Assignment 3: Fitting Data to Models

Siddharth D P
EE18B072

February 11, 2020

1 Introduction

The goal of the this assignment is to do the following:

- Analyse the data read from a file called **fitting.dat** file.
- Study the effect of noise on the given data.
- Plot graphs corresponding to various error measurements.

2 Assignment

2.1 Parts 1 and 2

Importing the standard libraries

```
from pylab import *
import scipy.special as sp
```

Defining the function

```
def g(t,A,B):
    y=A*sp.jn(2,t)+B*t # y=f(t) and t are vectors
    return(y)
```

The data, already generated, is loaded and parsed. The true value is calculated for reference.

```
a = loadtxt("fitting.dat")
time = a[:,0]
values = a[:,1:]
true_fn = g(time,1.05,-0.105)
values = c_[values, true_fn]
```

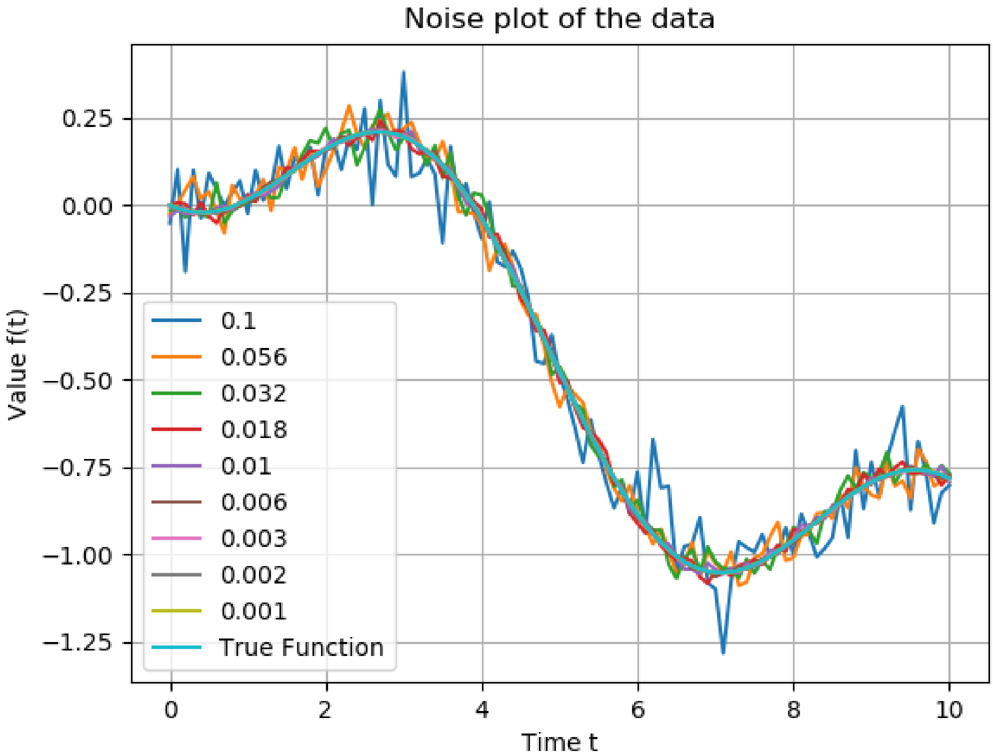

Figure 1: f(t) versus time t

2.2 Parts 3 and 4

Creating evenly spaced logarithmic values, we plot the function with various noise amounts. Having defined the original function, we then plot the graph with the true value as shown in Figure 1.

```
figure('Figure 1 - Noise')
scl=logspace(-1,-3,9) # noise stdev
for i in range(0,len(scl)):
scl[i]=round(scl[i],3)
plot(time,values)
title(r'Noise plot of the data')
xlabel('Time t'); ylabel('Value f(t)')
legend(list(scl)+["True Function"])
grid(True)
show()
```

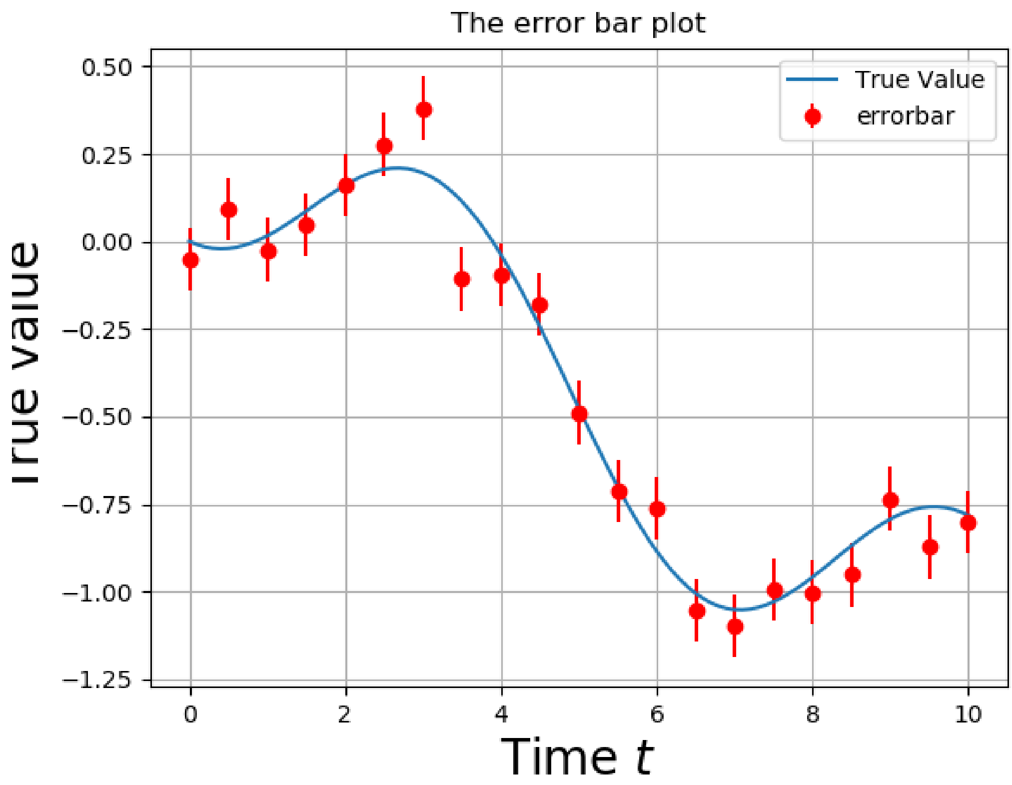

Figure 2: True value vs time t with error bars

2.3 Part 5

Generating the error bars withe given command, we plot the error-bar graph using the first column of data values as shown in Figure 2.

```
figure('Figure 2 - Error bar plot')
plot(time,true_fn,label='True Value')
title('The error bar plot')
std = std(values[:,0]-true_fn)
errorbar(time[::5],values[:,0][::5],std,fmt='ro',label='errorbar')
xlabel(r"Time $t$",size =20)
ylabel(r"True value",size =20)
legend(loc='upper right')
grid(True)
show()
```

2.4 Parts 6 and 7

A function is defined to calculate the least square error.g(t,A,B) as a column vector is obtained by creating a matrix equation.

```
def g_matrix(t,A=1.05,B=-0.105):
    J_val = sp.jn(2,t)
    t_val = t
    M = c_[J_val,t_val]
    x = array([A,B])
    return(matmul(M,x),M)

def mse(data,assumed_model):
    return(sum(square(data-assumed_model))/101)

j = [sp.jn(2,i) for i in time]
M = c_[j,time]
G = np.matmul(c_[[sp.jn(2,i) for  i in time],time],np.array([A0,B0]))
print ('Yes they are equal') if max([G[i]-true_fn[i] for i in range(0,len(G))])
<1e-10 else print('They are not equal')
```

2.5 Part 8

This graph involves plotting the contours of the error, considering the function g(t;A,B) to be the best fit to the given data considering A, B in the given range - A=[0.,0.01,...0.19.0.2], B=[-0.2,-0.19,...,-0.01,0]. The plot is shown in Figure 3.

```
A_range = arange(0,2.1,0.1)
B_range = arange(-0.2,0.01,0.01)

title("Contour Plot")
e_matrix = zeros((len(A_range),len(B_range)))
for A in enumerate(A_range):
    for B in enumerate(B_range):
        e_matrix[A[0]][B[0]] = mse(values[:,0],g_matrix(time,A[1],B[1])[0])
contour_obj = contour(A_range,B_range,e_matrix)
clabel(contour_obj,contour_obj.levels[0:5])
plot(1.05, -0.105,'ro', label = 'Exact Value')
legend(); xlabel('A -->'); ylabel('B -->')
```

2.6 Parts 9 and 10

We now plot the variation of error of each **A** and **B** with noise.The plot is non-linear in nature as can be seen from Figure 4.

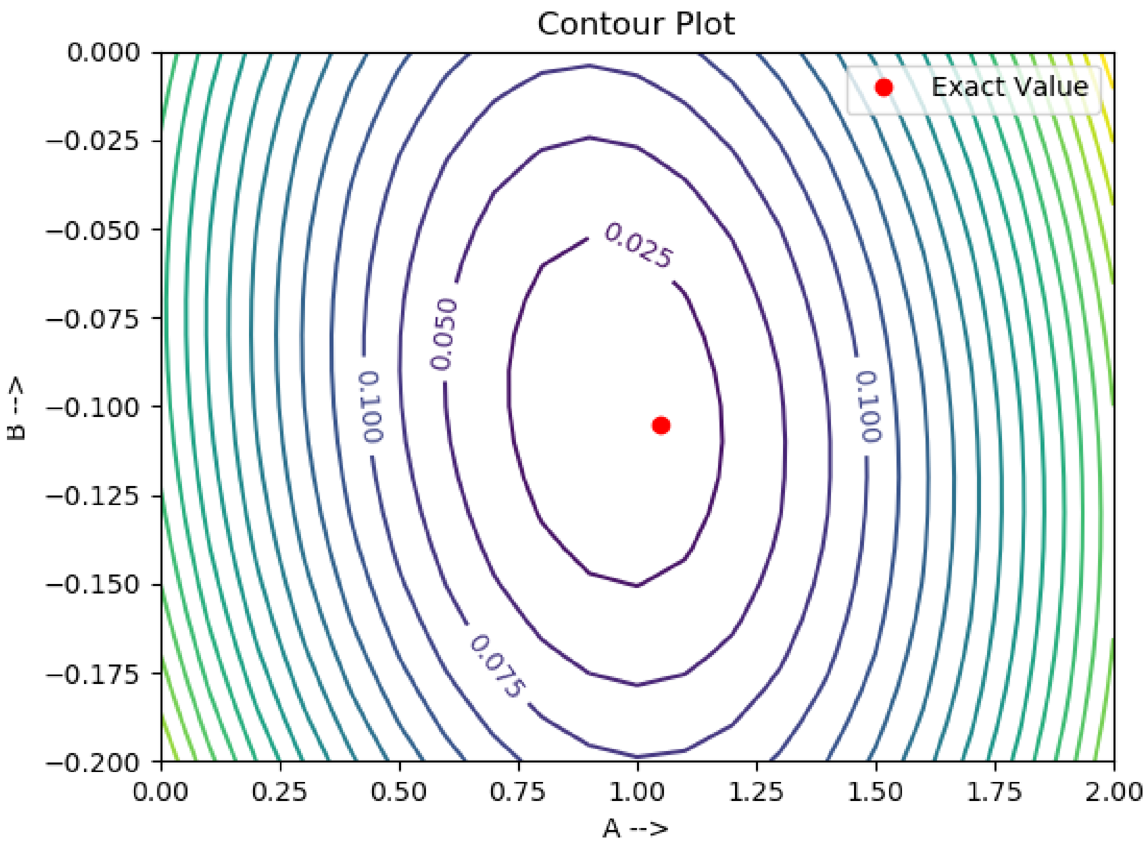

Figure 3: The Contour plot

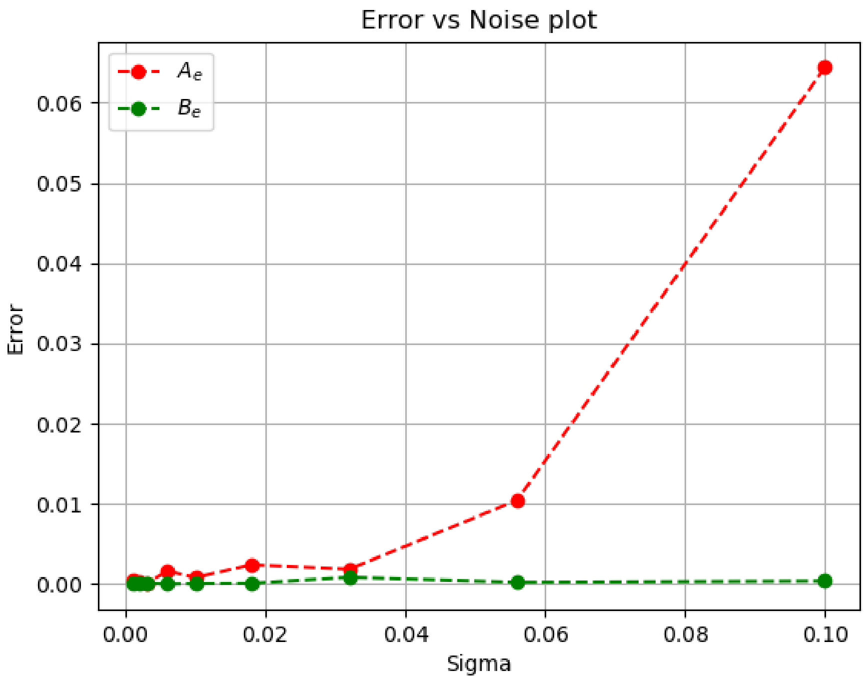

Figure 4: Error vs Noise

```
l_mse_A = zeros((9))
l_mse_B = zeros((9))
l_mse_error = zeros((9))
for i in range(9):
    temp = linalg.lstsq(g_matrix(time)[1],values[:,i],rcond=None)
    l_mse_error[i] = temp[1][0]
    l_mse_A[i],l_mse_B[i] = temp[0]
figure('Error vs Noise')
plot(scl,absolute(l_mse_A-1.05),'ro', label='$A_e$', linestyle='--')
plot(scl,absolute(l_mse_B+0.105),'go', label='$B_e$', linestyle='--')
title("Error vs Noise plot")
legend();grid(True);xlabel('Sigma');ylabel('Error')
```

2.7 Part 11

Using the *loglog* function to plot on a logarithmic scale, we get the following graph.

```
figure('Figure 5 - The log log plot')
title("Log Error vs Log Noise")
```

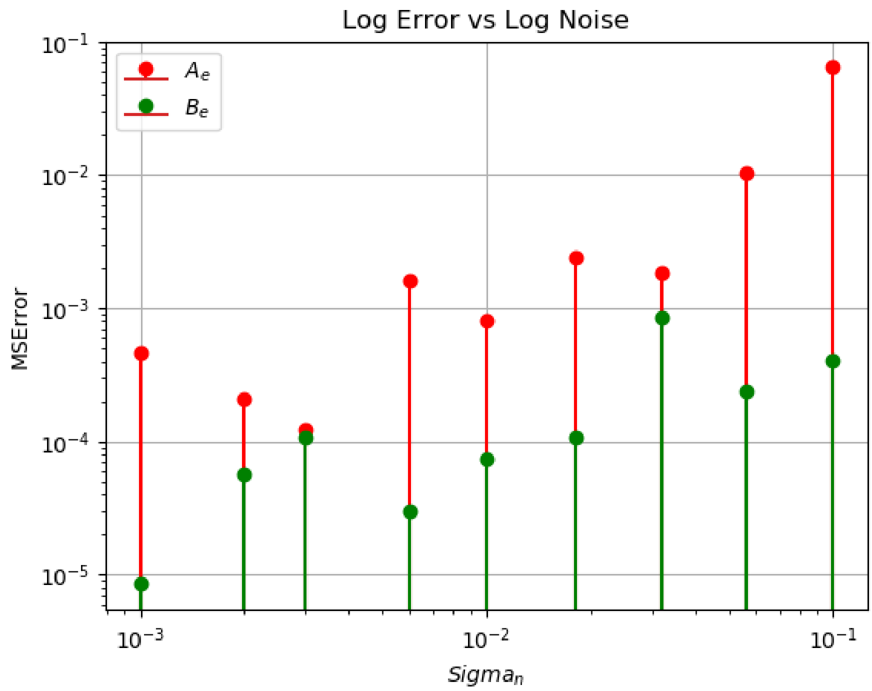

Figure 5: The log log plot

```
stem(scl,absolute(l_mse_A-1.05),'r','ro', label='$A_e$')
stem(scl,absolute(l_mse_B+0.105),'g','go', label='$B_e$')
legend();xscale('log');yscale('log');ylabel('MSError');xlabel('$Sigma_n$');grid
(True)
show()
```

3 Conclusion

- Least Squares Fitting can be used to find a best fit for an overdetermined system of equations.
- *log(error)* does not seem to be linear with *log(noise)*.