EE2703 Applied Programming Lab Assignment 3

Name: Hemanth Ram
Roll Number: EE18B132

February 9, 2020

1 Overview:

In this assignment, we generate noisy data of a given linear combination of the *Bessel* function. We visualize the data by plotting it and its errorbars. We also try to see how the MSE varies with the parameters through a contour plot. Then we estimate the parameters from the data using Least Mean Square approximation and analyze the error in the parameters.

2 Generated Outputs along with the Code

Importing required libraries:

```
%matplotlib qt
from pylab import *
from scipy.special import *
x = arange(0,10,0.1)
y = jv(0,x)
y1 = jv(1,x)
plot(x,y,x,y1)
show()
```

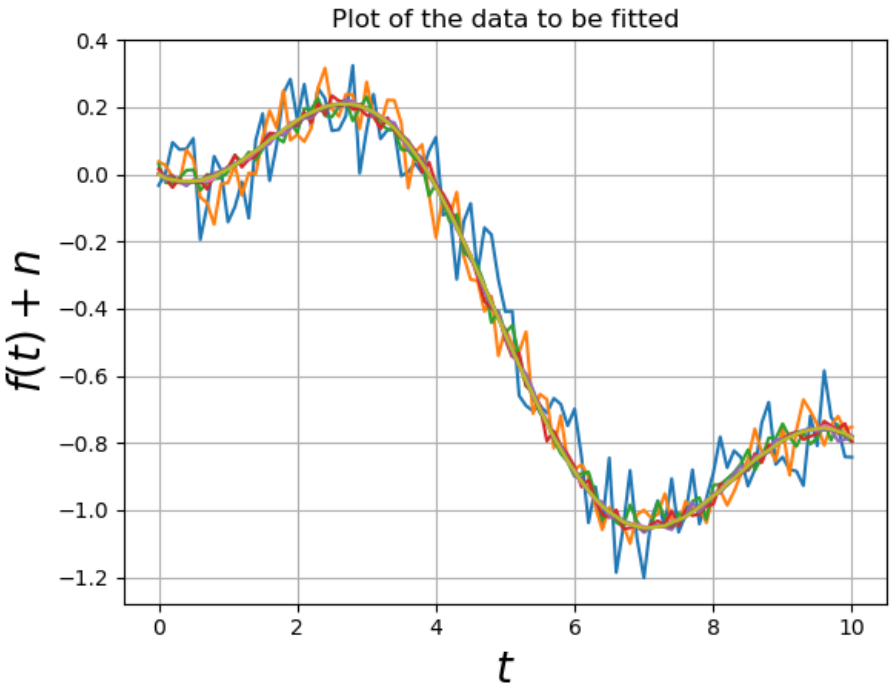
2.1 Part 1

Below code was given to generate the data with different errors for the assignment.

```
# script to generate data files for the least squares assignment
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
t=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()
```



2.2 Part 2

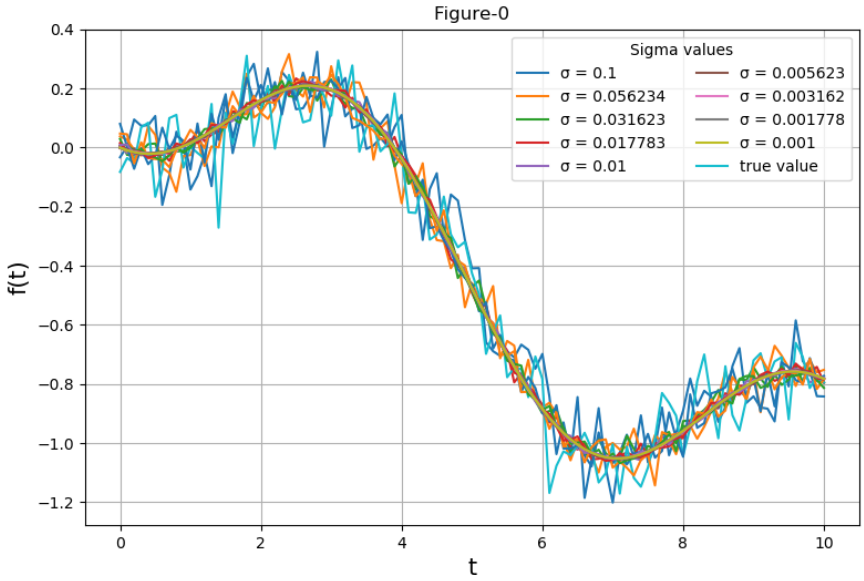
Loading the generated data and generating labels for the plot coming up next with different sigmas.

```
import numpy as np
a = loadtxt('fitting.dat', delimiter=' ')
labels = []
sigmas = logspace(-1,-3,9)
for i in range(9):
    labels.append('\u03C3 = '+str(np.round(sigmas[i],6)))
labels.append('true value')
```

2.3 Part 3-4

Plotting the true value along with the ones with errors generated above.

```
def gfunc(t,A,B):
    res = A*(jv(2,t))+B*(t)
    return res
A = 1.05
B = -0.105
g = gfunc(a[:,0],A,B)
plot(a[:,0],a[:,1:])
plot(a[:,0],g)
legend(labels,loc = 'upper right',ncol=2,title='Sigma values')
title('Figure-0')
xlabel('t',size = 15)
ylabel('f(t)',size = 15)
show()
```

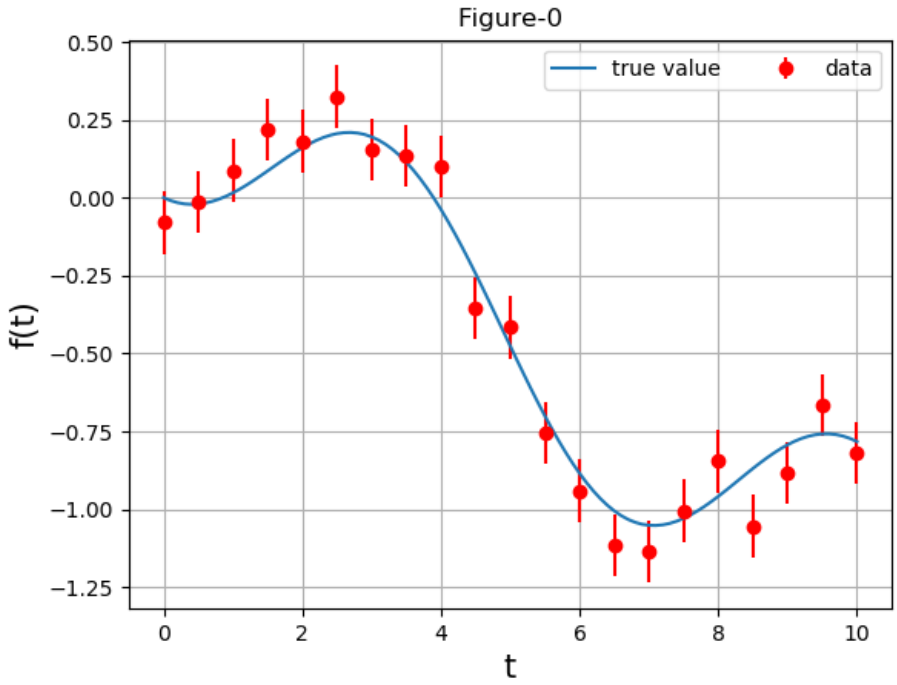


2.4 Part 5

Plotting the errorbar asked for the first coloum of data (with sigma = 0.1). Errorbar gives us the possible range of error visually in a plot.

```
errorbar(a[:,0][::5],a[:,1][::5],0.1,fmt = 'ro')
plot(a[:,0],g)
title('Figure-0')
xlabel('t',size = 15)
ylabel('f(t)',size = 15)
legend(['true value','data'],loc = 'upper right',ncol=2)
show()
```

The plot is as shown in the next page.



2.5 Part 6

Generating the M matrix by generating the J coloum separately and joining it with the first coloum (*t data*) of the generated data. Mulitplying it with p and checking if result matches with the one generated previously using *np.array_equal()*.

```
J = jv(2,a[:,0])
p = c_[[A,B]]
M = c_[J,a[:,0]]
g0 = np.dot(M,p)
g = c_[g]
np.array_equal(g,g0)
```

```
Out[64]: True
```

2.6 Part 7

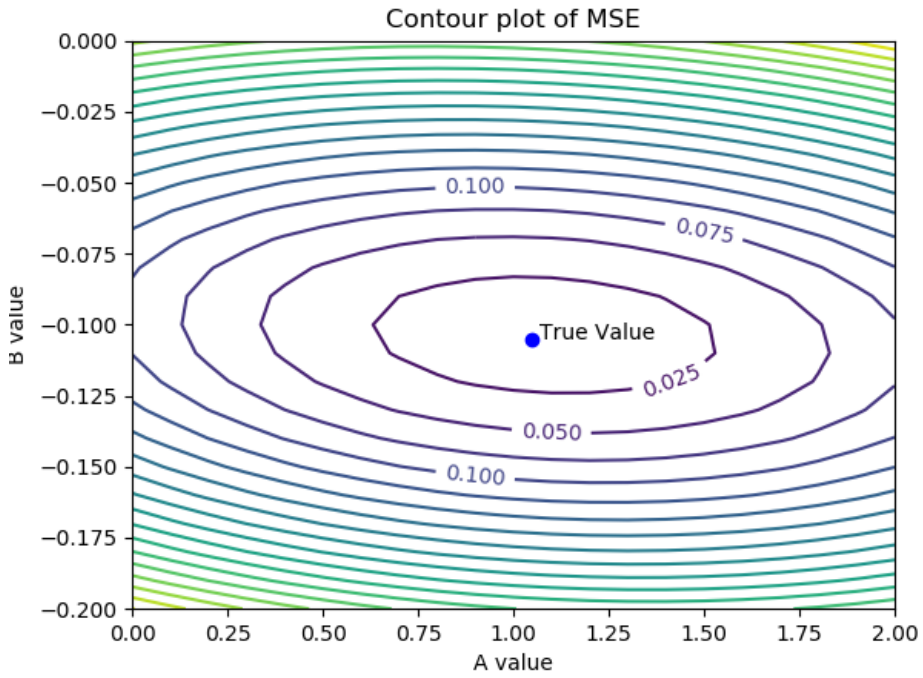
Writing function to find MS error for given A and B parameters by taking difference, squaring and using the *np.sum()*.

```
J = jv(2,a[:,0])
M = c_[J,a[:,0]]
f = c_[a[:,1]]
def MSE(A,B):
    p = c_[[A,B]]
    g0 = np.dot(M,p)
    return np.sum((f-g0)**2)
```

2.7 Part 8

We plot the contour plot of the MSE with A and B. For doing so, we first generate the meshgrid using *np.meshgrid()*, for A ranging from 0 -> 2 and for B ranging from -0.2 -> 0. Then finding the MSE for each pair of A and B possible using the above function and plotting using *contour()*. We mark the true value using the *text()* and using *clabel()* for labeling appropriate values in contour plot.

```
As = arange(0,2.1,0.1)
Bs = arange(0,0-.21,-0.01)
X,Y = np.meshgrid(As,Bs)
E = np.zeros(X.shape)
for i in range(X.shape[0]):
    for j in range(Y.shape[1]):
        E[i][j] = MSE(X[i][j],Y[i][j])/101
Cs = contour(X,Y,E,20)
title('Contour plot of MSE')
plot(1.05,-0.105,marker = 'o',color = 'b')
text(1.07,-0.105,'True Value')
clabel(Cs,Cs.levels[:5],fontsize = 10)
xlabel('A value')
ylabel('B value')
show()
```

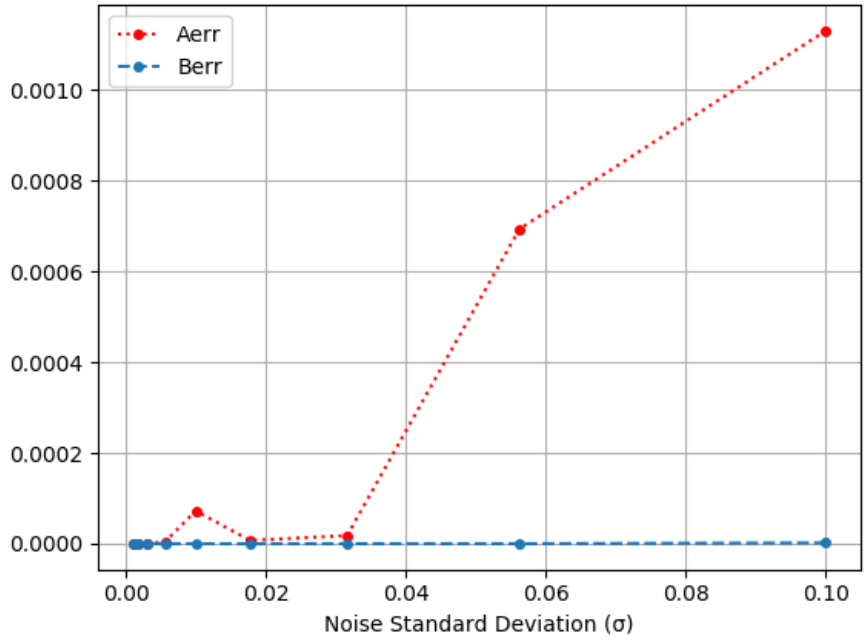


2.8 Part 9-10

Using *scipy.linalg.lstsq()*, finding A and B for minimum MSE possible for each data with different noises. Then, finding error in A and B for different noises from the original values of A and B

which is stored in *AB*. *ae* and *be* contain appropriate errors in A and B. Using this, we generate the plot w.r.t the sigma of the noise.

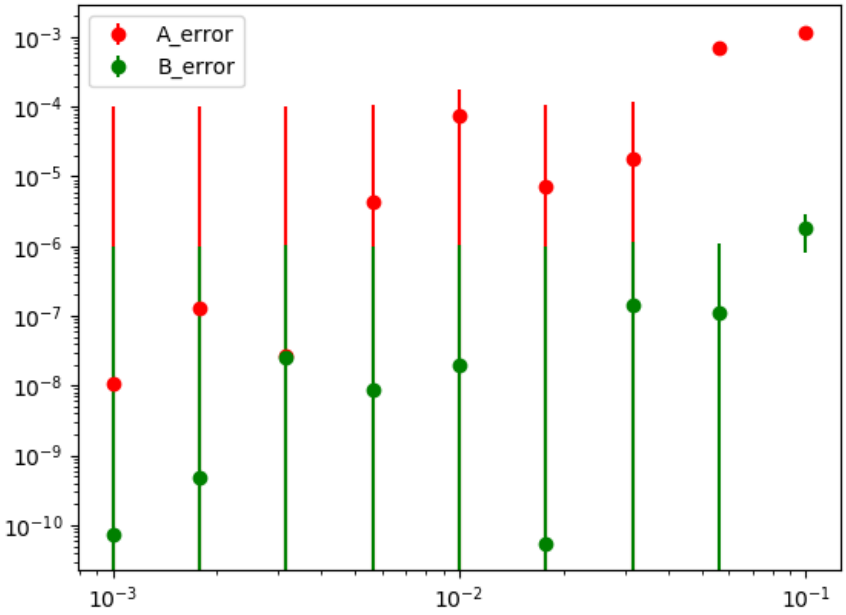
```
import scipy
AB = scipy.linalg.lstsq(M,g)[0]
ae = []; be=[];
for i in range(1,10,1):
    ab = scipy.linalg.lstsq(M,c_[a[:,i]])[0]
    error = AB-ab
    ae.append(error[0]**2)
    be.append(error[1]**2)
plot(sigmas,ae,'ro:',sigmas,be,'o--',markersize=4)
legend(['Aerr','Berr'])
xlabel('Noise Standard Deviation (\u03C3)')
grid()
show()
```



2.9 Part 11

Plotting the same in the logscale using the *loglog()* function, along with the errorbars also.

```
loglog(sigmas,ae,'o',markersize=4)
loglog(sigmas,be,'o',markersize=4)
errorbar(sigmas,ae,0.0001,fmt='ro',label = 'A_error')
errorbar(sigmas,be,0.000001,fmt='go',label = 'B_error')
legend()
show()
```



3 Result

From the above plots, we see that the **logarithm of the errors** in the parameters varies approximately linearly with the logarithm of the **standard deviation of the the noise** added.