EE2703 : Applied Programming Lab
Assignment 3
Fitting Data To Models

T.M.V.S GANESH
EE19B124

March 7, 2021

Introduction

The assignment is about

- Analysing the Data to Extract the information
- Effect of Noise on fitting
- Plotting Graphs using Pylab module

Q1

The generate_data.py file generates a text file(fitting.txt)which contains the Data

Python code:

```
    t = linspace(0,10,N)                          # t vector
    y = 1.05*sp.jn(2,t)-0.105*t                   # f(t) vector[True value]
    Y = meshgrid(y,ones(k),indexing='ij')[0]      # make k copies
    scl = logspace(-1,-3,k)                         # noise stdev
    n = dot(randn(N,k),diag(scl))                   # generate k vectors
    yy = Y+n                                         #Data with noise added
    savetxt("fitting.dat",c_[t,yy])              # write out matrix to file
```

About the Data:
The Data File contains 10 columns of Data of which the first one is Time measured from 0 to 10 sec with a step of 0.01
The remaining nine columns of Data is a value of function $g(t, A, B)$ with random noise added to it with different standard deviations

The Data corresponds to the functions:

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

$$f(t) = g(t, A_0, B_0) + n(t) \tag{2}$$

where: $g(t, A, B)$ = Data without noise for given A,B,t
$J_2(t)$ = Second order Bessel function of the first kind.
t = time
$n(t)$ = random noise
$f(t)$ = Data
A_0, B_0 = 1.05 , -0.105

Q2

The data in file can be extracted using `loadtxt` command which loads the Data into a matrix
Python code:

```
    DATA = loadtxt('fitting.dat',delimiter =' ') #EXTRACTING THE DATA
    TIME = DATA[:,0]
```

Array `DATA` contains 10 columns of the text file and columns can be extracted by the index of the column
As the First Column corresponds to time `DATA[:,0]` is the column vector of time

Q3,4

The noise corresponding to data in above function(equation 2) is a random point from the normal distribution with a given σ_n
The probability distribution of normal distribution is given by:

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

where: σ = Standard Deviation of Normal Normal Distribution

σ_n is defined by `logspace` command.This creates nine values btweem 0.1 and 0.001 whose logarithmic values are equally spaced between -1 and -3
Python code:

```
def g(time,A,B):                                #function to calculate the expression
     without noise
    result = A*(sp.jn(2,time))+B*(time)
    return result

TRUE_VALUE = g(TIME,1.05,-0.105)
sub_s = ['\u2080','\u2081','\u2082','\u2083','\u2084','\u2085','\u2086','\u2087',
    '\u2088','\u2089'] #unicode of subscripts
episilon_uni  ='\u03B5'
sigma_uni = '\u03C3'
# shadow plot
figure(figsize= (10,10))
xlabel(r'$t$',size=20)
ylabel(r'$f(t)+n$',size=20)
title(r'Q4:Plot of the data to be fitted',size=20)
for i in range(1,10):           # Graph of data with different noise
    plot(TIME,DATA[:,i],label = ''+sub_s[i]+str('=') + str(round(scl[i-1],4)),
   alpha = 0.8)
```

```
plot(TIME,TRUE_VALUE,label = 'True Value',color = "b")
legend(loc = 'lower left',ncol=1,title='Sigma values')
grid(True)
show()
```

The above code creates a function g(t,A,B) and plots the Data with different noise and also true value without any noise
The values of function over time with $A_0 = 1.05$ and $B_0 = -0.105$ are generated and treated as True Value
Graph :

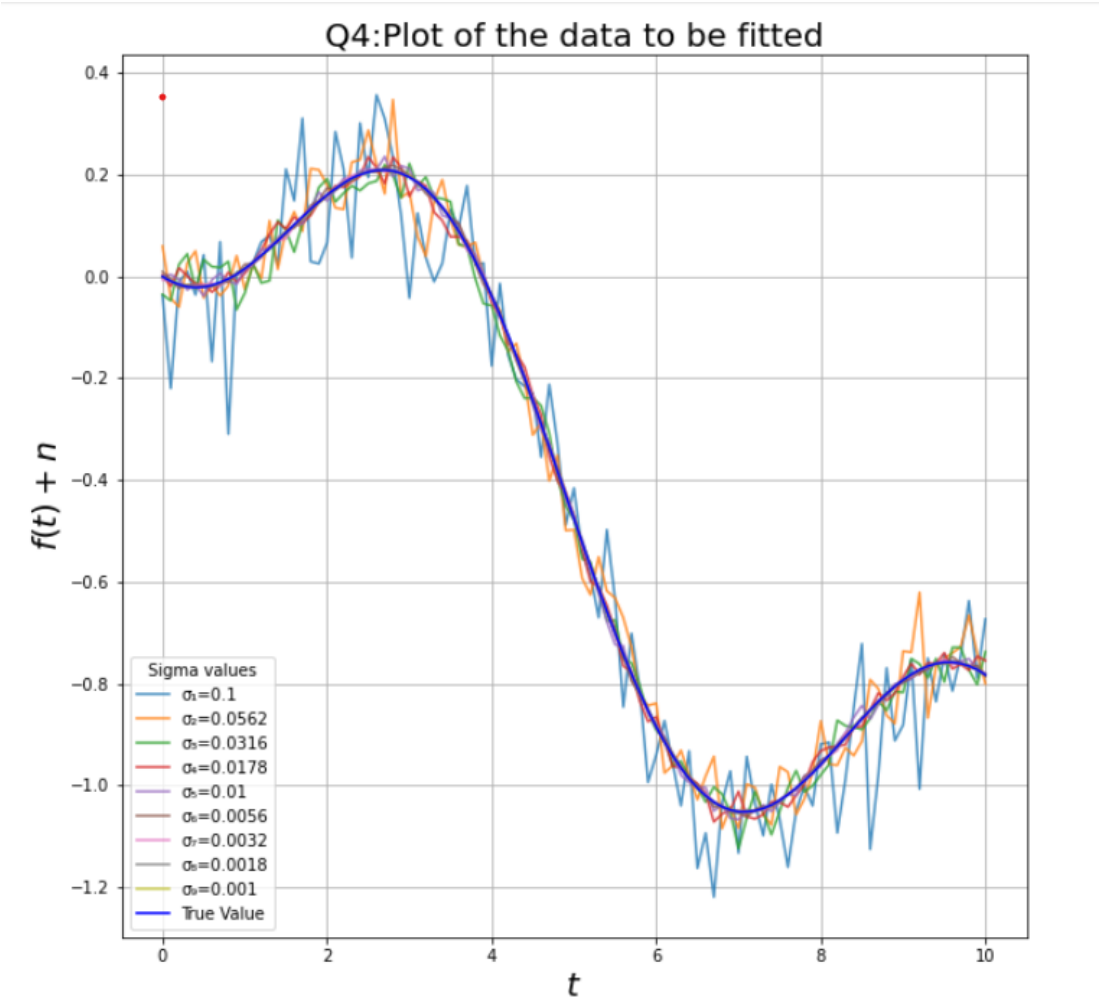

Figure 1: Data vs Time Graph with different noise

The graphs are labelled accordingly with σ_n value and as True Value

Q5

Plotting the Error bars:
Python code:

```
#Question5

figure(figsize=(10,10))  #Graph of error of  1st column of data with that of True
    value
xlabel(r'$t$',size=20)
ylabel(r'$Data$',size=20)
title(r'Q5:Data Points for  along with exact function',size=20)
for i in range(9):
    errorbar(TIME[::5],DATA[:,i+1][::5],scl[i],fmt='o',label='for'+ ''+sub_s[i
    +1])
plot(TIME,TRUE_VALUE,label = 'f(t)[True Value]')
grid(True)
legend(loc = 'upper right',ncol= 2,fontsize = 10,title = 'Error bars')
show()
```

The above code plots the True Value and every fifth point of the data with noise and Dispersion around the value according to the Standard Deviation.
We can observe from the probability distribution of normal distribution that as sigma decreases the probability of $n(t) = 0$ increases by orders of magnitude and the dispersion around the value decreases.
Graphs :

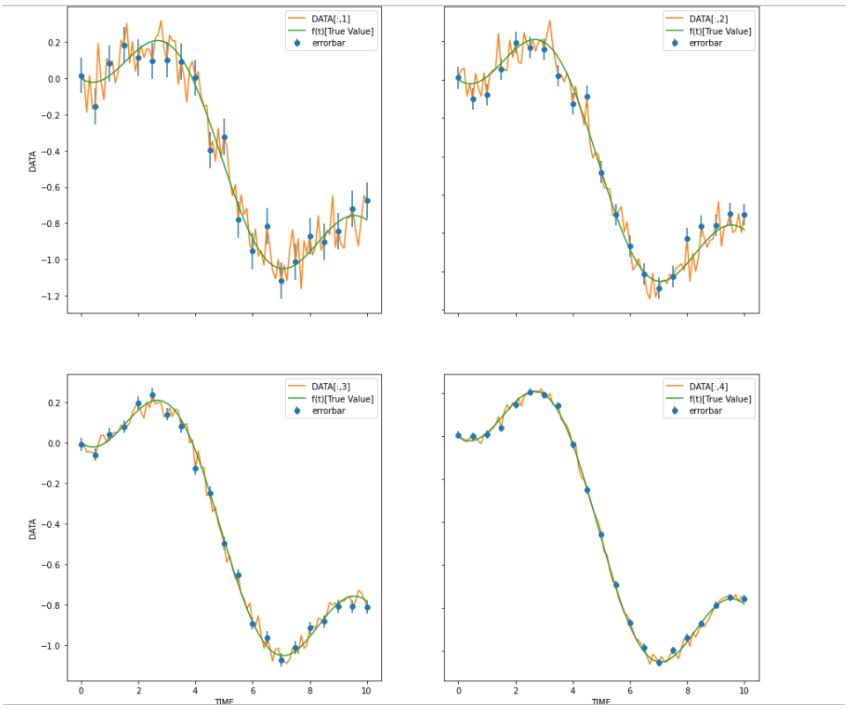

Figure 2: Error bars of DATA with Various σ

Q6

Generating the True value by using Matrix Equations:
Python code:

```
#Question6

J_vector = []
for i in range(0,N):
    J_vector.append(sp.jn(2,TIME[i]))
M = c_[J_vector,TIME]
p = c_[[1.05,-0.105]]
Matrix_value = np.array(np.dot(M,p)) #Calculating data from dot product of
    vectors
formula_value =c_[g(TIME,1.05,-0.105)] #Calculating data from the formula
if np.array_equal(Matrix_value,formula_value):
    print("They are Equal")
else:
    print('Check if you have changed something above')
```

M is a 101×2 matrix and p is 2×1 matrix resulting in a 101×1 matrix which is the values of True Value
np.array.equal command checks whether all the corresponding values of the two arraya are or not.

Q7

Computing the Mean Square Error for Differnt Values of A and B with respect to that of the First column of Data:

$$\epsilon_{ij} = \frac{1}{101}\sum_{k=0}^{101}(f_k - g(t_k, A, B))^2 \tag{4}$$

```
#Question7

Error_matrix = [np.zeros(21) for i in range(21)] #errors with all different
    values of A and B (21*21 values)
A = linspace(0,2,21)                              #with respect to 1st column of
    Data
B = linspace(-0.2,0,21)
for i in range(21):
    for j in range(21):
        Error_matrix[i][j] = np.sum((DATA[:,1] - g(TIME,A[i],B[j]))**2)*(1/101)
```

Q8

Contour plot of ϵ_{ij} :
Python Code:

```
#Question8

figure(figsize=(10,10))
val = contour(A,B,Error_matrix,15)  #Contour plot of error (20 lines will be
    plotted)
plot(1.05,-0.105,marker = 'o',color = 'b')
text(1.07,-0.105,'True Value',color = 'g',size=15)
clabel(val,val.levels[:6],fontsize=10)
xlabel('A',size=20)
ylabel('B',size=20)
title('Q8:Contour plot of ',size=20)
```

The above code creates a Error_matrix of dimension 21×21 with all possible combinations of A anb B values

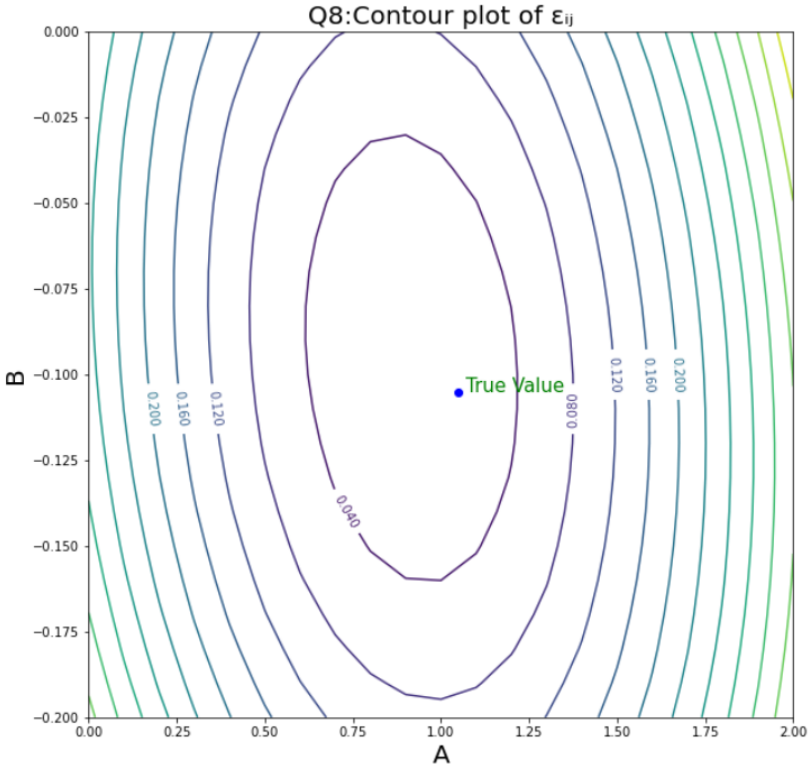

Figure 3: Contour plot of Mean Square Error

The value of ϵ are labelled in the line and as th value of A and B move away from The true values the error increses and the plot has a minimum at values of A and B which best fits the data in 1st column of Data

Q9

Finding the Best Estimate of A and B:
Using the lstsq command from scipy.linalg it solves the M matrix defined earlier and the column vectors of DATA with noise
Python Code:

```
#Question 9

A_err = [np.zeros(1) for i in range(9)]
B_err = [np.zeros(1) for i in range(9)]
for i in range(1,10):
    p=lstsq(M,DATA[:,i],rcond=None)  #estimating the A,B which fits for data with
     different noises
    A_err[i-1]=abs(p[0][0]-1.05)      #error of A,B with that of A,B
    corresponding to true value
    B_err[i-1]=abs(p[0][1]+0.105)
```

Q10

The above code in Q9 returns the best estimate of A and B and A_error an B_error are the values of absolute difference between the A,B and A_0,B_0
The best estimate of A and B is the position of minimum in the contour plot of the graph above 3
Python Code:

```
#Question 10

figure(figsize=(10,10))
for i in range(9):
    axvline(scl[i],color='g',alpha = 0.5) # plotting the error values with that
    of sigma values
xlabel('Standard Deviation of Noise',size=20)
ylabel('MS error',size=20)
title('Variation of error with noise',size = 20)
plot(scl,A_err,'--ro',label='Aerror')
plot(scl,B_err,'--bo',label='Berror')
legend(loc = 'upper left',fontsize = 15)
grid(True)
show()
```

ERROR PLOT:

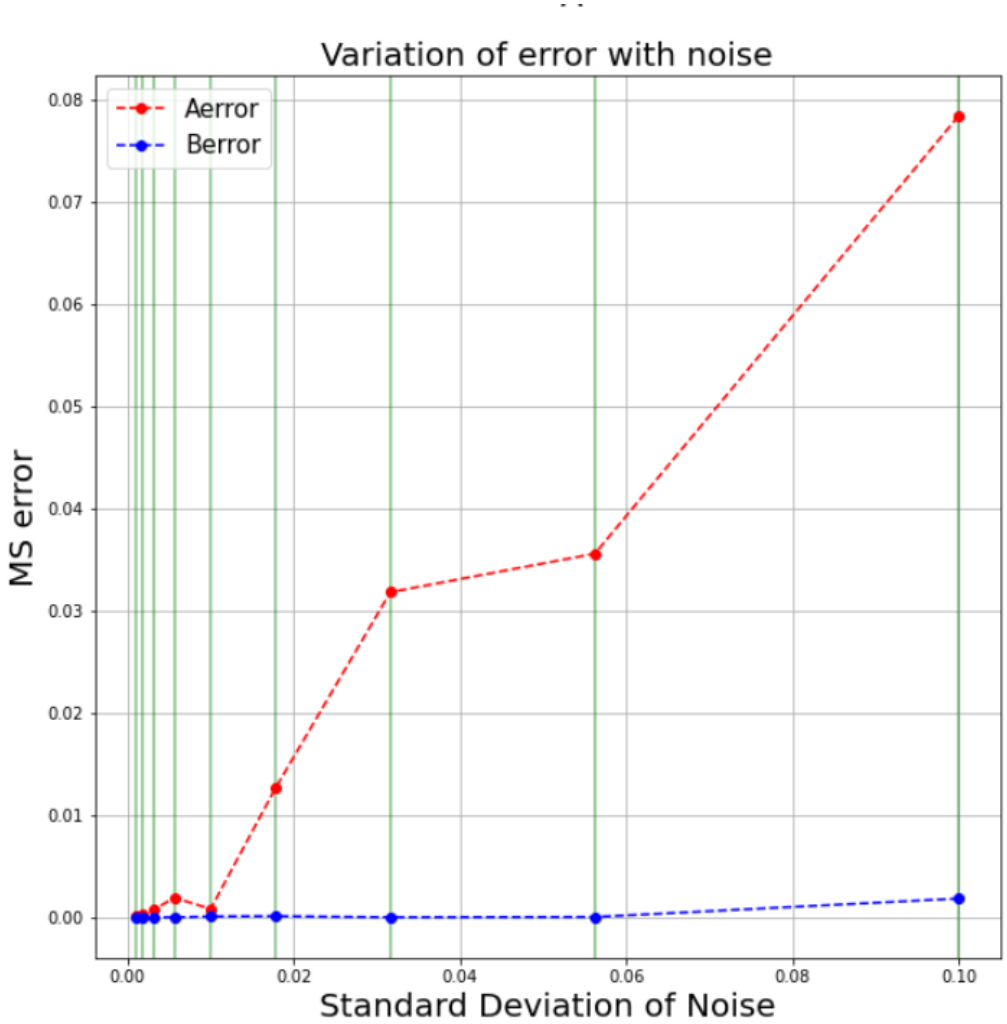

Figure 4: Variation of error in A,B with σ

The plots of error are not linearly varying withσ as the values of sigma are not uniform on normal scale and are concentrated towards 0.01

Q11

Replotting the above graphs in log scale:
Python Code:

```
#Question 11

figure(figsize =(10,10))   #plotting the above graph in log scale
xlabel('Standard Deviation of Noise',size=20)
ylabel('MS error',size=20)
title('Variation of error with noise in log scale',size = 20)
```

```
for i in range(9):
    axvline(scl[i],color='g',alpha = 0.5)
loglog(scl,A_err,'--ro',label='Aerror')
loglog(scl,B_err,'--bo',label='Berror')
legend(loc = 'upper left',fontsize = 15)
show()
```

loglog command plots the logarithmic values of x vs logarithmic values of y and σ is uniformly defined on the log scale.
Graph:

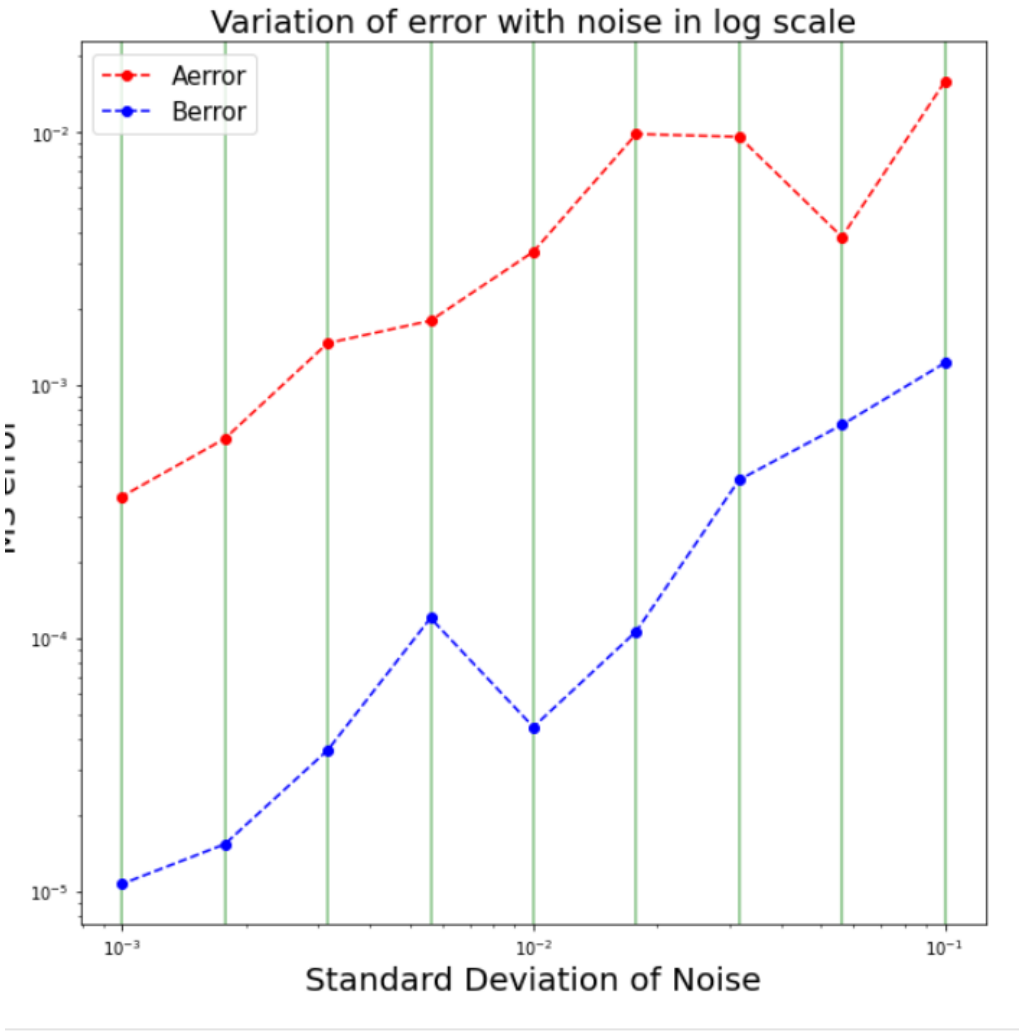

Figure 5: Variation of error in A,B with σ

The Plots are nearly linear in the log scale as the dispersion in the data increases with that of σ and the randomness of noise can be treated as the noise in value of A and B
Therefore the variation in A and B varies linearly with σ when sigma is high the error is high and is low when σ is low