Assignment 3

Aditya - EE17B047

February 13, 2019

1 Introduction

This assignment primarily deals with the nuances of plotting various graphs in Python language. This is done mainly by employing the matplotlib.pyplot. The assignment involves collecting data from a **fitting.dat** file. The data is composed of the time in the first column and the rest of the columns represent a function f(t) added with some noise in each case.

Now, we must fit a predefined function g(t;A,B) with arbitrary A,B to the given function f(t) using the lstsq function in Python.

$$g(t; A, B) = AJ_2(t) + Bt \tag{1}$$

Then, we must plot the required graphs accordingly. The function is:

$$f(t) = 1.05J_2(t) - 0.105t + n(t) \tag{2}$$

Where, $J_n(t)$ is the Bessel function and $n(t)$ is the noise which has been added to the signal. The nature of the noise added is gaussian distribution.

$$Pr(n(t)|\sigma) = 1/(\sigma\sqrt{2\pi})exp(-n(t)^2/2\sigma^2) \tag{3}$$

2 Graphs

2.1 Graph1

This plot is composed of all the 9 functions composed of noise as well as the true function. The legend is composed of the standard deviation values of each of the plots. The corresponding graph is Figure 1 in page 2. The code corresponding to it is :

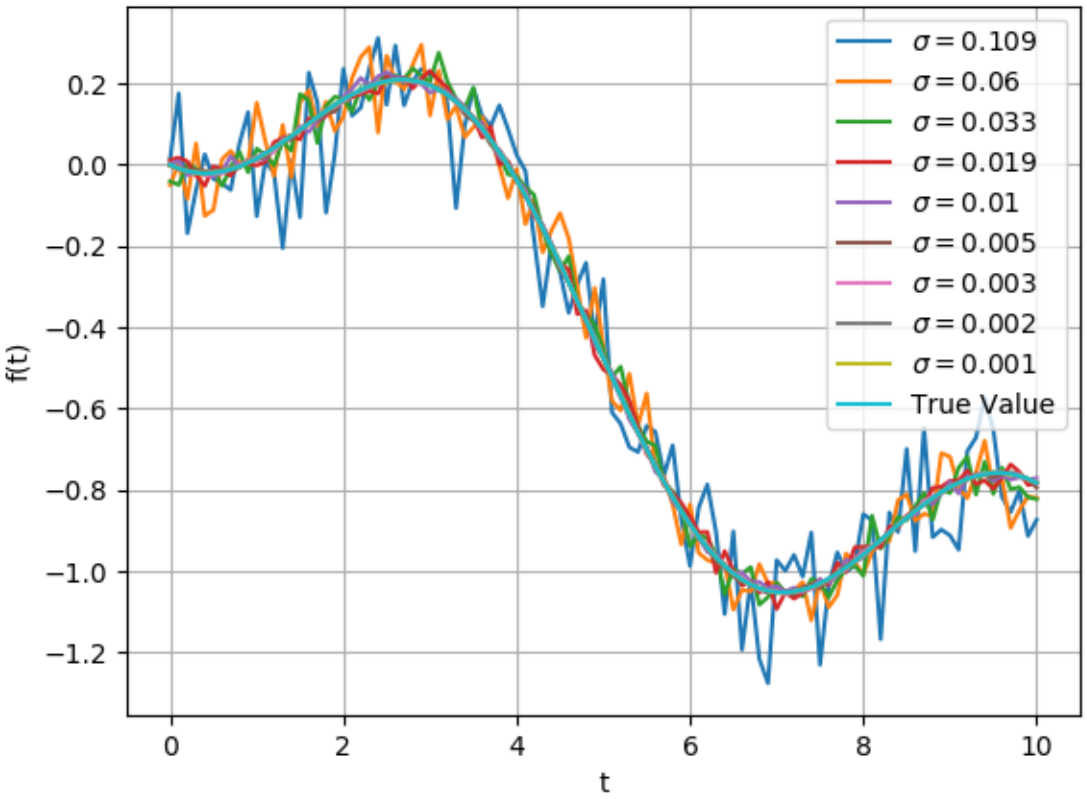

Figure 1: Graph composed of all the functions to which randomised noise has been added

2.2 Graph2

This plot involves printing the errorbar of the data in the column 2. This can be done using the errorbar() function. The corresponding graph is Figure 2 as shown below.

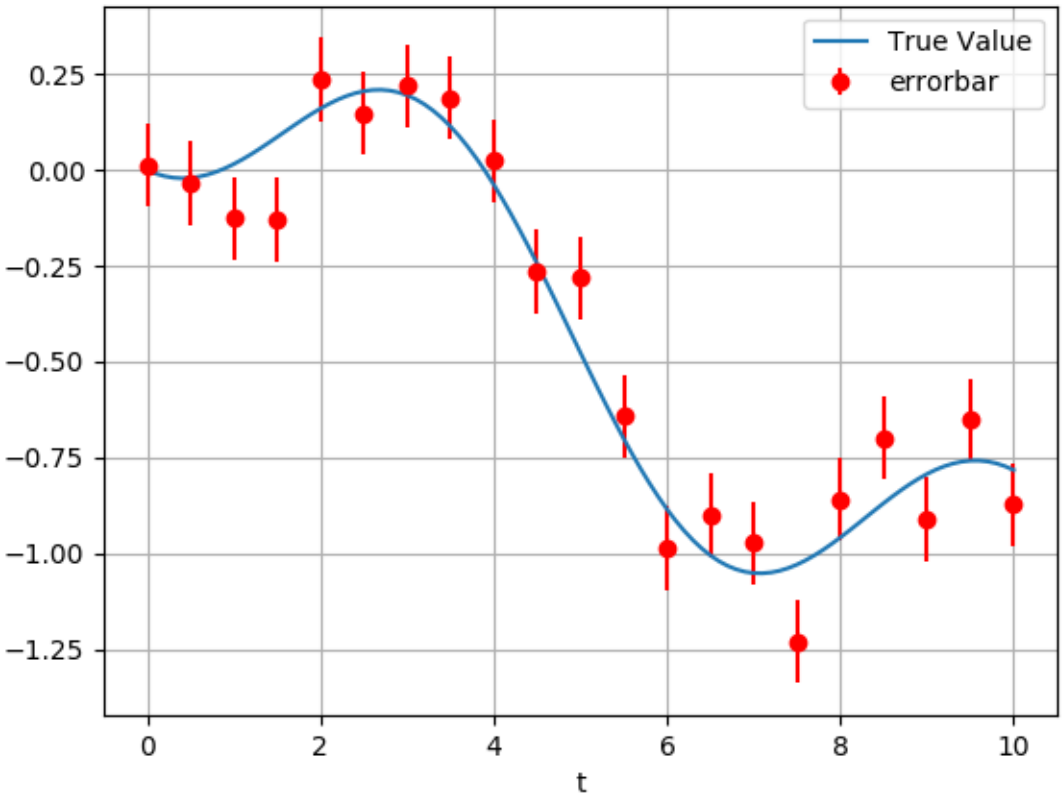

Figure 2: Errorbars for data added with noise

2.3 Graph3

This graph involves plotting the contours of the error in considering the function g(t;A,B) to be the best fit to the given data considering A,B in a given range.

$$A=[0.,0.01,...0.19.0.2],B=[-0.2,-0.19,...,-0.01,0]$$

The corresponding plot is as shown below Figure 3

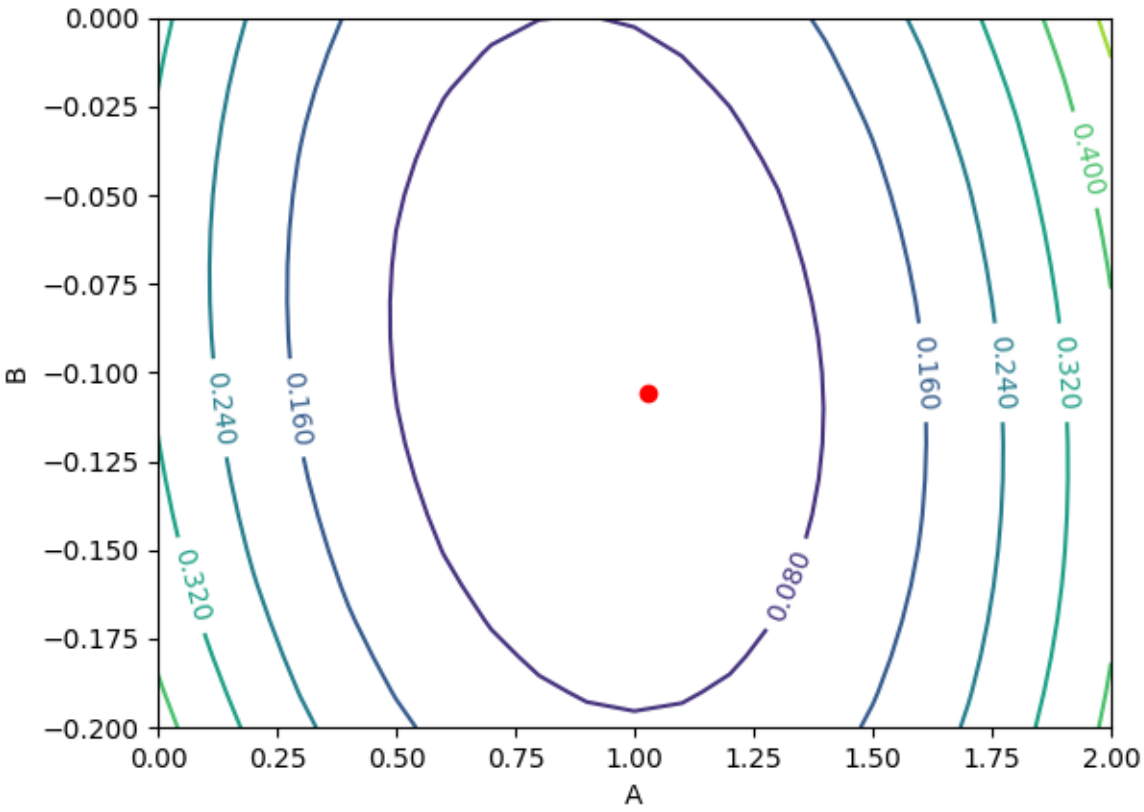

Figure 3: Contour Plot of Error estimates for various A and B

2.4 Graph4

This plot illustrates the effect of variance on the error in the value of coefficients A,B while fitting the data. The corresponding graph is as shown below Figure 4. The plot obtained is non linear in nature.

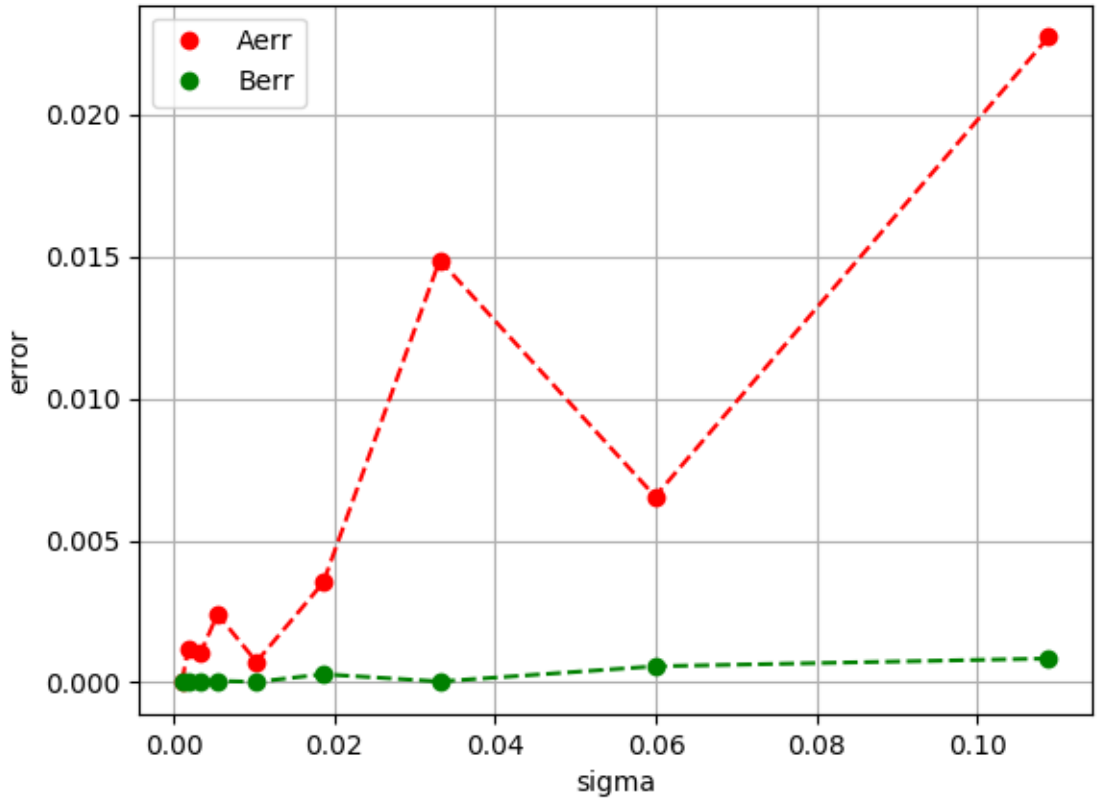

Figure 4: Standard Deviation vs Error in A and B

2.5 Graph5

This plot comprises of the exact same one as in Graph4 but, the scale of the X and Y axis has been changed to be logarithmic. The corresponding plot is as shown below Figure 5. The functions that have been used is **stem**.

The plot is non linear in nature. This implies that there is no direct relation between the error in A,B and the standard deviation corresponding to it.

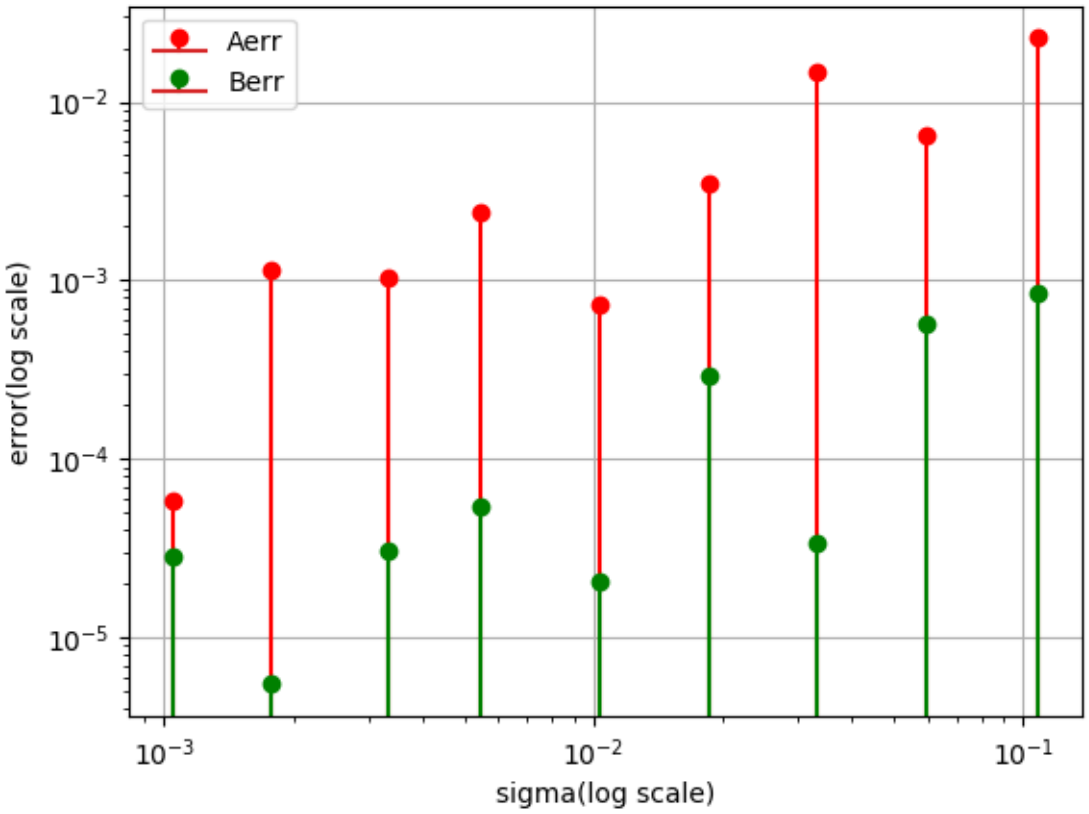

Figure 5: Standard Deviation vs Error in A and B (logscale)

3 Code

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

data=[np.loadtxt('fitting.dat',usecols=(i)) for i in range(0,10)]
def g(t,A,B):
    return A*sp.jn(2,t)+B*t
def stdev(data,t,A0,B0):
    return math.sqrt(sum([(data[i]-g(t[i],A0,B0))**2 for i in range(0,len(
                                                        data))])/len(data)
t = np.array(data[0])
A0=1.05
B0=-0.105

figure('fig1')
[plot(t,data[i],label=r'$\sigma=$' + str(round(stdev(data[i],t,A0,B0),3)))
                                                    for i in range(1,len(data))]
true_value = g(t,1.05,-0.105)
plot(t,true_value,label='True Value')
grid(True);legend(loc='upper right');xlabel('t');ylabel('f(t)');savefig
                                                        ('fig1_fin.png')

figure('fig2')
errorbar(data[0][::5],data[1][::5],stdev(data[1],data[0],A0,B0),fmt='ro',
                                                        label='errorbar')
plot(t,true_value,label='True Value')
grid(True);xlabel('t');legend(loc='upper right');savefig('fig2_fin.png')

j = [sp.jn(2,i) for i in t]
M = c_[j,t]
G = np.matmul(c_[[sp.jn(2,i) for  i in t],t],np.array([A0,B0]))
print ('equal') if max([G[i]-true_value[i] for i in range(0,len(G))])<1e-1
                                                    else print('not equal


A_calc_list=np.array([np.linalg.lstsq(M,data[i],rcond=-1)[0][0] for i in
                                                    range(1,len(data))])
```

```
B_calc_list=np.array([np.linalg.lstsq(M,data[i],rcond=-1)[0][1] for i in
                                                    range(1,len(data))])
A = linspace(0,2,21);B= linspace(-0.2,0,21)
epsilon=np.array([sum([(data[1][z]-(A[i]*sp.jn(2,t[z])+B[j]*t[z]))**2 for
z in range(len(data[1]))])/len(data[1]) for i in range(len(A)) for j in
                                                    range(len(B))])
epsilon.shape=(len(A),len(B))
figure('fig3')
con= contour(A,B,epsilon)
plot([A_calc_list[0]],[B_calc_list[0]],'ro')
clabel(con,inline=1,fontsize=10);xlabel('A');ylabel('B')
savefig('fig3_fin.png')

std_dev = [stdev(data[i],data[0],A0,B0) for i in range(1,len(data))]
figure('fig4')
plot(std_dev,abs(A_calc_list-A0),'ro',label='Aerr')
plot(std_dev,abs(A_calc_list-A0),'r',linestyle='--')
plot(std_dev,abs(B_calc_list-B0),'go',label='Berr')
plot(std_dev,abs(B_calc_list-B0),'g',linestyle='--')
legend();grid(True);xlabel('sigma');ylabel('error')
savefig('fig4_fin.png')

figure('fig5')
stem(std_dev,abs(A_calc_list-A0),'r','ro',label='Aerr')
stem(std_dev,abs(B_calc_list-B0),'g','go',label='Berr')
legend();xscale('log');yscale('log');ylabel('error(log scale)')
xlabel('sigma(log scale)');grid(True);savefig('fig5_fin.png')
show()
```