Indian Institute of Technology, Madras
Department of Electrical Engineering
Applied Programming Lab

Lab Report
Assignment 3

Fitting Data to Models

Nithin Uppalapati
EE18B035

11-02-2020

Contents

1 Abstract **1**

2 Introduction **1**

3 Theory **2**

3.1 Generating the Data . . . 2

3.1.1 Generation of Noise . . . 2

3.1.2 Data File . . . 2

3.2 Analysing the Generated Data . . . 3

3.3 Minimising the Error . . . 4

3.3.1 Mean Squared Error Method . . . 4

3.3.2 Least Squares Method . . . 4

3.4 Measure of Deviation from the Actual Data: . . . 5

3.4.1 Standard Deviation of Noise in the Data . . . 5

3.4.1.1 Actual Data with the noise added . . . 5

3.4.2 Plotting Errorbars along with the Exact Function . . . 5

3.4.2.1 Error Bar of Noisy Bessel Function relative to exact Bessel Function . . . 6

3.4.3 Plotting Errorbars in Variation of Error of Coefficients with noise . . . 6

3.4.3.1 Variation of A_{err} and B_{err} with σ_n [Linear plot] . . . 7

3.4.3.2 Variation of A_{err} and B_{err} with σ_n [loglog plot] . . . 7

3.4.4 Calculating the Mean Squared Error . . . 8

3.4.4.1 Contour Plot of Noisy Bessel Function with varying A and B . . . 8

3.5 The Optimal Coefficients . . . 8

4 Conclusions **9**

4.1 On the Contour Plot of ϵ_{ij} . . . 9

4.2 On the Linearity of Error (of $A's$ and $B's$) with the Standard Deviation . . . 9

1. Abstract

Aim :
The aim of this project is to accurately fit the given data (measured data) to some specified functions. Specifically, in this assignment, we are fitting the data to a Bessel function of order 2 and a first order polynomial, both of them as a functions of time(t)

Implementation :
So, the method of prediction is as follows, We proceed empirically, by assuming a set of valid range coefficients for the function and calculate the output of this 'trial function', and then we calculate the error of this 'trial function' with the actual output, by mean squared error method. Thus, we can predict the coefficients, by minimising the error.

2. Introduction

Assignment 3 is based on

- Analysing the data to extract information by plotting graphs
- Study the effect of noise on the fitting process using scipy, pylab libraries in python.
- Plotting graphs using the pylab library

3. Theory

This assignment, is provided with a program named - "*generatedata.py*" which creates the data to be fitted to the desired function, by adding a noise to the output of the actual function in the given range of time samples.

3.1 Generating the Data

3.1.1 Generation of Noise

We are producing the data from the given model, and adding a *randomly generated* noise to it. The program ensures that the standard deviation of the noise is in the range of (0.001 0.1) and the standard deviation is chosen randomly and then is added to the actual data for the given range of time samples...

3.1.2 Data File

The generated data should be collected and parsed from the file named "*fitting.dat*", for collection of the "measured" data (data with noise) and the time samples. The data file consists of 10 columns. The first column is time, while the remaining columns are data. The data corresponds to the function : $f(t) = 1.05J(2,t) - 0.105t + n(t)$

Where J(2, t) is Bessel function of order 2, imported from scipy.special library.

The data can be collected from *fitting.dat* in this way, as shown in the segment below:

```
 f=open("fitting.dat")
dat=f.readlines()
N=len(dat) # N=101, in the fit.data
k=len(dat[0].strip().split(' ')) #A=1.05
data=array([zeros(k)]*N) #B=-0.105

for i in range(0,N):
dat[i]=dat[i].strip()
```

```
dat[i]=dat[i].split(' ')
for j in range(0,k):
data[i][j]=float(dat[i][j])
t=data[:,0]
y=g(t,A,B)

y_data=array([zeros(N)]*(k-1))
stdev=array(zeros(k-1))
for i in range(0,k-1):
y_data[i]=data[:,i+1]
stdev[i]=std(g(t,A,B)-y_data[i])
```

where

```
t=data[:,0]  ###corresponds to the first column in the data-array
```

3.2 Analysing the Generated Data

So, once the data and the time samples are collected, we store them in arrays of suitable sizes.

$$t = \begin{bmatrix} t_0 \\ t_1 \\ t_2 \\ \vdots \\ t_{100} \end{bmatrix} \tag{3.1}$$

and the measured data(remaining columns other than the first one) matrix in the form:

$$y\_data = \begin{bmatrix} y_{0,1} & y_{0,2} & \cdots & y_{0,9} \\ y_{0,2} & y_{02} & \cdots & y_{0,9} \\ \vdots & \vdots & \ddots & \vdots \\ y_{100,n} & y_{100,2} & \cdots & y_{100,9} \end{bmatrix} \tag{3.2}$$

Now, create a matrix, with each column being a functional values (to the given time samples) to which the data to be fitted.

$$F = \begin{bmatrix} J(2,t) & t \end{bmatrix} \tag{3.3}$$

And also let us create the column matrix of the actual coefficients,

$$Coeff\_exact = \begin{bmatrix} A & B \end{bmatrix} \tag{3.4}$$

Where $A = 1.05$ and $B = -0.105$ defined above So, now we multiply this **functional values matrix** with a **guess matrix** of coefficients (where the coefficients are varied in the vicinity of the actual coefficients) and then the output of this multiplication is the predicted data, which should be compared with the measured data, and the error should be stored into an array.

And the guess coefficients matrix is generated by using:

```
A_guess=linspace(0,2,21)
B_guess=linspace(-0.2,0,21)
```

3.3 Minimising the Error

3.3.1 Mean Squared Error Method

If a vector of n predictions generated from a sample of n data points on all variables, and Y is the vector of observed values of the variable being predicted, with $\hat{Y_i}$ being the predicted values (e.g. as from a least-squares fit), then the within-sample MSE of the predictor is computed as

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y_i})^2$$

3.3.2 Least Squares Method

The method of least squares is a standard approach in regression analysis to approximate the solution of overdetermined systems (sets of equations in which there are more equations than unknowns) by minimizing the sum of the squares of the residuals made in the results of every single equation. The most important application is in data fitting. The best fit in the least-squares sense minimizes the sum of squared residuals (a residual being: the difference between an observed value, and the fitted value provided by a model).

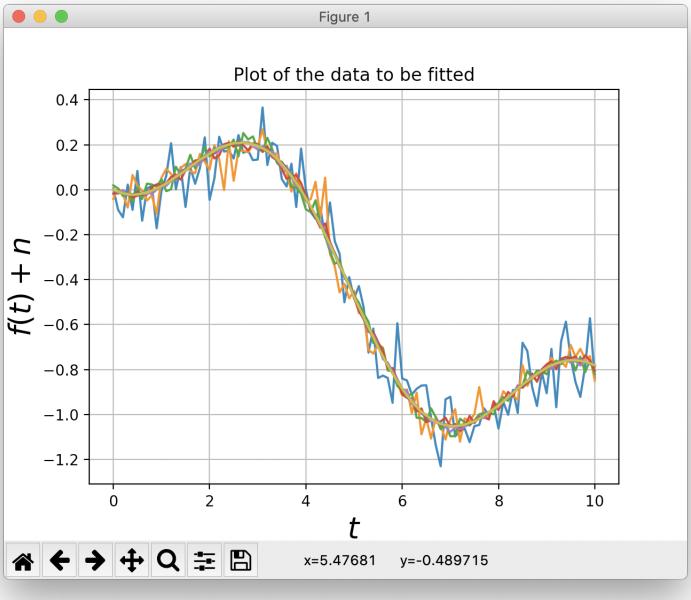
So, as the coefficients are varied to minimize the error, we differentiate the error wrt. each coefficient and equate it to zero, in order to obtain the vector of optimal coefficients ($p_i's$).

And in python, in scipy.linalg library, there is a dedicated function, named $lstsq()$ [leastsqares], which accepts the arguments as : $lstsq(F, x)$

3.4 Measure of Deviation from the Actual Data:

3.4.1 Standard Deviation of Noise in the Data

3.4.1.1 Actual Data with the noise added



The Standard deviation of the noise present in the data can be extacted by the inbuilt function,

```
std(y\_data - g(t,A,B))
```

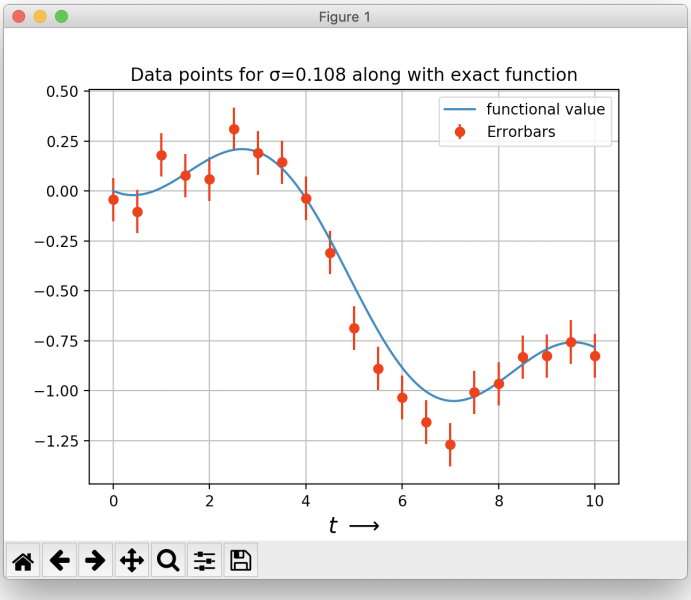
and the standard deviation of the noise in each column is updated as follows:

```
for i in range(0,k-1):
y_data[i]=data[:,i+1]
stdev[i]=std(g(t,A,B)-y_data[i])
```

3.4.2 Plotting Errorbars along with the Exact Function

```
errorbar(t[::5],y_data[i][::5],label="Errorbars",yerr=stdev[0],fmt="ro")
plot(t,y,label= "Exact function")
xlabel('noise standard deviation  ''$\longrightarrow$',size=15)
ylabel('A and B error  ''$\longrightarrow$',size=15)
```

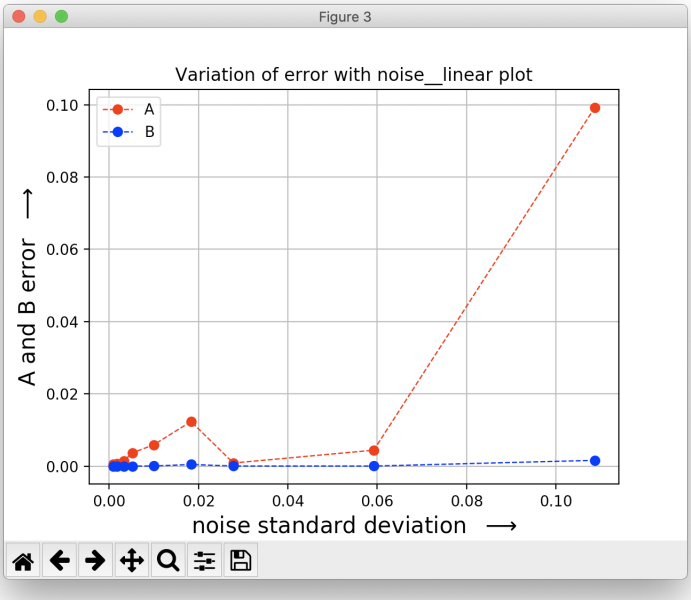
3.4.2.1 Error Bar of Noisy Bessel Function relative to exact Bessel Function



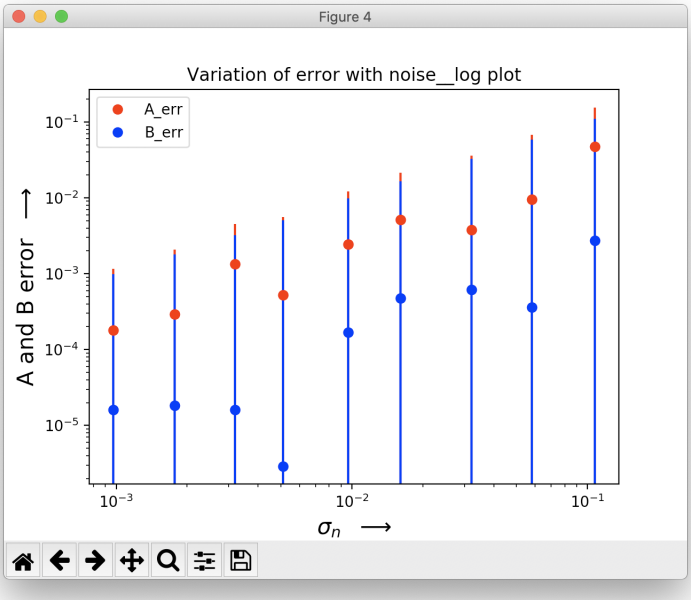
3.4.3 Plotting Errorbars in Variation of Error of Coefficients with noise

```
errorbar(stdev,abs(A-predict_coeff[0]),yerr=std(A-predict_coeff[0]))
loglog(stdev,abs(A-predict_coeff[0]))
errorbar(stdev,abs(B-predict_coeff[1]),yerr=std(B-predict_coeff[1]))
loglog(stdev,abs(B-predict_coeff[1]))
xlabel('$\u03C3_n$' '  ' '$\longrightarrow$',size=15)
ylabel('A and B error  ''$\longrightarrow$',size=15)
title(r'Variation of error with noise__log plot')
```

3.4.3.1 Variation of A_{err} and B_{err} with σ_n [Linear plot]



3.4.3.2 Variation of A_{err} and B_{err} with σ_n [loglog plot]

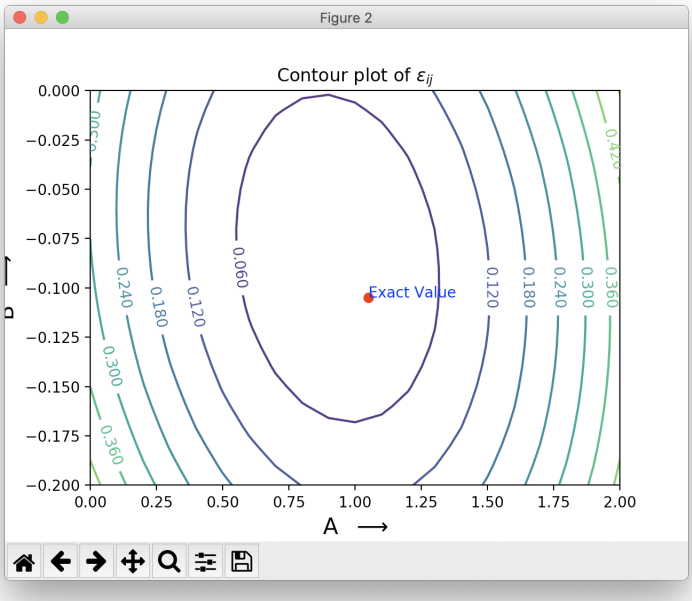


3.4.4 Calculating the Mean Squared Error

```
A_guess=linspace(0,2,21)
B_guess=linspace(-0.2,0,21)
appx_err_val=array([zeros(21)]*21)

for i in range(0,21):
for j in range(0,21):
appx_err_val[i][j] = (square( y_data[0]-g(t,A_guess[i],B_guess[j]) )).mean()
```

3.4.4.1 Contour Plot of Noisy Bessel Function with varying A and B



3.5 The Optimal Coefficients

For calculation of optimal coefficients, I use the lstsq() function for the accurate prediction of coefficients

```
predict_coeff=array([zeros(k-1)]*(2))
for i in range(0,k-1):
p,resid,rank,sig=lstsq(F,y_data[i])
predict_coeff[0][i]=p[0]
predict_coeff[1][i]=p[1]
```

And the coefficients are stored in the arrays $predict\_coeff$ array.

4. Conclusions

4.1 On the Contour Plot of ϵ_{ij}

There is one minima for for every contour of ϵ, and the co-ordinates of that minima are given by the output of lstsq() function.

So, on overall as there are nine readings, there are nine locations of minima of ϵ, which corresponds to the nine pair of accurate coefficients, calculated from the lstsq() function.

4.2 On the Linearity of Error (of $A's$ and $B's$) with the Standard Deviation

The Error in the predicted coefficients with respect to the Actual coefficients is varying linearly with the standard deviation of the noise.