Assignment 3: Applied Programming Lab

Saurabh Vaishampayan, EP17B028

11 February 2020

1 Abstract

The goal of the assignment is to learn the theory of Linear Least squares fitting, conditions for applicability; followed by application in python. The data given to us is a pure signal added with white gaussian noise. We study the effects of severity of the added noise on estimation of parameters.

2 Introduction

Given a signal(or a function), we wish to extract the parameters for the theoretic model we are trying to explain the behaviour with. Since real life data is noisy, our data will almost never fit exactly with the model for any value of the parameters. If the noise is assumed to be uncorrelated, parameters can be extracted by linear least squares, ie finding the parameters st th energy of the difference signal is minimum. In this assignment we are to implement the following tasks:

1. Import the dat file, extract the data
2. Plot the data corresponding to different noise levels, also plot errorbar graph
3. Understand least squares, calculate and plot the mean squares error for different values of A and B. Understand the behaviour of function from contour plot
4. Implement least squares, find estimates and errors.
5. Plot errors in estimates as a function of noise. Try and identify the relationship.

3 Question 1

This involves executing the file given to create a dat file of signal added with varying amounts of noise. We import the dat file into our main program and perform operations accordingly.

4 Question 2

4.1 Description

Load fitting.dat and extract the data. The first column is time while remaining columns are data, which are extracted and stored accordingly

4.2 Codes

```
import numpy as np
import scipy.special as sp
import matplotlib.pyplot as plt
from scipy.linalg import lstsq

#Load the dat file into dat_array
dat_array = np.loadtxt('fitting.dat')

arraystdev = np.logspace(-1,-3,9)
list_labels = [None]*9

for i in range(9):
    list_labels[i] = str(arraystdev[i])

t = np.zeros(101)
for i in range(101):
    t[i] = dat_array[i][0]

#Create column vectors for data of each of the 9 instances
data_matrix = np.zeros((9,101))
for k in range(1,10):
    for i in range(101):
        data_matrix[k-1][i] = dat_array[i][k]
```

We have extracted the data from the file now for further use

5 Question 3 and 4

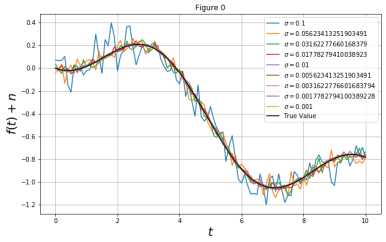
5.1 Description

Create an array containing values of different standard deviations chosen with equal spacing(logarithmically) Create a function to calculate $AJ_2(x) + bx$ Plot the data along with the pure signal(given by computing the function at the true value), with appropriate labels

5.2 Codes

```
#Plot the raw data with labels
fig1, ax1 = plt.subplots(figsize=(10,6))
for i in range(9):
    ax1.plot(t,data_matrix[i],label=r'$\sigma = $'+list_labels[i])

ax1.set_xlabel(r'$t$',size=20)
ax1.set_ylabel(r'$f(t)+n$',size=20)
ax1.set_title(r'Figure 0')
ax1.legend()
ax1.grid(True)

#Function definition of g
def g(x, a, b):
    y=a*sp.jn(2,t)+b*t
    return y
#Definition of true values
A_true = 1.05
B_true = -0.105

#Add the pure signal to the earlier plot
trueval = g(t,A_true,B_true)
ax1.plot()
plt.plot(t,trueval,'k-',label='True Value')
plt.xlabel(r'$t$',size=20)
plt.ylabel(r'$f(t)+n$',size=20)
plt.title(r'Figure 0')
plt.legend()
plt.grid(True)
plt.show()
```

5.3 Plots



Plot for Q3 and Q4

6 Question 5

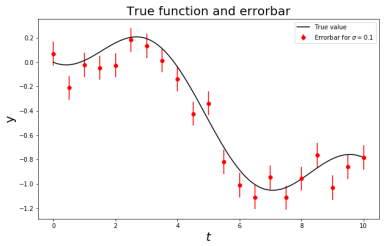
6.1 Description

To plot the first column of the data with errorbars. Every 5th data item is plotted for easy viewing. The noise is taken to be σ_1. After this the exact curve is plotted alongside with proper annotations

6.2 Code

```
#Plot data of first column with errorbars along with true function
plt.figure(figsize=(10,6))
plt.plot(t,trueval,'k-',label='True value')
plt.errorbar(t[::5], data_matrix[0][::5],0.1,fmt='ro',label='Errorbar for '+r'$\
plt.legend()
plt.xlabel(r'$t$',size=20)
plt.ylabel('y',size=20)
plt.title('True function and errorbar',size=20)
plt.savefig('Errorbar.png')
plt.show()
```

6.3 Plots



Plot for Q5

7 Question 6

7.1 Description

Constructing the matrix M st Mp = g(t,A,B) where p = [A,B]. This will be used later for least squares fitting

7.2 Code

```
#Create matrix M in the format asked.
#This is later used for least squares fitting
jn = sp.jn(2,t)
```

```
M = np.c_[jn,t]
p = [A_true,B_true]
g_0 = np.dot(M,p)
```

8 Question 7 and Question 8

8.1 Description

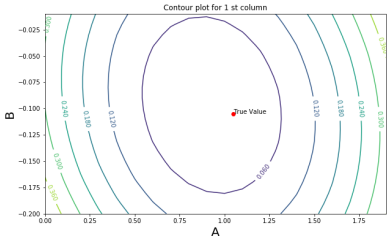
We create a mesh of A and B where A ϵ 0,0.1,...,2 and B ϵ -2,-0.19,...,0
Then we calculate the mean square error given by:
$\epsilon_{ij} = (1/101)(\sum_{k=0}^{100}(f_k - g(t_k, A_i, Bj)^2)$

We plot a contour plot of ϵ vs A and B for the domain mentioned above, and try to observe the characteristics of the function like does it have multiple minima, shape of contours etc

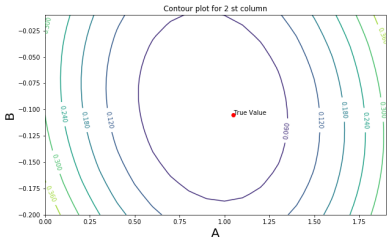
8.2 Code

```
#Meshgrid of A and B for contour plots
A = np.arange(0,2,0.1)
B = np.arange(-0.20,0,0.01)
err = np.zeros((2,len(A),len(B)))
a, b = np.meshgrid(A,B)

#Calculate and contour plot error values for A,B belonging to the mesh
for cols in range(2):
        for i in range(len(A)):
            for k in range(len(B)):
                dummy = g(t,A[i],B[k])
                for w in range(len(t)):
                    err[cols][i][k] += (data_matrix[cols][w]-dummy[w])**2
                err[cols][i][k] = (1/101)*err[cols][i][k]

        plt.figure(figsize=(10,6))
        cp = plt.contour(a,b,err[cols])
        plt.clabel(cp,inline=True,fontsize=10)
        plt.plot(A_true,B_true,'ro')
        plt.annotate('True Value',(A_true,B_true))
        plt.title('Contour plot for %d st column' %(cols+1))
        plt.xlabel('A',size=20)
        plt.ylabel('B',size=20)
        plt.savefig('Contourplots%d' %(cols+1))
        plt.show()
```

8.3 Plots



Plot for first column of data



Plot for 2nd column of data

9 Question 9

9.1 Description

Implement least squares using the array created in question 6.
Estimate optimal values of A and B for different noise levels.
Also estimate errors in A and B which will be plotted in question 10 and 11

9.2 Code

```
#Calculate a estimates and errors in a and b by lstsq
a_estimate = np.zeros(9)
b_estimate = np.zeros(9)
a_err = np.zeros(9)
b_err = np.zeros(9)

for i in range(9):
    a_estimate[i], b_estimate[i] = lstsq(M, data_matrix[i])[0]
    a_err[i] = np.abs(A_true-a_estimate[i])
    b_err[i] = np.abs(B_true-b_estimate[i])
```

10 Question 10 and 11

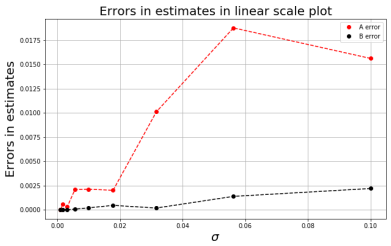
10.1 Description

Plot errors in estimates from the data in Question 9, in linear and log scale and conclude characteristics of linear least squares fitting

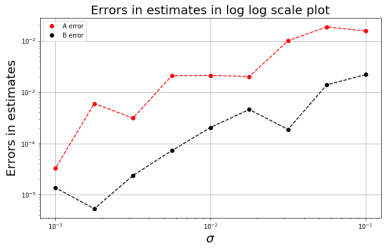
10.2 Code

```
#Linear scale plot of errors in estimate vs stdev
plt.figure(figsize=(10,6))
plt.plot(arraystdev,a_err,'ro',label='A error')
plt.plot(arraystdev,a_err,'r--')
plt.plot(arraystdev,b_err,'ko',label='B error')
plt.plot(arraystdev,b_err,'k--')
plt.ylabel('Errors in estimates',fontsize=20)
plt.xlabel(r'$\sigma$',fontsize=20)
plt.title('Errors in estimates in linear scale plot',size=20)
plt.grid(True)
plt.legend()
plt.savefig('Linear.png')
plt.show()

#Log Log scale plot of errors in estimate vs stdev
plt.figure(figsize=(10,6))
plt.plot(arraystdev,a_err,'ro',label='A error')
plt.plot(arraystdev,a_err,'r--')
plt.plot(arraystdev,b_err,'ko',label='B error')
plt.plot(arraystdev,b_err,'k--')
plt.ylabel('Errors in estimates',fontsize=20)
plt.xlabel(r'$\sigma$',fontsize=20)
plt.yscale('log')
plt.xscale('log')
plt.title('Errors in estimates in log log scale plot',size=20)
plt.grid(True)
plt.legend()
plt.savefig('Loglog.png')
plt.show()
```

10.3 Plots



Plot for Q10 : Errors vs noise, linear scale



Plot for Q10 : Errors vs noise, loglog scale

11 Conclusions

1. The error function has only a single local minimum in the domain of interest. The contour plots also gave insight on how gradient descent is implemented.

2. The errors in estimate were found to be linearly varying(roughly) with the level of noise(standard deviation). Since the added noise is white gaussian, the estimates themselves are random variables, which follow a probability distribution also given by a gaussian which is centred about the true value. Therefore the distribution for MSE is exponential, and hence Mean(Error in estimate) is proportional to σ^2. This agrees well with the simulated results of slope of about 20dB/decade for error in estimate vs stdev of noise