Assignemnt 3: Fitting Data to Models

Gudivada Harshad Kumar
EE20B038

March 8, 2022

1 Abstract

The main aim of the assignment Fitting Data to Models is

- Reading "fitting.dat" file and parse data from it
- Analyse the data to extract information out of it
- To study the effect of noise on fitting process
- Plotting graphs for good understanding

2 Generating fitting.dat and analyzing and extracting given data

On running the given *generate.py* file we get a plot which is shown below and a file named fitting.dat is generated. The plot consists of a function with noises. The generated *fitting.dat* file consists of 10 coloumns with 101

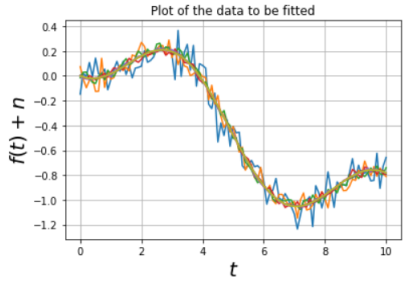

Figure 1: Data Plot

rows of data. The first coloumn is the time values and the next nine are noisy values of the function. The noises are generated as follow:

```
scl=logspace(-1,-3,k)
n=dot(randn(N,k),diag(scl))
```
After loading *fitting.dat*, extraction of time and data is in the following way
```
data = np.loadtxt("fitting.dat",dtype=float)
t = np.array(data[:,0])
d = np.array(data[:,1:])
```

3 Defining the function

The function required to be implemented is defined in the following way

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

The figure0 plot asked in the question can be generated by adding the noises to the function defined above, the python code snippet is

```
for i in range(9):
plot(t,d[:,i],label = r'$\sigma$=%.3f'%sigma[i])
y0 = g(t,1.05,-0.105)
```

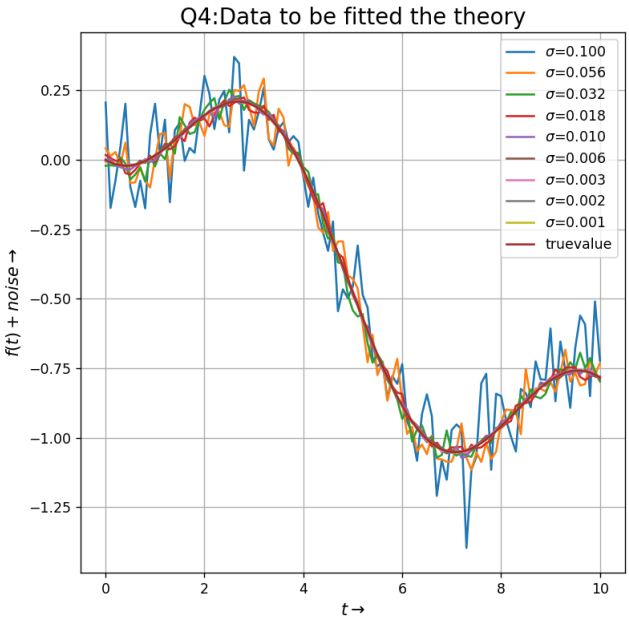

Figure 2: Figure0

4 The errorbar plot

Errorbar plot is an easy way of visualizing the noises present in the measurement. The first column of the data is plotted with errorbars. For readability every 5th item is plotted. The code snippet used for plotting is shown below

```
plot(t,y0,color = 'black',label="f(t)")
errorbar(t[::5],d[::5,0],sigma[0],fmt='ro',label ="Errorbar")
```

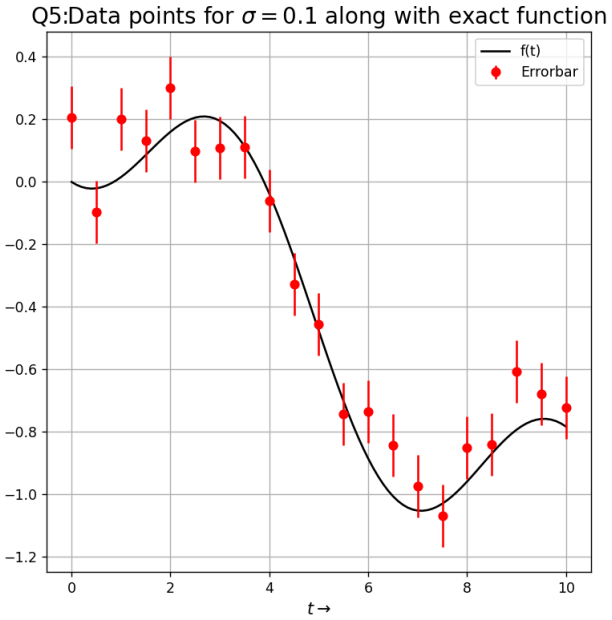

Figure 3: Errorbar Plot

5 Matrix Generation

The truevalue function can also be obtained by a coloumn vector by creating a matrix equation. The matrix M created is multiplied by matrix (A,B) and substituting A = 1.05 and B = -0.105, and the matrices can be compared using the following code snippet

```
J2 = sp.jv(2,t)
M = c_[J2,t]
P = np.array([1.05,-0.105])
y = np.dot(M,P)
equi = y-y0
#due to some precision errors it may happen that few values go very close to zero
```

```
#but not zero
if all(equi<=(1e-15)):
    print("Both the solutions are equal")
else:
    print("Both the solutions are not equal")
```

6 Mean squared Error calculation

The mean squared error between the fk and assumed model can be claculated by the formula

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

The python code snippet for mean squared error calculation is:

```
A = np.linspace(0,2,21)
B = np.linspace(-0.2,0,21)
epsilon = np.zeros((21,21),float)
for i in range(21):
    for j in range(21):
        epsilon[i][j] = np.mean(np.square(y0 - g(t,A[i],B[j])))
```

The contour Plot for ε is

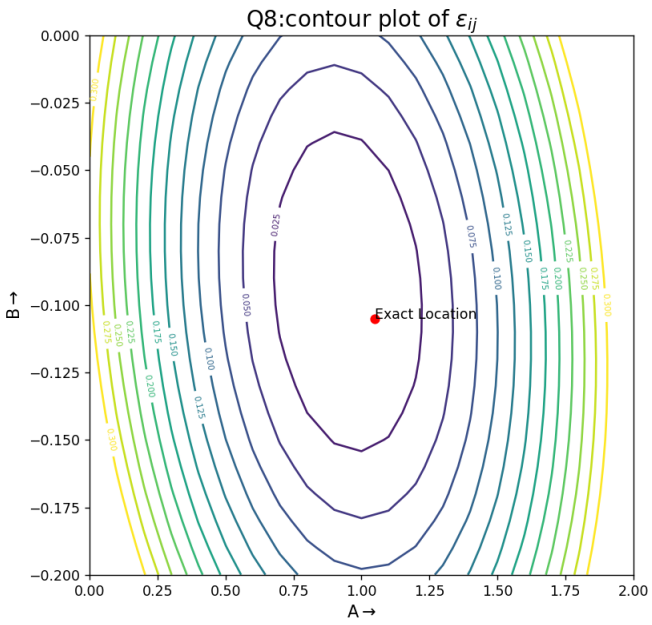

Figure 4: Contour Plot

7 Least Squares Program

The value of the paramters A and B corresponding to the least mean square error is the best estimate that can model the data and is found using the following python code snippet:

```
Ex = np.empty((9,1))
Ey = np.empty((9,1))
for j in range(9):
AB = linalg.lstsq(M,data[:,j+1],rcond=None)
Ex[j] = abs(AB[0][0]-P[0])
Ey[j] = abs(AB[0][1]-P[1])
```

The np.linalg.lstsq function returns the least-squares solution to a linear matrix equation. It is seen that this error is in a non-linear fashion with respect to standard deviation of noise. Plotting the error on log-log scale gives

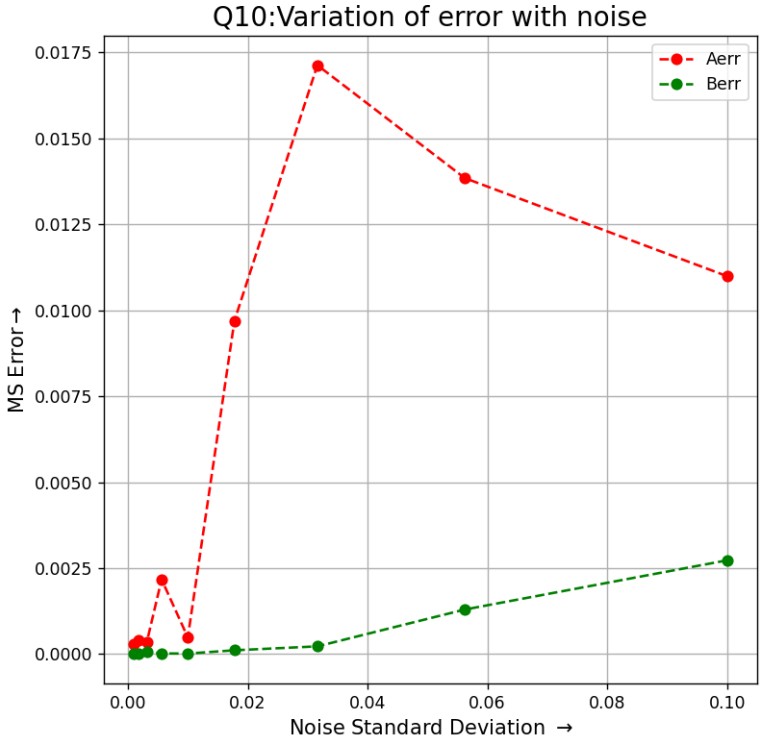

Figure 5: Variation of Error

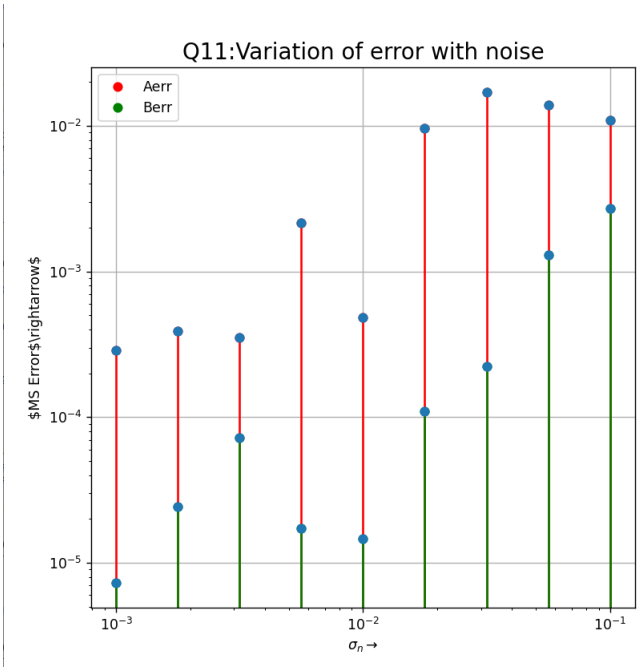

Figure 6: Variation of Error on log scale

8 Conclusion

A best possible estimate for modal parameters are found by minimising the least mean square error, it can be seen that mean square error is directly proportional to σ(standard deviation) of noise in logscale