

# **EE2703 : Applied Programming Lab Assignment 3**

RAGHU DINESH T  
EE18B144

February 13, 2020

## Q1 Creating the data file

**Run the given python3 program.** A datafile that contains time in its first column and corresponding bessell function values with noise of 9 different standard deviations, which vary logarithmically from 0.1 to 0.01, in the next 9 columns is generated.

### Codes

This is the python3 program that is supposed to be executed.

```
# script to generate data files for the least squares assignment
from pylab import *
import scipy.special as sp
N=101 # no of data points
k=9
# no of sets of data with varying noise
# generate the data points and add noise
1t=linspace(0,10,N)
# t vector
y=1.05*sp.jn(2,t)-0.105*t # f(t) vector
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
# add noise to signal
# shadow plot
plot(t,yy)
xlabel(r"$t$",size=20)
ylabel(r"$f(t)+n$",size=20)
title(r"Plot of the data to be fitted")
grid(True)
savetxt('fitting.dat',c_[t,yy]) # write out matrix to file
show()
```

### Results

A file named “**fitting.dat**” that contains all the mentioned data with randomised noise is generated.

## Q2 Extracting the Data

Extract the data contained in “**fitting.dat**” into a variable as a numpy array. This can be done using the `loadtxt('fitting.dat')` command.

### Codes

The following command loads all the data into a variable ‘**data**’

```
data=pl.loadtxt("fitting.dat")
```

## Q3 Plotting the Data

**Plot** all the data extracted, with the **first column of data on X-axis** and each of the **others columns on Y-axis**.

### Codes

The following block of code displays the required 9 graphs with their corresponding standard deviation, sigma.

```
t=data[:,0]
values=data[:,1:]
sigma=pl.logspace(-1,-3,9)
y_true=g(t,1.05,-0.105)

for i in range(len(sigma)):
    pl.plot(t,values[:,i],label="sigma "+str(sigma[i]))

pl.plot(t,y_true,label="true value")
pl.title('Figure 0')
pl.xlabel('t')
pl.ylabel('Values')
pl.legend()
pl.show()
```

## Results

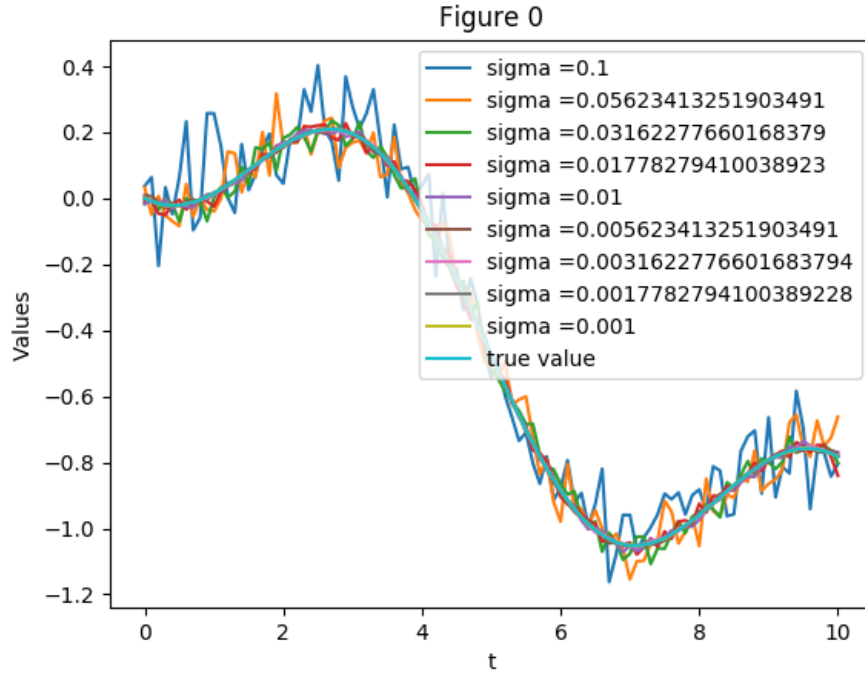


Figure 1: Plot of values Vs time for different sigma of values

## Conclusions

From Figure 5, it is clear that as the sigma increases, the variations of the values from the true value also increase.

## Q4 Define the expected function of the model

Define a function  $g(\cdot)$  which takes the time array and two numbers  $A$ ,  $B$  as arguments and return the value of expression  $g(t, A, B) = A * sp.jn(2, t) + B * t$

## Codes

This block of code creates the required function.

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

## Q5 Plotting with Errorbar

**Plot** the values of the first column of the values against time with **errorbar**.  
This can be done by the function `errorbar(t[:5],data[:5],sigma,fmt='ro')`

### Codes

The following block of code creates the required plot.

```
pl.errorbar(t[:5],values[:,0][:5],sigma[0],fmt='ro')
pl.plot(t,y_true,label="true value")
pl.title('errorbar for first column of data')
pl.xlabel('t')
pl.ylabel('Values')
pl.legend()
pl.show()
```

### Results

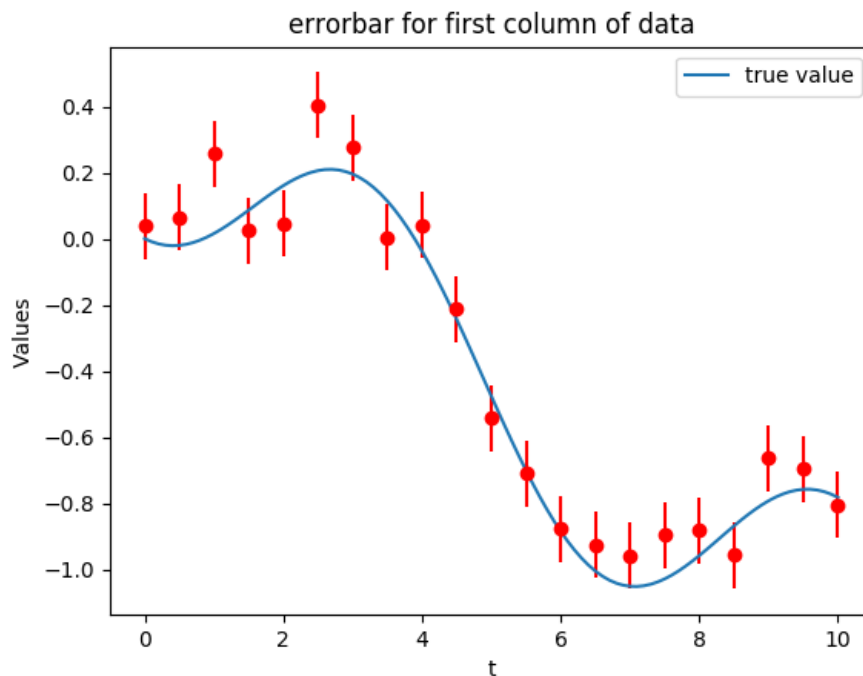


Figure 2: Sample image

## Q6 Matrix multiplication for obtaining g(.)

Create a matrix as the true bessel function values as first column and the time as second column. Obtain g(t,A,B) by multiplying the created matrix with [A,B] and verify whether it is same as the matrix returned by the already created function g(.)

### Codes

The following creates the required matrix and performs matrix multiplication and verifies whether the vectors are equal

```
M=pl.c_[sp.jn(2,t),t]
p=[1.05,-1.05]
print((pl.dot(M,p)==g(t,p[0],p[1])).all())
```

### Results

The results of both the matrix multiplication and the g(.) function are same

## Q7 Mean Square Error(MSE) for different coefficients of Bessel function

For A = 0, 0.1, . . . , 2 and B = 0.2, 0.19, . . . , 0, for the data given in columns 1 and 2 of the file, compute

$$\epsilon_{ij} = \frac{1}{101} \sum_{k=0}^{101} (f_k - g(t_k, A_i, B_j))^2$$

### Codes

The following block of code creates a matrix containing the MSE values for different A and B

```
A=pl.arange(0,2.1,0.1)
B=pl.arange(-0.2,0.01,0.01)
print(A,B)
e=pl.zeros([len(A),len(B)])
```

```

k=0
for i in A:
    p=0
    for j in B:
        e[k][p]=(pl.sum((values[:,0]-g(t,i,j))**2))
        p+=1
    k+=1

```

## Results

A matrix e is created where elements  $e[i][j]$  corresponds to  $A[i]$  and  $B[j]$

## Q8 Plotting a contour plot of MSE

Plot the MSE contour for different values of A and B

## Codes

The following block of code generates the contour.

```

a, b=pl.meshgrid(A, B)
cp=pl.contour(a,b,e,21)
pl.clabel(cp)
pl.xlabel("A")
pl.ylabel("B")
pl.show()

```

## Results

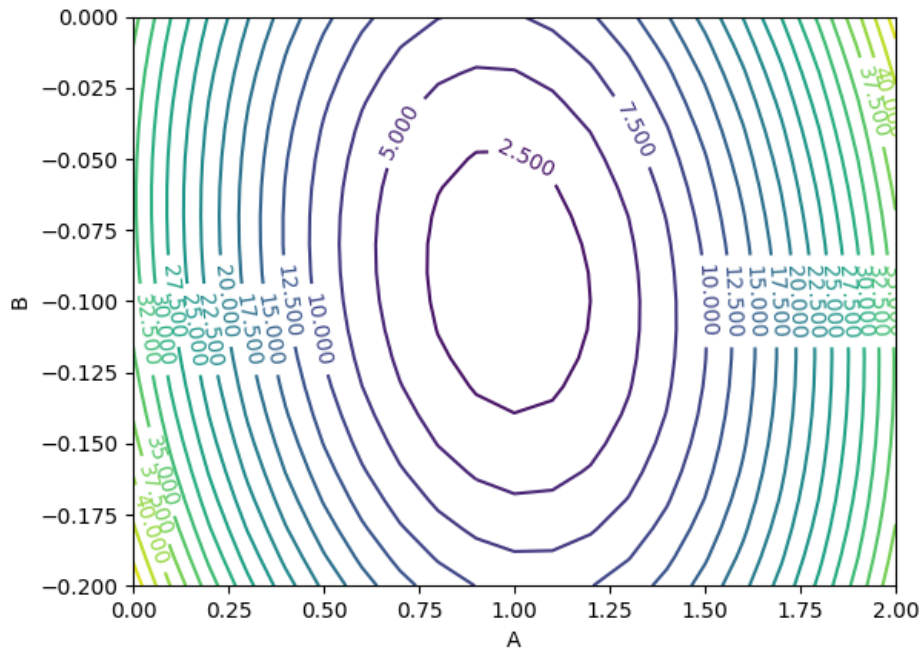


Figure 3: Sample image

## Conclusions

The contour converges to a minimum value where A and B are 1.05 and -0.105

## Q9 Finding the best estimate of A and B

Use the Python function `lstsq` from `scipy.linalg` to obtain the best estimate of A and B for the first column of data.

## Codes

The following block of code give the best estimate of A and B

```
A_best, B_best=spl.lstsq(M,values[:,0])[0]
print("For first Column: the best estimate of A={} and B={}".format(A_best,B_best))
```

## Results

A=1.03 and B=-0.106, which are very good estimate of the true function which has A=1.05 and B=-0.105

## Q10 Error in the estimate of A and B for different data files versus the noise $\sigma$

Calculate the best estimate of A and B for all columns of data and **Plot** the error in the estimate of A and B for different data files versus the noise  $\sigma$

## Codes

```
A_err=[];B_err=[];
for i in range((len(sigma))):
    A_best, B_best=spl.lstsq(M,values[:,i])[0]
    A_err.append(abs(A_best-1.05)); B_err.append(abs(B_best+0.105))
pl.subplot(211)
pl.title("Estimation of A")
pl.xlabel("Noise standard deviation")
pl.ylabel("MS Error in A")
pl.plot(sigma,A_err,color="red",marker='o',linestyle='dashed')
pl.subplot(212)
pl.title("Estimation of B")
pl.xlabel("Noise standard deviation")
pl.ylabel("MS Error in B")
pl.plot(sigma,B_err,color="green",marker='o',linestyle='dashed')
pl.show()
```

## Results

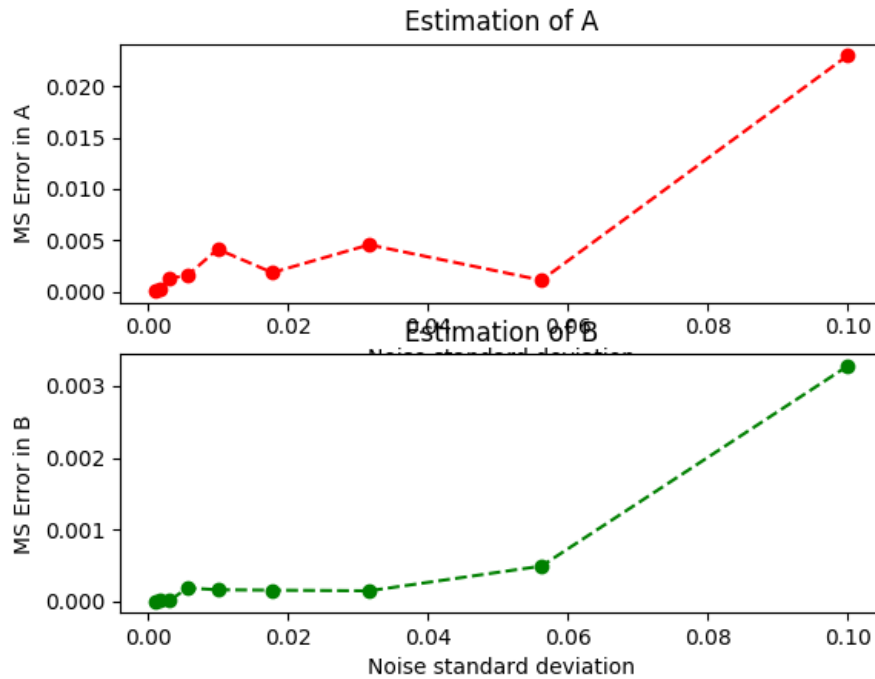


Figure 4: Sample image

## Conclusions

The error in A and B do not vary linearly with Noise standard deviation.

## Q11 Plotting $\log(\text{Error in A or B})$ Vs $\log(\sigma)$

Replot the above curves using loglog.

## Codes

```
pl.title("log(A_err) vs log(sigma) and log(B_err) vs log(sigma)")
pl.xlabel("log(Noise standard deviation)")
pl.ylabel("log(MSError)")
pl.loglog(sigma,A_err,color="red",marker='o',linestyle="None",label="A_err")
pl.errorbar(sigma,A_err,sigma,color="red",linestyle="None")
pl.loglog(sigma,B_err,color="green",marker="o",linestyle="None",label="B_err")
```

```

pl.errorbar(sigma,B_err,sigma,color="green",linestyle="None")
pl.legend()
pl.show()

```

## Results

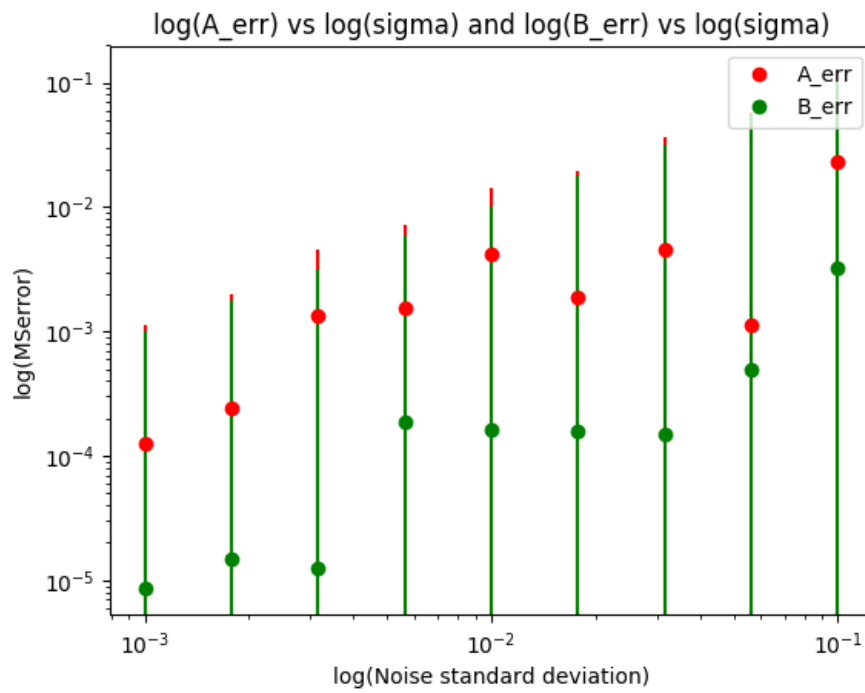


Figure 5: Sample image

## Conclusions

The log of error in A and B do not vary linearly with log of Noise standard deviation.