EE2703 : Applied Programming Lab
Assignment 3

Aman Singhal
EP17B002

February 12, 2020

Abstract

This assignment is about data analysing data generated from a given function with 2 parameters. The function contains 2nd degree Bessel function. Noise with varying standard deviation has been added to k copies of the n data points and the data has been provided to us.

Introduction

The function we are working with is the following:
f(t) = A*J2 (t)B*t +n(t)
where the noise(n(t)) is given normally distributed with 0 and different standard deviations varying from 0.1 to 0.001. And J2(t) is the 2nd order bessel function.
The actual parameters are:
A = 1.05
B = -0.105
k = 9
n = 101
standard deviation of noise = [0.1, 0.05623413, 0.03162278, 0.01778279, 0.01, 0.00562341, 0.00316228, 0.00177828, 0.001]

Problem

1) Loading data from fitting.data

2) Plotting curves with noise

3) Plotting the true curve without noise

4) Errorbar plot of the first column of the data.

5) Forming the matrix equation for f(t)

6) Making the mean square error matrix of by taking the following values of A and B and plotting the contour plot of the matrix.

A = 0, 0.1, ..., 2
B = -2, -1.9, ..., 0

7) Estimation of A and B for the given data.

8) Plotting the error in A and B with standard deviation.

0.0.1 Complete Code

```
import matplotlib.pyplot as plt
from pylab import *
import scipy.special as sp
from numpy import *

#function for returning x number of points of function with parameters a and b
def bessel(x,a,b):
    g = a*sp.jn(2,x) + b*x
    return g

#returns the data points of function with parameters a and b through matrix multiplicatio
def vector_bessel(x,a=0,b=0):
    matrix = array([zeros(2) for i in range(n)])
    for i in range(len(x)):
        matrix[i,0] = sp.jn(2,x[i])
        matrix[i,1] = x[i]
    vector = array([[a],[b]])
    return dot(matrix,vector) ,matrix

#function for returning the error matrix for a given range of A and B
def mse(A,B,vector_data):
    mse_matrix = array([zeros(len(A)) for i in range(len(B))])
    for i,a in enumerate(A):
        for j,b in enumerate(B):
            v = vector_data - vector_bessel(t,a,b)[0]
            mse_matrix[i,j] = dot(v.T,v)/n
    return mse_matrix

#function for finding least square estimate from linalg.lstsq
def estimate(A,x):
    est = linalg.lstsq(A, x)
    return est[0]
```

```
#defining parameters requires for working
k = 9       #no. of copies
n = 101     #no. of data points
t = linspace(0,10,n)    #input values for the function
scl=logspace(-1,-3,k)   #std dev values
sigma = 0.1             #std dev of 1st column
coloumn = 0             #1st coloumn

#true_value is the actual values of function without noise a = 1.05 and b = -0.105
true_value = bessel(t,1.05,-0.105)

#loading data as columns from fitting.data
data = loadtxt("fitting.dat",usecols = range(1,10),unpack = True)
vector_data = array([zeros(1) for i in range(n)])
for i,j in enumerate(data[coloumn]):                #coloumn of data
    vector_data[i] = j

#plotting data in fitting.data
for ind,i in enumerate(data):
    plt.plot(t,i,label=r'$\sigma$'+str(ind+1)+" "+str(round(scl[k-ind-1],4)))
#plotting the actual function without noise
plt.plot(t,true_value,label='True Value')
plt.xlabel(r'$t$',size=15)
plt.ylabel(r'$f(t)+n$',size=15)
plt.title(r'Plot of the data')
plt.legend()
plt.show()

#error bar plot
plt.plot(t,true_value)
plt.errorbar(t[::5],data[0][::5],sigma,fmt='ro',label = "Errorbar")
plt.plot(t,true_value,label='True Value')
plt.xlabel(r'$t$',size=15)
plt.title(r'data poinsts for $\sigma$ = 0.1 along with exact function')
plt.legend()
plt.show()

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

```
g = plt.contourf(A,B,mse(A,B,vector_data))
plt.clabel(g, inline=1, fontsize=10)
plt.xlabel(r'$A$',size=15)
plt.ylabel(r'$B$',size=15)
plt.title(r'Contour plot of the error matrix')
plt.show()

#finding A_estimate and B_estimate
a_est = []
b_est = []
for col in range(9):
    vector_data = array([zeros(1) for i in range(n)])
    for i,j in enumerate(data[col]):                #coloumn of data
        vector_data[i] = j
    a_est.append(estimate(vector_bessel(t)[1], vector_data)[0][0])
    b_est.append(estimate(vector_bessel(t)[1], vector_data)[1][0])

#finding Aerr and Berr
a_error = []
for i in a_est:
    x = abs(i-1.05)
    a_error.append(x)
b_error = []
for i in b_est:
    x = abs(i+0.105)
    b_error.append(x)

#plot for Aerr and Berr
plt.plot(scl,a_error,label='A error')
plt.plot(scl,b_error,label='B error')
plt.xlabel('Noise standard deviation',size=15)
plt.ylabel('MS error',size=15)
plt.title(r'Plot for A_err and B_err')
plt.legend()
plt.show()

#loglog plot for Aerr and Berr
plt.loglog(scl,a_error,basey=10,label='A error')
plt.loglog(scl,b_error,basey=10,label='B error')
plt.xlabel('Noise standard deviation',size=15)
plt.ylabel('MS error',size=15)
plt.title(r'Semilog plot for A_err and B_err')
plt.legend()
plt.show()
```

Output of my code:

After running the above code you will get the following plots:

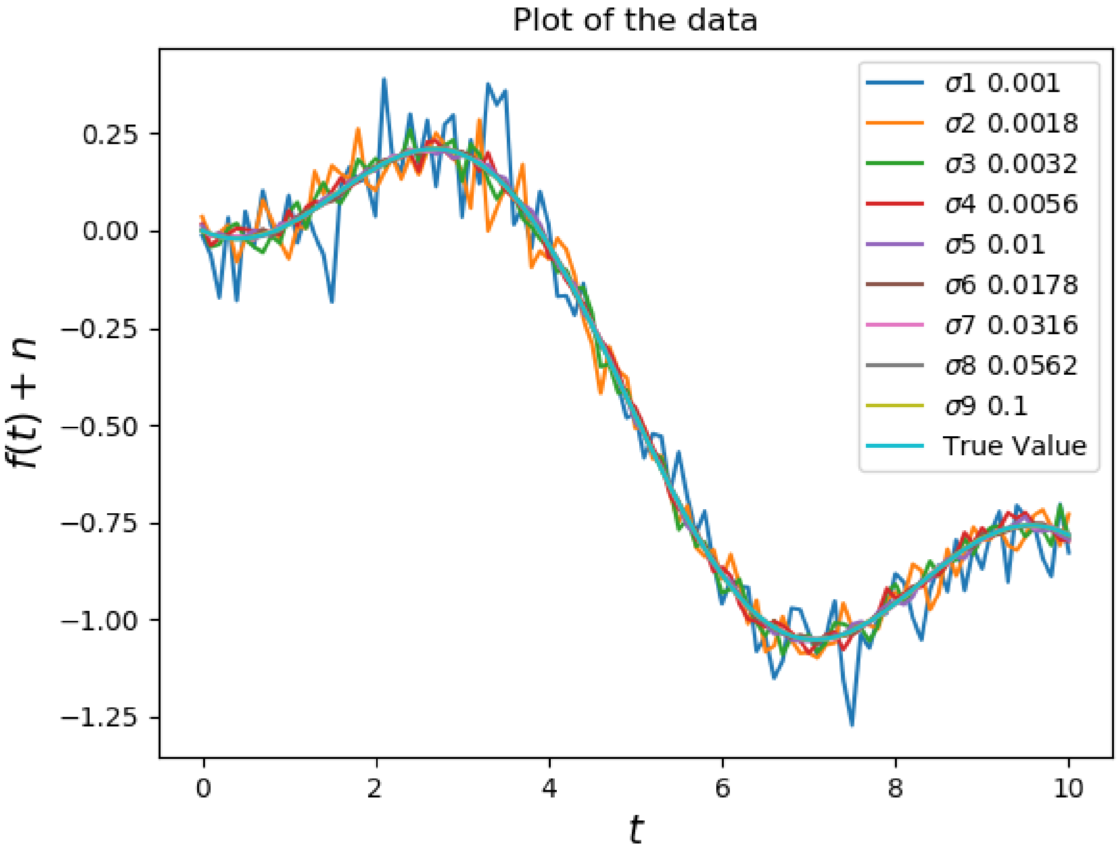

Figure 1: Plot of true function and the given data

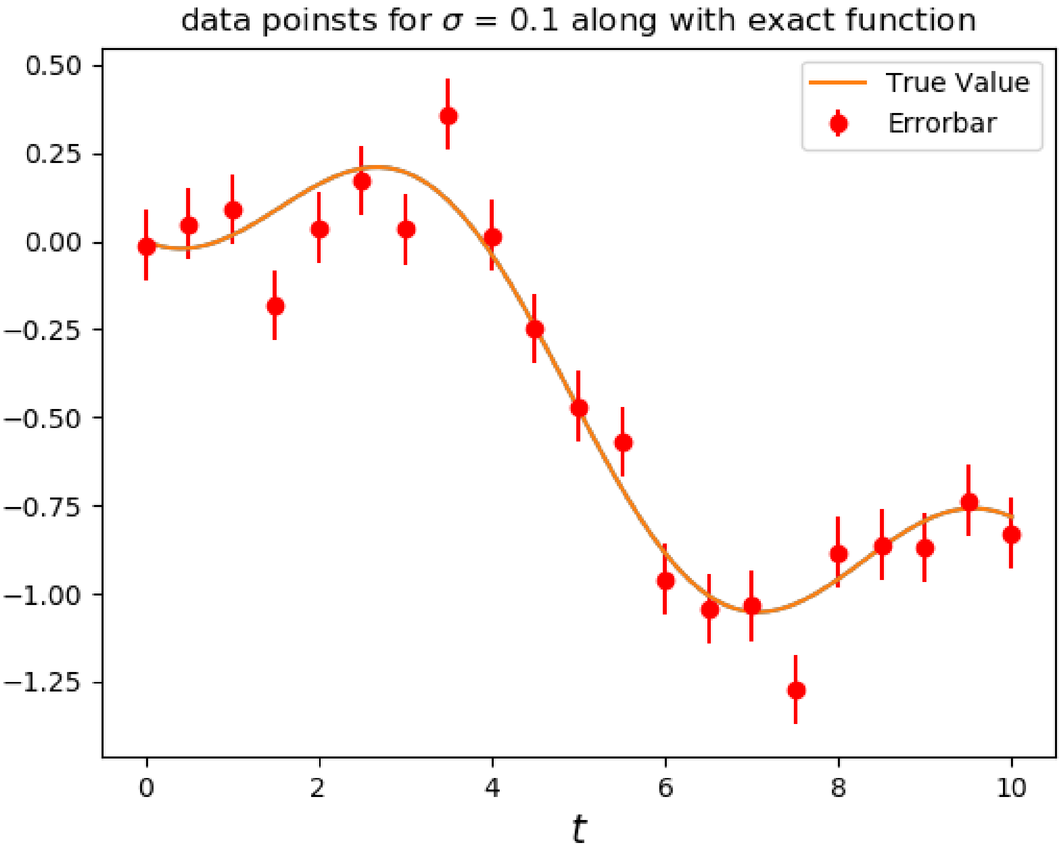

Figure 2: Errorbar graph

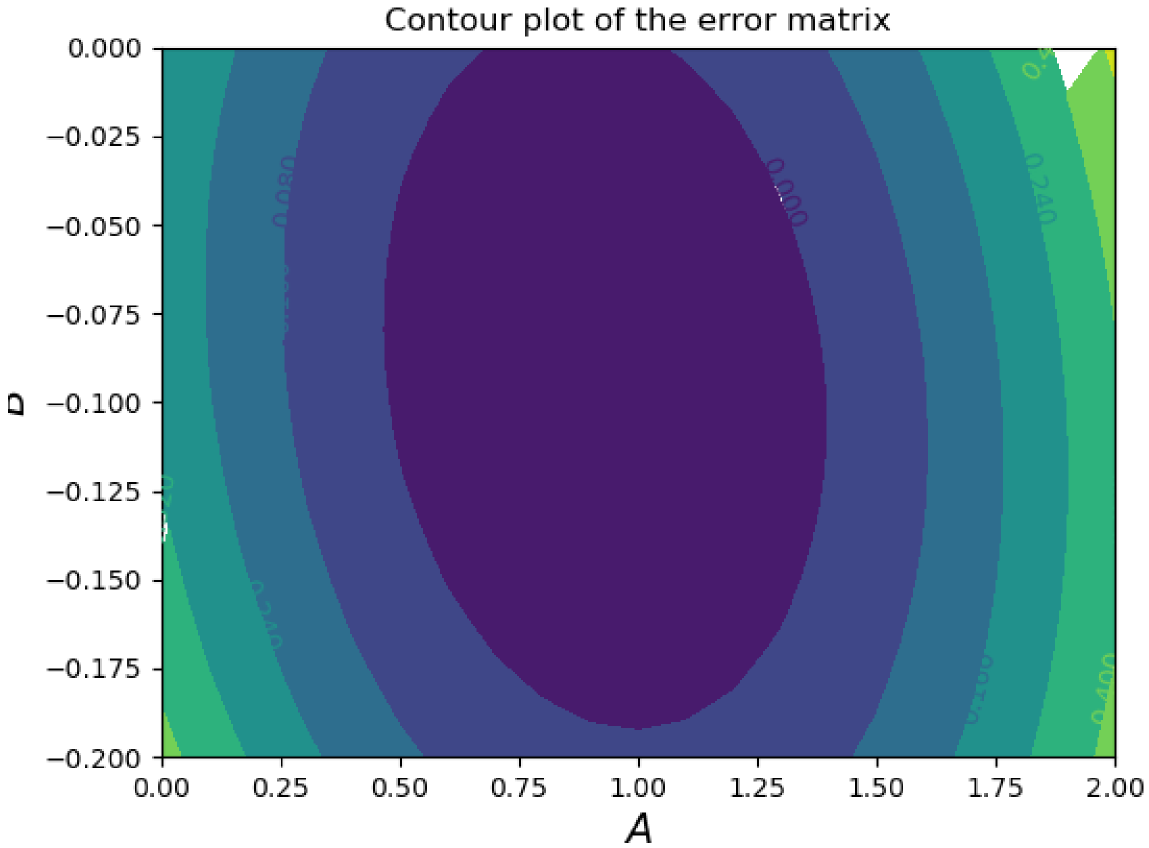

Figure 3: contour plot of error mean square error matrix

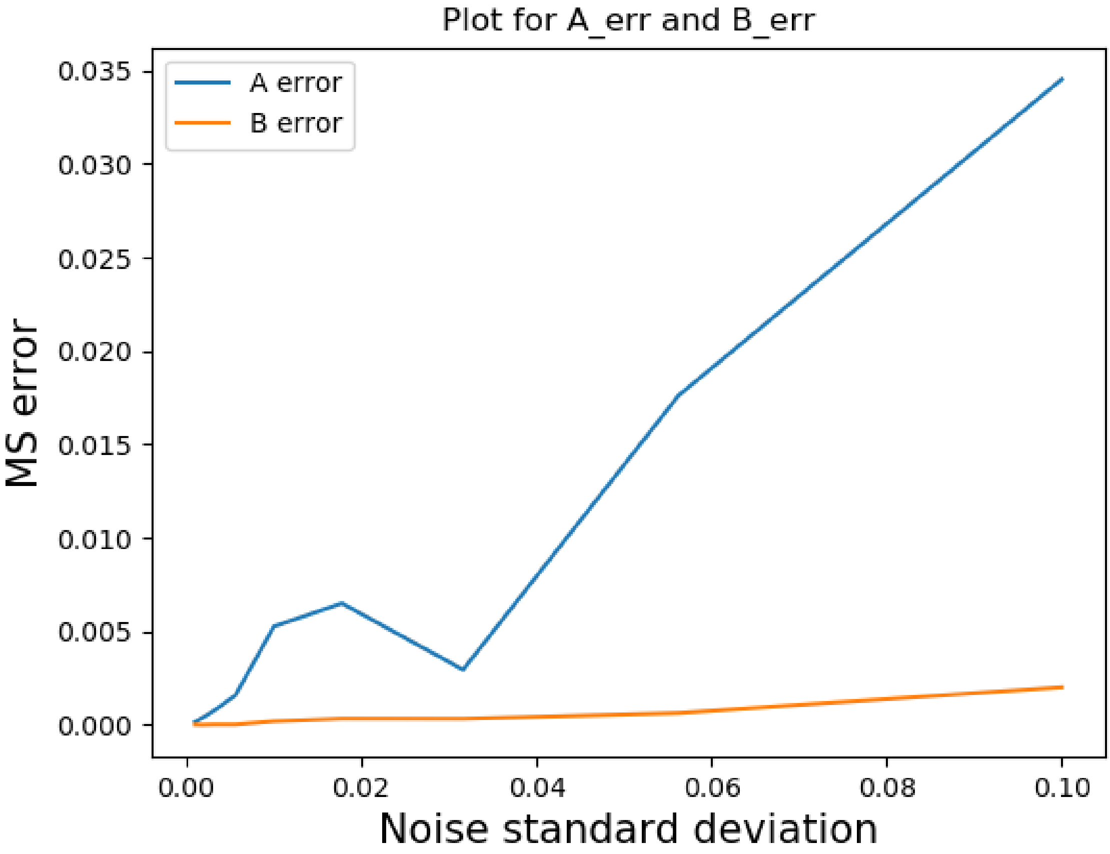

Figure 4: Error in estimates of A and B

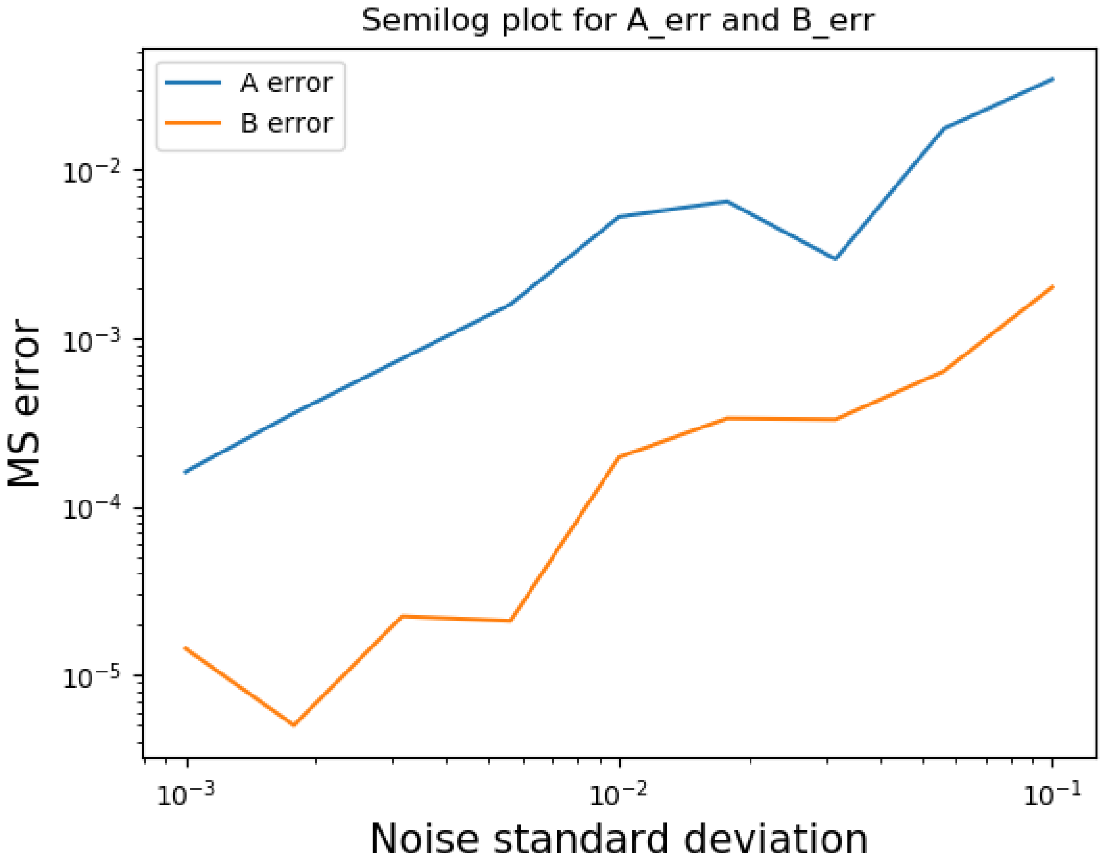

Figure 5: loglog graph for error in estimates of A and B