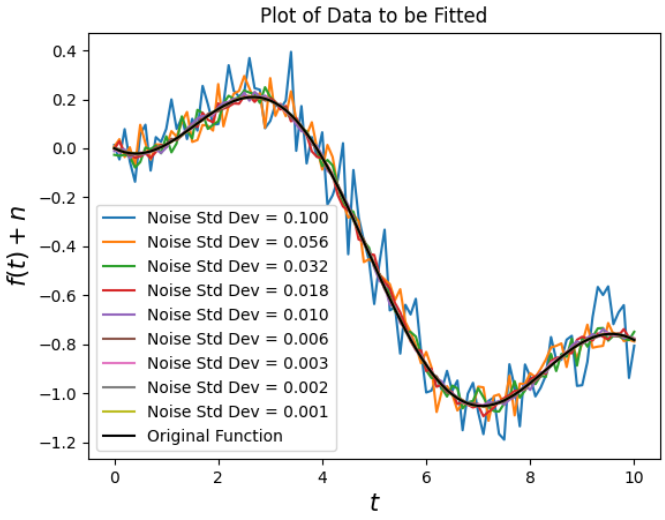

Figure 1: Plot of all data with varying noise and the true function

3.2 Plot of Error Bars

```
plt.errorbar(time[::5],data[0][::5],sigma[0],fmt='ro')
legends = []
legends.append('Original Function')
plt.plot(time, orig_func)
legends.append('Error Bars')
plt.legend(legends)
plt.xlabel(r'$t$')
plt.title(r'Error Bar and Original Function')
plt.show()
```

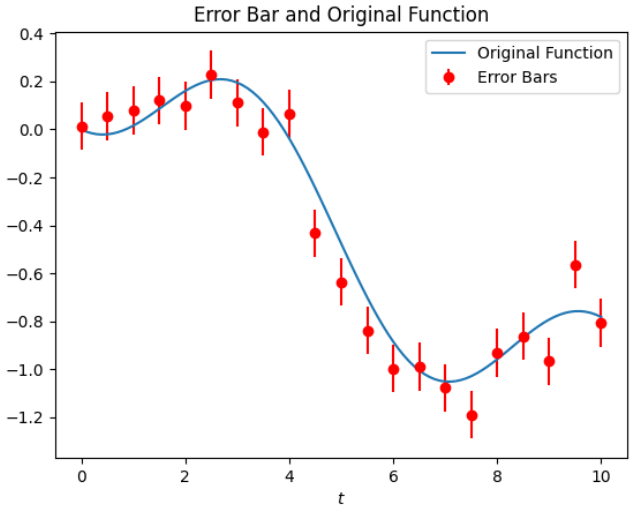

Figure 2: Error Bars

3.3 Constructing the Matrix

We need to construct the matrix M,

$$\begin{pmatrix} \mathrm{J}_2(t_1) & \mathrm{t}_1 \\ \mathrm{J}_2(t_2) & \mathrm{t}_2 \\ \ldots & \ldots \\ \mathrm{J}_2(t_m) & \mathrm{t}_m \end{pmatrix}$$

This is done using the following code:

```
def generate_matrix(time):
    x = sp.jn(2, time).T
    y = time.T
    M = np.c_[x,y]
    return M
```

3.4 Calculating Error for Different Values of A and B and Plotting a Contour Plot

The error is computed using the following equation:

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

Here, f_k is some column of the data.

```
A = np.arange(0, 2.1, 0.1)
B = np.arange(-0.2, 0.01, 0.01)
(x1, y1) = np.meshgrid(A, B)
mse_error = np.zeros((len(A), len(B)))

for i in range(len(A)):
    for k in range(len(B)):
        mse_error[i][k]+=
        (np.matmul((data[0].T - np.array([g(t, A[i], B[k]) for t in time]).T).T,
        data[0].T - np.array([g(t, A[i], B[k]) for t in time]).T))/101
```

The Contour is plotted using the following code:

```
fig, ax = plt.subplots()
CS = ax.contour(x1, y1, mse_error, levels =10)
ax.clabel(CS)
ax.scatter([1.05], [-0.105])
ax.set_xlabel(r'Parameter A')
ax.set_ylabel(r'Parameter B')
ax.set_title(r'Contour Plots of MSE Error')
plt.show()
```

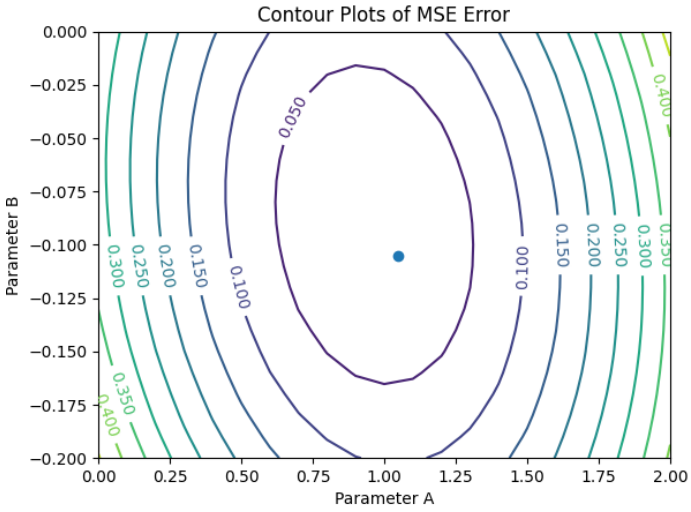

Figure 3: Contour Plot of MSE Error

3.5 Estimating A and B

Estimating the parameters A and B is done using the method of **Least Squares**. The Least Squares function is builtin to Python through *scipy.linalg.lstsq()*. Estimates for A and B using datasets of all noise levels are found using the following code:

```
for i in range(len(data)):
    estimated_parameters.append(scipy.linalg.lstsq(generate_matrix(time),
    data[i].T)[0])
```

3.6 Plot of Variation of Error with Sigma

The MS Error was calculated using the following code:

```
mse_A = (estimated_A - 1.05)**2
mse_B = (estimated_B + 0.105)**2
```

The plot of error in A and B on a linear scale was plotted using the following code:

```
plt.plot(sigma, mse_A, marker = 'o', markersize = 5, linestyle = 'dashed')
plt.plot(sigma, mse_B, marker = 'o', markersize = 5, linestyle = 'dashed')
plt.legend(['Error in A', 'Error in B'])
plt.title(r'Errors in A and B in linear scale')
plt.xlabel(r'Standard Deviation')
```

```
plt.ylabel(r'MS Error')
plt.show()
```

The plot of error in A and B on a LogLog scale was plotted using the following code:

```
plt.stem(sigma, mse_A, use_line_collection=True)
plt.stem(sigma, mse_B, 'r','ro', use_line_collection=True)
plt.legend(['Error in A', 'Error in B'])
plt.title(r'Errors in A and B in log scale')
plt.xlabel(r'Standard Deviation')
plt.ylabel(r'MS Error')
plt.xscale('log')
plt.yscale('log')
plt.show()
```

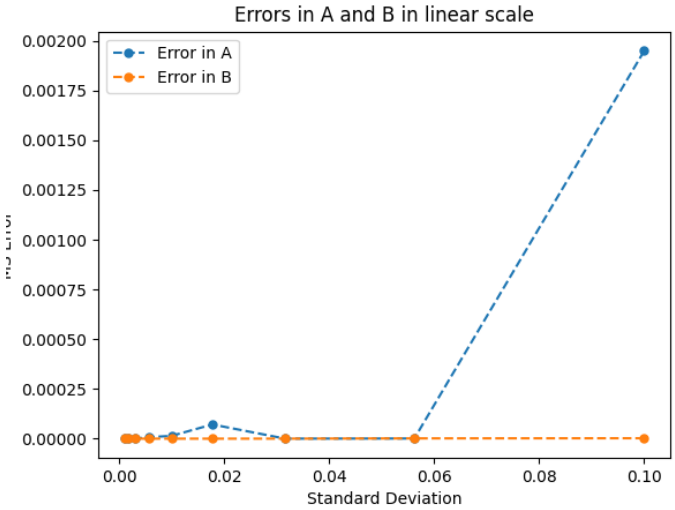

Figure 4: Plot of Error vs Sigma on a Linear Scale

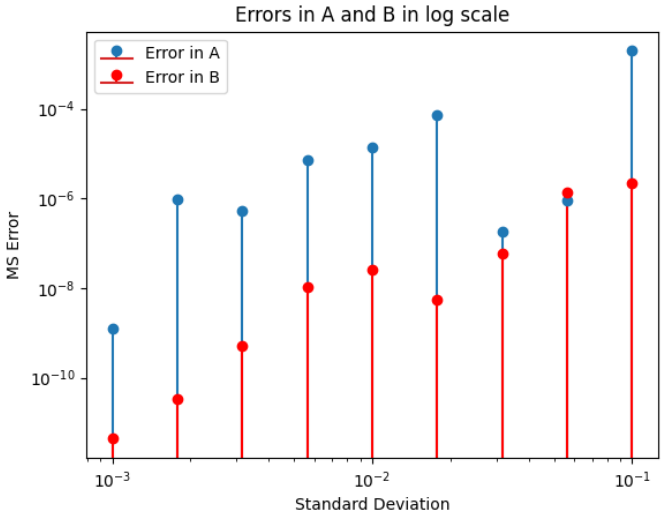

Figure 5: Plot of Error vs Sigma on a LogLog Scale

4 Result

In this assignment, we have estimated the parameters of a linear combination of the Bessel Function and $f(t) = t$ using the SciPy Library and also plotted various plots using the Matplotlib Library. The error plots of A and B were nearly linear because the initial standard deviation of the noise was chosen to vary linearly on a log scale.