EE2703 : Applied Programming Lab
Assignment 3
Report

Kesava Aruna Prakash R L
EE20B061

February 18, 2022

Abstract

This week's assignment focuses on **fitting different data sets into models.** Here, we are implementing linear fitting with the function in scipy module:

```
linalg.lstsq()
```

Furthermore, we will

- load the data from files.
- analyse the data to extract information from that.
- plot the data and use legends.
- plot errorbars to data.
- study the effects on noise on linear fits.
- look at the errors in linear fitting.

1 Data

The data is generated in *generate_data.py* which saves it in *fitting.dat*

The data file contained 10 columns of data with 101 rows. The first column corresponded to the time axis data and other 9 columns contained noisy data with different levels of noise.

The noise generated was normally distributed with sigma generated by the function:

```
sigma = logspace(-1,-3,9)
```

2 Correct data and function

To analyse the noise in the data, the actual function is produced and plotted.

```
import scipy.special as sc
def g(t, A, B):
   return A*sc.jn(2,t)+B*t
```

The actual function is just

```
g(t, 1.05, -0.105) where t = column[0] of the data
```

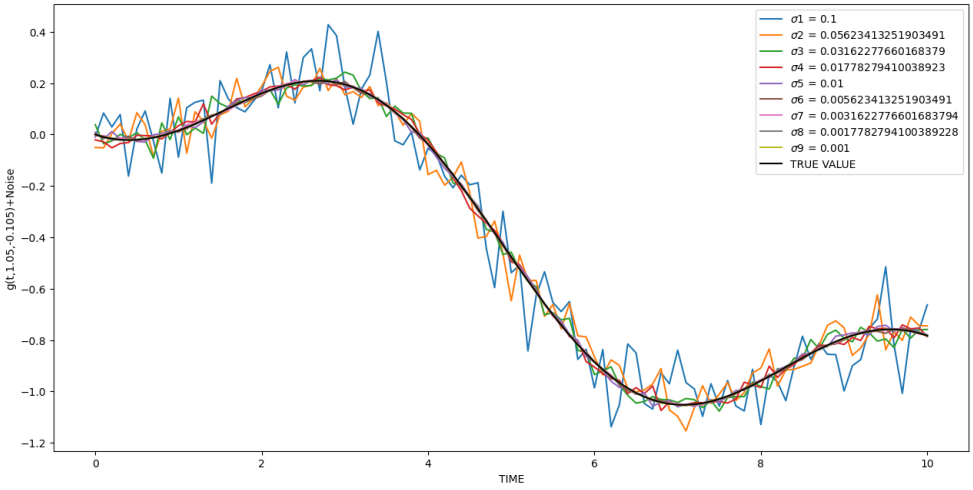

Figure 1: Data plot

In Figure 1, we could see the noisy data over the actual data produced by the function g.
Python code for plotting this:

```
pt.plot(fobj[:,0],fobj[:,1:])
r = []
for i in range(len(sigma)):
    r.append(f"$\sigma${i+1} = {sigma[i]}")
pt.plot(fobj[:,0],g(fobj[:,0],1.05,-0.105),color='k')
r.append("TRUE VALUE")
pt.legend(r)
```

3 Noise

The noise can be visualised by plotting errorbars at the data points to give the range where *correct* data could be.
The errorbar is plotted for noisy data whose corresponding sigma is 0.1.

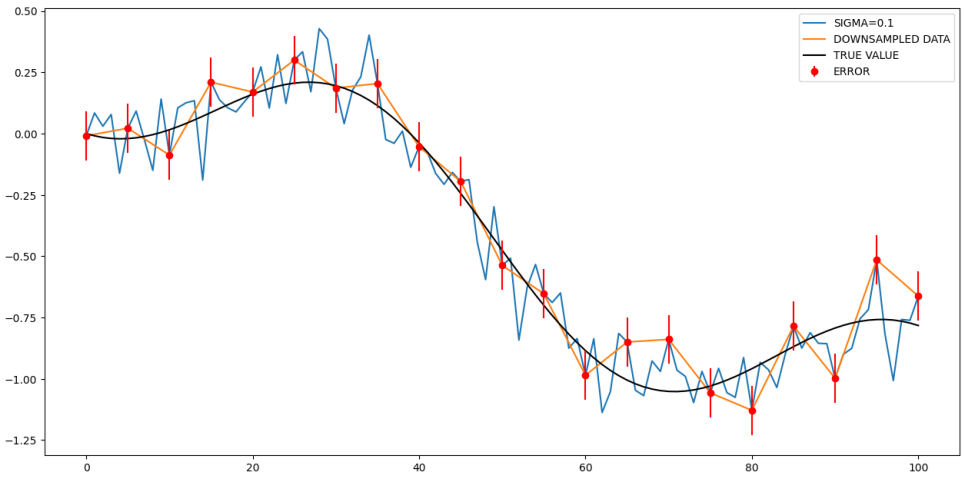

Figure 2: Errorbar plot

Python code for plotting this:

```
h=[]
pt.plot(fobj[:,1])
h.append("SIGMA=0.1")
k=np.arange(0,101,5)
pt.plot(k,fobj[::5,1])
h.append("DOWNSAMPLED DATA")
pt.plot(g(fobj[:,0],1.05,-0.105),color='k')
h.append("TRUE VALUE")
pt.errorbar(k,fobj[::5,1],sigma[0],fmt='ro')
h.append("ERROR")
pt.legend(h)
pt.show()
```

4 Matrix formation

Here g function is expressed as a product of two matrices $M*N$.
M will contain two column vectors

$$Column[0] = J_2(t_i)$$

$$Column[1] = t_i$$

N contains the column vector with A and B values.
This helps later when the best fit of A and B is calculated.

5 Contour plot

Mean square error of noisy data (column[1] of data) from value obtained from g function is found out for different values of A and B

$$e_{ij} = \frac{1}{101}\sum_{k=0}^{100}(f(t_k) - g(t_k, A_i, B_j))^2$$

$$A = 0, 0.1, 0.2, 0.3.....2.0$$

$$B = -0.20, -0.19, -0.18.....0$$

Python code for generating this:

```
e=np.zeros([21,21])
a_array=np.arange(0,2.1,0.1)
b_array=np.arange(-0.2,0.01,0.01)
for i in range(len(a_array)):
    for j in range(len(b_array)):
        for k in range(len(fobj[:,0])):
            e[i][j]+=((fobj[k][1]-g(fobj[k][0],a_array[i],b_array[j]))**2)/101
```

After getting the e matrix, it is plotted as contour in **Z** with A and B for **X** and **Y** respectively. Levels values are added for each contour lines

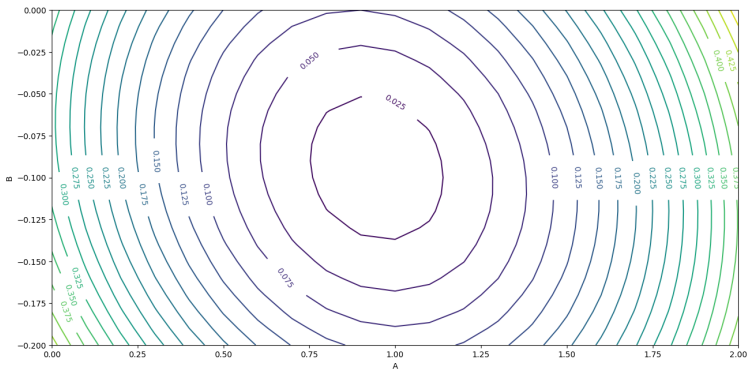

Figure 3: Contour plot

Python code for plotting the contour:

```
cntr =pt.contour(a_array,b_array,e,levels=20)
pt.clabel(cntr)
pt.xlabel("A")
pt.ylabel("B")
pt.show()
```

6 Linear fitting, Errors in the linear fitting

We use inbuilt function in *scipy* to implement the above. The M matrix mentioned above is passed as an argument, and the linear fit is done on the noisy data(column[1] of data) to get the best N matrix.
Python code for implementing the above:

```
fobj=np.loadtxt("fitting.dat",dtype='float')
p,w,q,z=scp.linalg.lstsq(g_array,fobj[:,1])
```

p is an array containing values A and B for the best fit.

$$p[0] = A$$

$$p[1] = B$$

w is the residue in finding the best fit(basically the error).

7 Plots for Errors vs sigma

Now the same procedure is repeated for all the columns of data array(noise data with different sigma), the error in linear fitting, error in A, error in B, log-log plot of error in linear fitting, log-log plot of errors in A and B with errorbars are plotted.

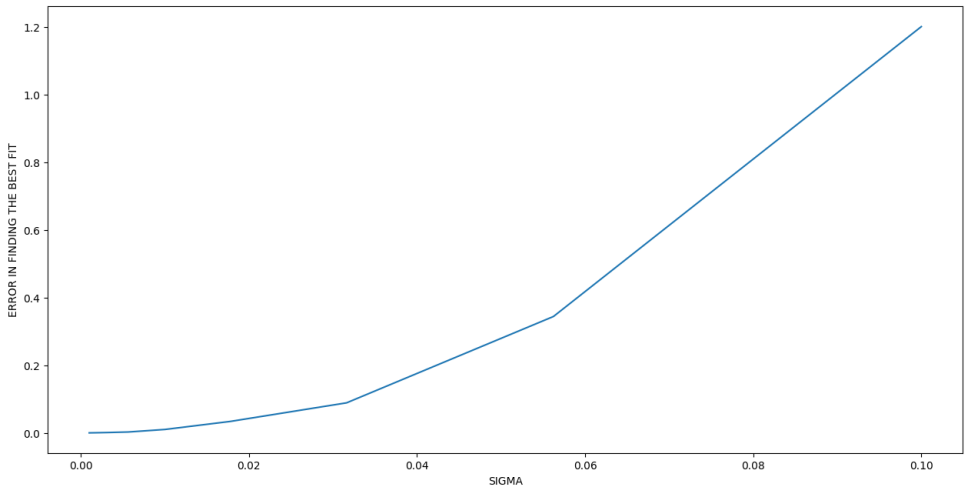

Figure 4: Error in Linear fitting

It is observed that Figure 4 is exponential.

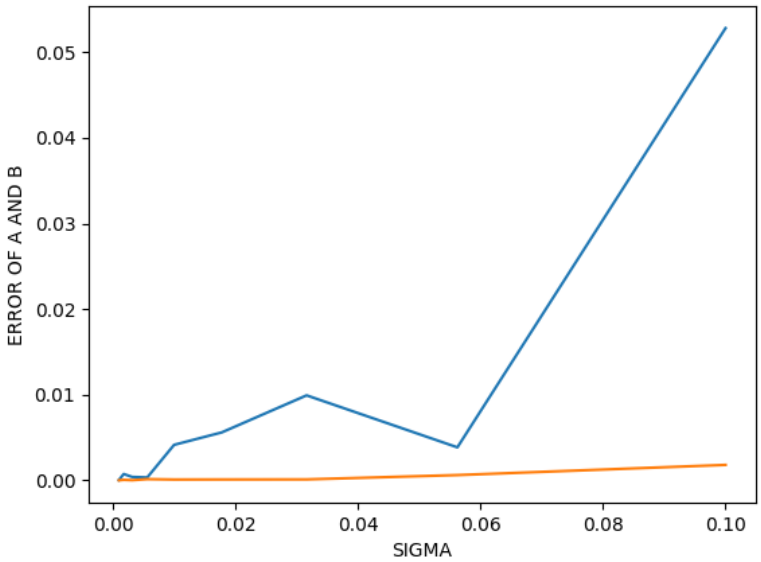

Figure 5: Errors in A and B

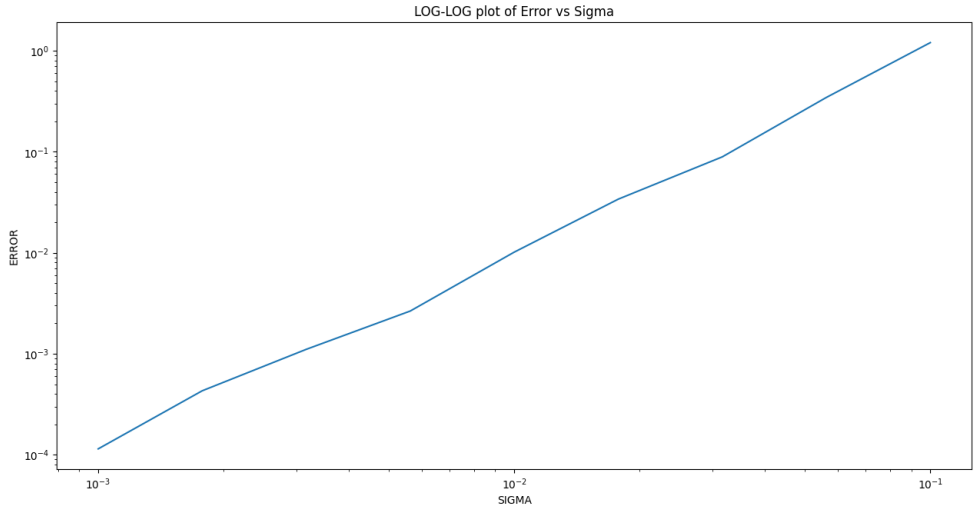

Figure 6: log-log plot of Error in linear fitting

It is observed that Figure 6 is linear.

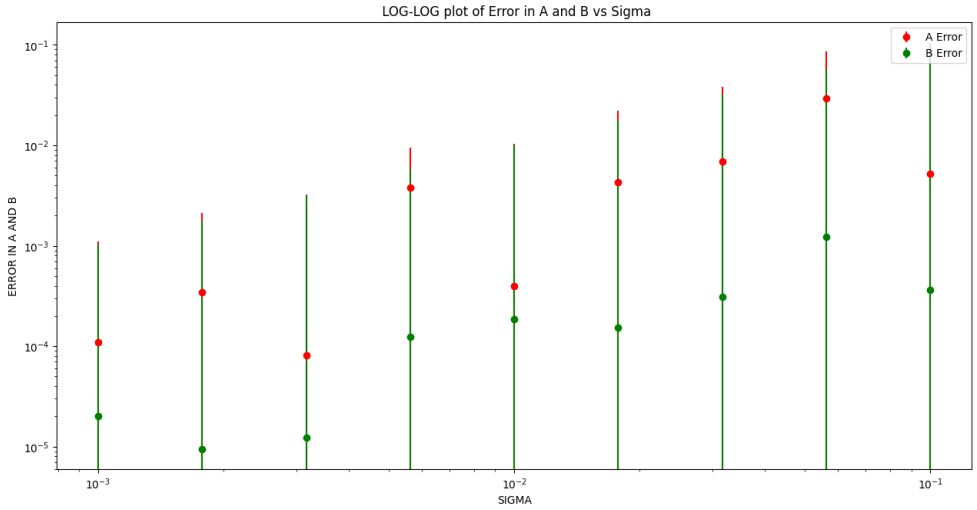

Figure 7: log-log plot of Error in A and B

Figure 7 shows the error in A and B

Python code for the above plots:

```
error=[]
array_a=[]
array_b=[]
for i in range(len(sigma)):
      s,l,q,z=scp.linalg.lstsq(g_array,fobj[:,i+1])
      error.append(l)
      array_a.append(s[0])
      array_b.append(s[1])
pt.plot(sigma,error)
pt.xlabel("SIGMA")
pt.ylabel("ERROR IN FINDING THE BEST FIT")
pt.show()

sum=0
a_error=[]
b_error=[]
for i in array_a:
    sum+=i/len(array_a)
for j in array_a:
    a_error.append(((j-1.05)**2)**0.5)
#print(a_error)
sumb=0
for i in array_b:
    sumb+=i
for j in array_b:
    b_error.append(((j+0.105)**2)**0.5)
pt.plot(sigma,a_error)
#pt.show()
pt.plot(sigma,b_error)
pt.ylabel("ERROR OF A AND B")
pt.xlabel("SIGMA")
pt.show()

pt.loglog(sigma,error)
pt.title("LOG-LOG plot of Error vs Sigma")
pt.xlabel("SIGMA")
pt.ylabel("ERROR")
pt.show()

pt.errorbar(sigma,a_error,sigma,fmt='ro')
pt.errorbar(sigma,b_error,sigma,fmt='go')
```

```
leg=["A Error","B Error"]
pt.legend(leg)
pt.title(" LOG-LOG plot of Error in A and B vs Sigma")
pt.ylabel("ERROR IN A AND B")
pt.xlabel("SIGMA")
pt.loglog()
pt.show()
```

8 Conclusions and Inference

- By minimising the Mean Square Error of the data points using least square method, a close approximation to A and B are obtained.
- This can be verified by looking at the minima in the counter plot.
- Error in A is greatly affected by amplitude of noise, while B is not greatly affected by it.
- Thus the noisy data obtained is fitted into a well defined model.
- This has many applications in real-life.