EE2703 : Applied Programming Lab

G.Rohith Kumar
EE18b008

February 11, 2020

Abstract

Reading data from files and parsing them.
1) Analysing the data to extract information
2) Study the effect of noise on the fitting process
3) Plotting graphs

Introduction

Perhaps the most common engineering use of a computer is the modelling of realdata. Thatistosay,some device, say a tachometer on an induction motor, or a temperature sensor of an oven or the current through a photodiode provides us with real-time data. This data is usually digitised very early on in the acquisition process to preserve the information, leaving us with time sequences,

$$(t, x) = \{t_i, x_i\}_{i=1}^{N} \tag{1}$$

If the device has been well engineered, and if the sensor is to be useful in diagnosing and controlling the device, we must also have a model for the acquired data:

$$f(t; p_1, p_2, ..., p_N) \tag{2}$$

For now we are only dealing with Models which are linear in parameters

$$f(t; p_1, p_2, ..., p_N) = \sum_{i=1}^{\infty} p_i F_i(t) \tag{3}$$

Clearly the general problem reduces to the inversion of a matrix problem. However, the number of parameters, N, is usually far less than the number of observations, M. In the absence of measurement errors and noise, any non-singular N X N submatrix can be used to determine the coefficients p$_i$.

We wish to get the "best" guess for $\overrightarrow{p}$. For us, this means that we need to minimize the L2 norm of the error. The error is given by

$$\epsilon = F.\vec{p} - \vec{x} \tag{4}$$

The norm of the error is given by

$$\epsilon^T.\epsilon = \sum_i \epsilon_i^2 \tag{5}$$

Now the matrix M is symmetric (just take the transpose and see). So the equation finally becomes, written as a vector expression

$$\bigtriangledown_{error(\vec{p})} = 2(F^T F)\vec{p_0} - 2F^T \vec{x} \tag{6}$$

$$\vec{p_0} = (F^t F)^{-1} F^T \vec{x} \tag{7}$$

In Python, it is a library function called lstsq:
from scipy.linalg import lstsq
p, resid, rank, sig= lstsq(F,x)
where p returns the best fit, and sig, resid and rank return information about the process. In Scilab or Matlab, it take the form:
p0 =inv(F'*F)*F'*x;

We have real data with each having different noise which are stored in a file named fitting.dat as columns

```
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
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)
savetxt("fitting.dat",c_[t,yy]) # write out matrix to file named fitting.dat
```

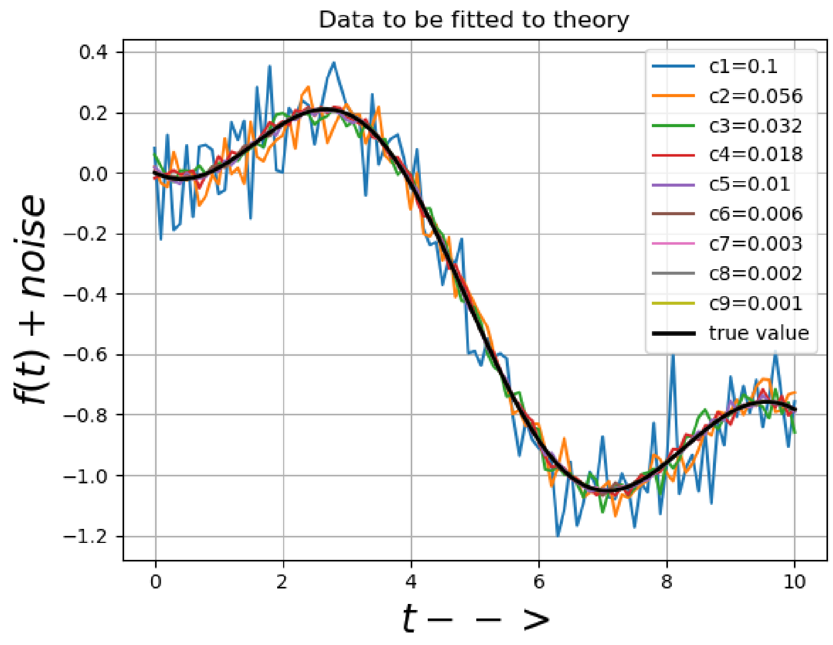
Q3 and Q4: Data to be fitted

CODE:

```
for i in range(1,10):
    plot(f1[0],f1[i],label='c'+str(i)+'='+str(round(sigma[i-1],3)))
    legend(loc='upper right')
y=1.05*sp.jn(2,t)-0.105*t
plot(t,y,color='black',linewidth=2,label='true value')
legend(loc='upper right')
```

Equations:

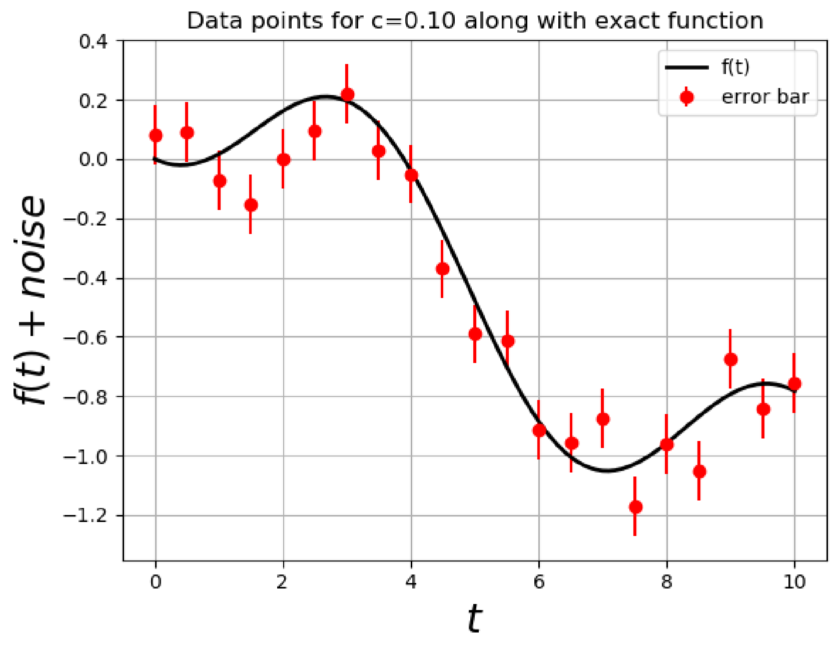
$$f(t) = 1.05J_2(t) - 0.105t g(t; A, B) = AJ_2(t) + Bt \tag{8}$$



Q5

CODE:

```
errorbar(t[::5],f1[1][::5],sigma[0],fmt='ro',label='error bar')
plot(t,y,color='black',linewidth=2,label='f(t)')
legend(loc='upper right')
```



Q6:Verification

CODE:

```
M=array([[sp.jn(2,i),i] for i in t])
p=array(c_[[a,b]])
d=dot(M,p)
y=array([[y[i]] for i in range(len(y))])
if(d.all()==y.all()):
    print('They are EQUAL')
```

Equations

$$g(t;A,B) = \begin{pmatrix} J_2(t_1) & t_1 \\ J_2(t_2) & t_2 \\ \dots & \dots \\ J_2(t_m) & t_m \end{pmatrix} \begin{pmatrix} A \\ B \end{pmatrix} = M.p \tag{9}$$

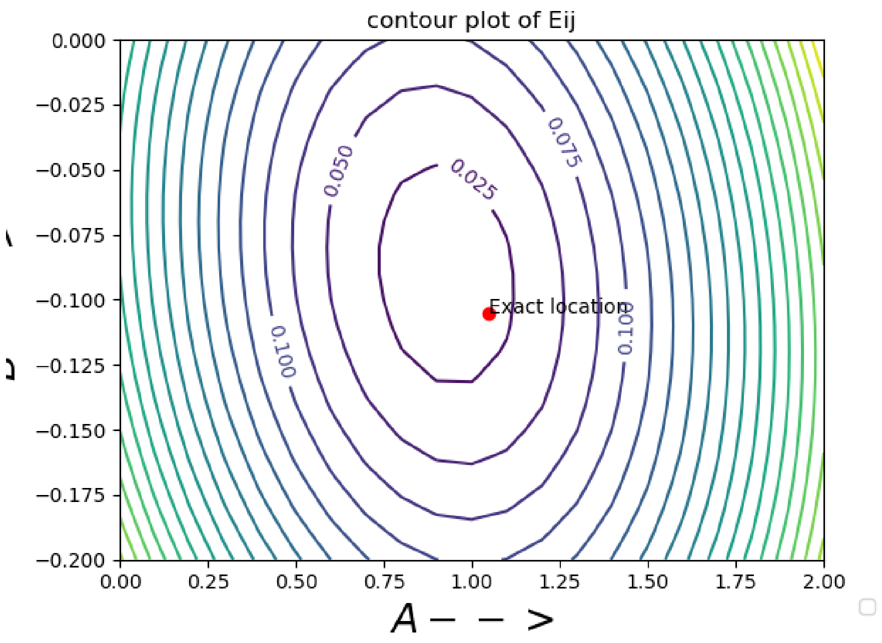
Q7&Q8:ϵ_{ij}, Evaluating and Plotting its Contour

```
A=linspace(0,2,21) # A array
B=linspace(-0.2,0,21) # B array
```

```
e=[[] for i in range(len(A))]
for i in range(len(A)):
    for j in range(len(B)):
        v=0
        for k in range(len(t)):
            v=v+pow((f1[1][k]-(A[i]*sp.jn(2,t[k])+B[j]*t[k])),2)
        v=v/101
        e[i].append(v)
A,B=meshgrid(A,B)
c=contour(A,B,array(e),20)
clabel(c,c.levels[:5],inline=True,fontsize=10)
plot(a,b,color='red',marker='o')
text(a,b,'Exact location')
legend(loc=(1.05,-0.105))
```

Equations:

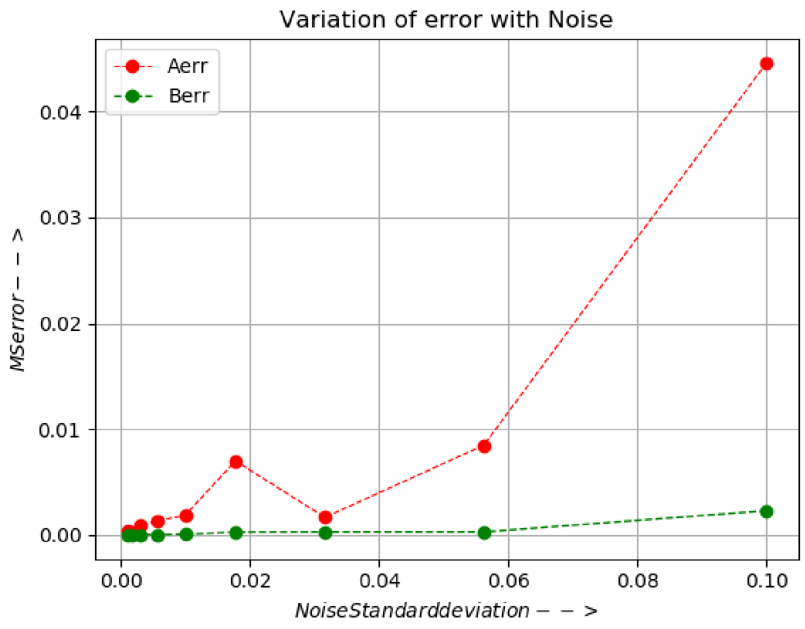
$$\epsilon_{ij} = \frac{1}{101}\sum_{k=0}^{101}(f_k - g(t_k, A_i, B_j))^2 \tag{10}$$



Q9&Q10:Solving p and M.p=g(t;A,B) and Plotting Errors in A and B

CODE:

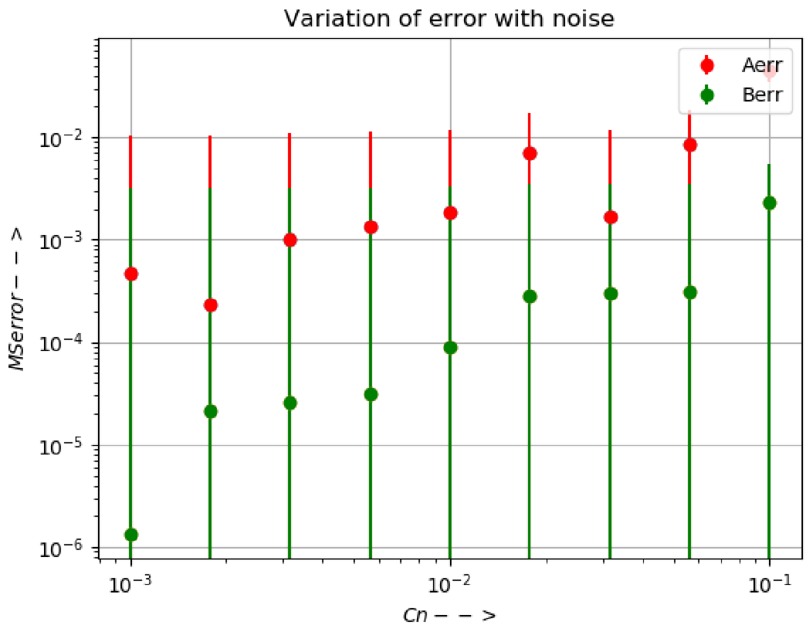
```
p=[]
for i in range(9):
    p.append(lstsq(M,f1[i+1]))
Aerr=[]
Berr=[]
for i in range(len(p)):
    Aerr.append(abs(p[i][0][0]-1.05))
    Berr.append(abs(p[i][0][1]+0.105))
plot(sigma,Aerr,'r--o',linewidth='0.8',label='Aerr')
plot(sigma,Berr,'g--o',linewidth='1',label='Berr')
legend(loc='upper left')
```



Q11:Plotting the Errorbars of A and B with loglog

CODE:

```
loglog(sigma,Aerr,'o')
loglog(sigma,Berr,'o')
errorbar(sigma,Aerr,sigma[4],fmt='ro',label='Aerr')
errorbar(sigma,Berr,sigma[6],fmt='go',label='Berr')
legend(loc='upper right')
```



Conclusion

Plotting curves with noise and legends and errorbars. Plotting contours On A vs B axes.Finding the best estimate value of A and B using lstsq(M,x). Finding errors in A and B and plotting them vs sigma(stdev noise).