# Fitting Data

P Harshavardhan EE17B061

February 13, 2019

## 1 functions used

```
these are the important function used in my code
def g(t,A,B):
    return A*jn(2,t)+B*t

def msq(v1):
    return np.square(v1).mean()
# to calculate mean square value of elements in an array

def msqError(fk,trueVal):
    if fk.ndim == 1:
        fk = fk.reshape(len(fk),1)
    if trueVal.ndim == 1:
        trueVal = trueVal.reshape(len(trueVal),1)
    return msq(fk-trueVal)

# to calculate mean square Error of elements in an signal with respect to true value

#for loading data
def loadData():
    data = loadtxt('fitting.dat' if len(argv)==1 else argv[1])
    global t
    global yy
    t,yy = hsplit(data,np.array([1]))# t and yy are vectors t.shape == (N,1)
There is also a \textbf{generateNewData} which does the same thing as generate data
```

I have included every thing needed to plot a fig in a function and called it latter.

```
    if __name__ == '__main__':
    generateNewData()
    loadData()
    N,k = 101,9
    A0,B0=1.05,-0.105
    scl0 = logspace(-1,-3,k)

    M = c_[jn(2,t),t]# both are already column vectors
    p = c_[array([A0,B0])]
    y = g(t,A0,B0)

    plot1()
    plot2()
    verifyMPG(y)# to verify if M.p is approx equal to g())
    Eij_s()    #msq error for different signal noise and
    plot3()
    plot45(*errorEstimation(),*avgErrorEstimation())
    #savefigs()
    show()
```

### 1.1 plot1

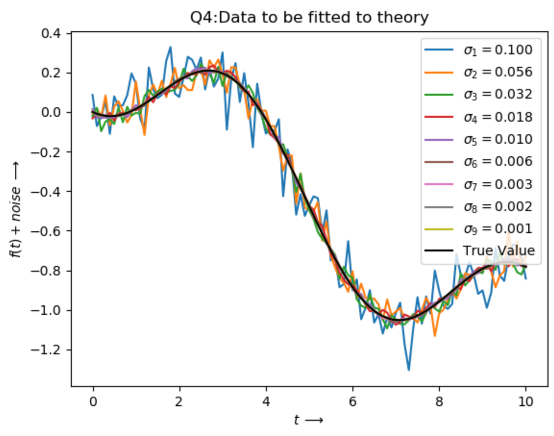
noise added signals from fitting.dat and true value. plotted with legend.

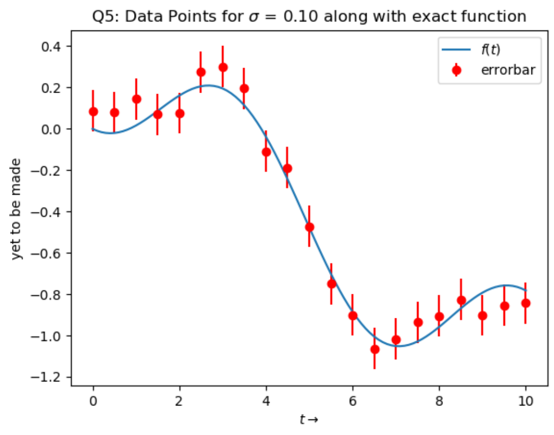
```
 def plot1():
    plt.figure(1)
    plt.plot(t,yy)# t is column matrix
    plt.plot(t,g(t,A0,B0),c='black')
    legendList = [r'$\sigma_{0}={1:0.3f}$'.format(i+1,scl0[i]) for i in range(k)]+[
    plt.legend(legendList,loc='upper right')
    plt.xlabel(r'$t\;\longrightarrow$')
    plt.ylabel(r'$f(t)+noise\;\longrightarrow$')
    plt.title('Q4:Data to be fitted to theory')
```

### 1.2 plot2

Errorbar plot with every fifth point plotted with standard deviation of noise in the signal as error.
the true value of the curve lies within in the error(standard deviation) range of the noise

```
    def plot2():
```





```
plt.figure(2)
plt.errorbar(t[::5],yy[:,0][::5],scl0[0],fmt='ro',label='errorbar')
plt.plot(t,g(t,A0,B0),label=r'$f(t)$')
plt.legend()
plt.xlabel(r'$t\rightarrow$')
plt.ylabel(r'yet to be made')
plt.title(r'Q5: Data Points for $\sigma$ = 0.10 along with exact function')
```

### 1.3 Verify M.p == G

```
M = c_[jn(2,t),t]# both are already column vectors
p = c_[array([A0,B0])]
y = g(t,A0,B0)

def verifyMPG(y,tolerance=1e-8):
```

```
if y.ndim==1:
    y = y.reshape(len(y),1)
if allclose(M.dot(p),y,atol=tolerance):
    print('M.p and y are equal')
else:
    print('Not equal')
```

the above function checks if the product of M and p matrices is equal to g(t,1.05,-0.105)

### 1.4 Calculating Eij for diffent values of A and B

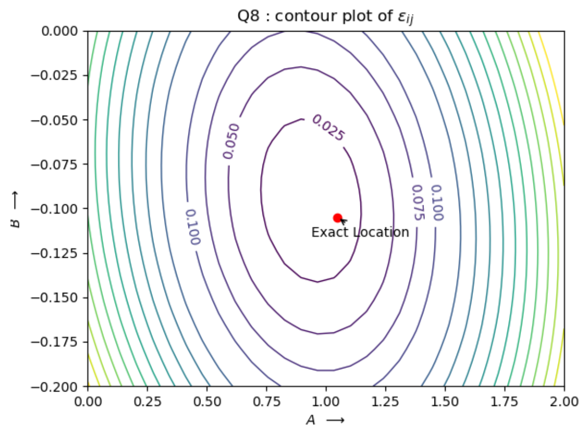
```
def Eij_s():
for fkIndex in (0,1):
    print('column_{}'.format(fkIndex+1))
    for i,j in zip(linspace(0,2,21),linspace(-0.2,0,21)):
        print("A={:0.2f},B={:0.2f},\tE_ij={:0.7f}".format(i,j,msqError(yy[:,fkIn
```

### 1.5 plot3

contour plot for the mean square error from noise signal,and the exact loaction plotted
the contour plot has only one minimum at A0,B0

```
def plot3():
X, Y = meshgrid(linspace(0,2,30),linspace(-0.2,0,30))
# x,y should be same shape
#contour plot for first row(maximum noise data)
Z =np.array([msqError(yy[:,0],g(t,i,j)) for i,j in zip(X.flatten('F'),Y.flatten
plt.figure(3)
plt.scatter(A0,B0,c='r')
plt.annotate(r"Exact Location",xy=(A0,B0),xycoords='data',
    xytext=(-20,-15),textcoords='offset points',
    arrowprops = dict(arrowstyle="->"))

levels = [0] + [0.025*i if i<5 else 0.028*i for i in range(1,16)]
c = plt.contour(X,Y,Z,levels,linewidths=1)#,setting levels for contour ploy
plt.clabel(c,c.levels[:4],inline=1)
plt.title(r'Q8 : contour plot of $\epsilon_{ij}$')
plt.xlabel(r'$A\;\;\longrightarrow$')
plt.ylabel(r'$B\;\;\longrightarrow$')
```



**plot4 and plot5 change every time the code is executed**
but generally error is more for higher standard deviation more info about this in plot 6 and plot 7

```
def errorEstimation():
    estimatedA,estimatedB = zeros(k),zeros(k)
    for i in range(k):
        estimatedA[i],estimatedB[i] = sp.linalg.lstsq(M,yy[:,i])[0]
    return abs(estimatedA-A0),abs(estimatedB-B0)
```

the absolute difference between the lstsq value and the true value is plotted

## 1.6 plot4

absolute error plot for A,B from lstsq estimate

```
    plt.figure(i)
        plt.plot(scl0,errorA,'o--',c='red',label='Aerr')
        plt.plot(scl0,errorB,'--o',c='green',label='Berr')
        plt.title('Q10: Variation of error with noise')
        plt.xlabel(r'$Noise\;standard\;deviation\;\longrightarrow$')
        plt.ylabel(r'$MS\;error\;\;\longrightarrow$')
        plt.legend()
        plt.grid(True)
```

## 1.7 plot5

absolute error plot for A,B from lstsq estimate in loglog scale and with standard deviation of noise as yerr

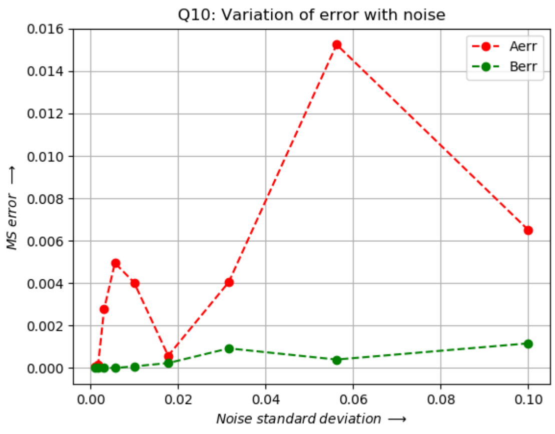


Figure 1: plot4

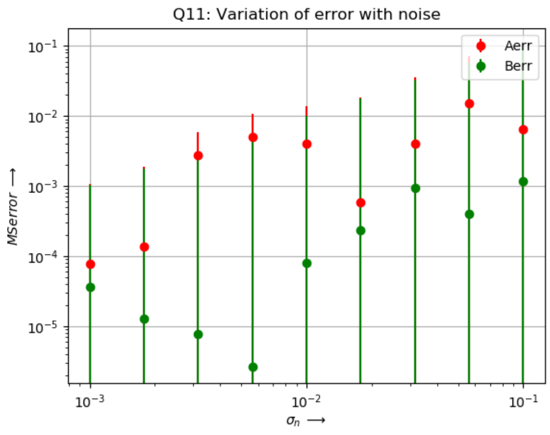
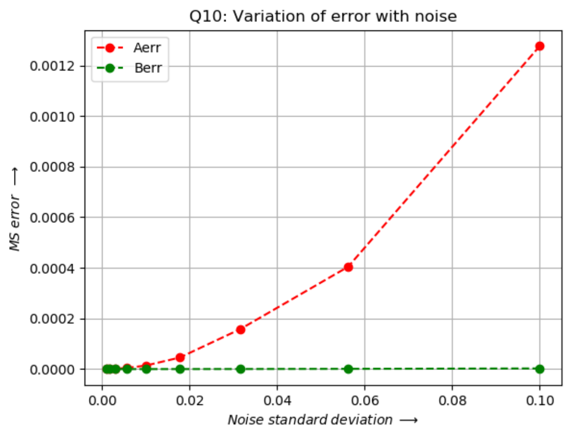


Figure 2: plot5

the two plots below are generated by refreshing fitting.dat 100 times and then plotting the mean square error for A,B(from 100 testcases) for each noise signal

## 1.8 plot6

mean square error plot for A,B from lstsq estimate in loglog scale and with standard deviation of noise as yerr



## 1.9 plot7

```
plt.figure(5)
loglog()
errorbar(scl0,errorA,yerr=scl0,fmt='ro',label='Aerr')
errorbar(scl0,errorB,yerr=scl0,fmt='go',label='Berr')
title('Q11: Variation of error with noise')
xlabel(r'$\sigma_{n}\;\longrightarrow$')
ylabel(r'$MSerror\;\longrightarrow$')
legend()
grid(True)
```

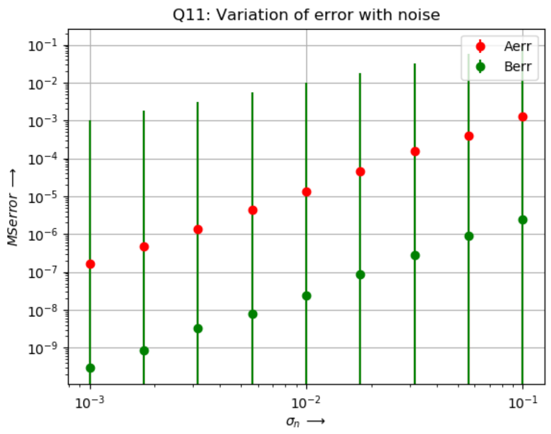
mean square error plot for A,B from lstsq estimate in loglog scale and with standard deviation of noise as yerr

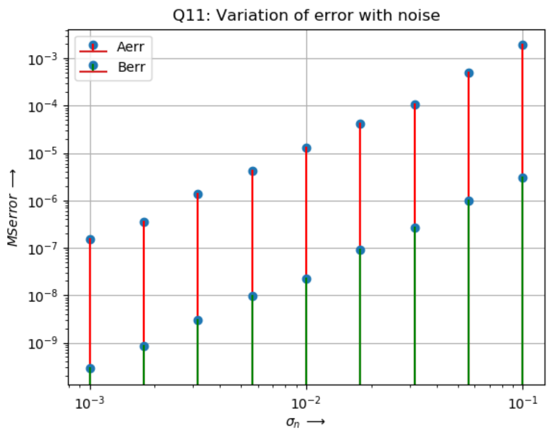
## 1.10 observation

plot4,plot5 varies a lot with each generation of data. but from plot6 and plot8(Mserror from 100 testcases)the error varies almost linearly with standard deviation.

**Code**

```
from pylab import *
from scipy.special import jn
```





```
import scipy as sp
from sys import argv

def generateNewData():
    N,k = 101,9
    t = linspace(0,10,N)
    scl = logspace(-1,-3,k)
    n = dot(randn(N,k),diag(scl))
    g = lambda t,A,B: A*jn(2,t)+B*t
    def msq(fk,t,A,B):
        """fk is column"""
        x = fk-g(t,A,B)
        x.resize((1,len(t)))
        #print(fk.shape,x.shape)
        return dot(x,x.T)/len(t)
    y = g(t,1.05,-0.105)
    Y = meshgrid(y,ones(k),indexing='ij')[0]
    yy = Y+n
    savetxt('fitting.dat',c_[t,yy])

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

def msq(v1):
    return np.square(v1).mean()

def msqError(fk,trueVal):
    if fk.ndim == 1:
        fk = fk.reshape(len(fk),1)
    if trueVal.ndim == 1:
        trueVal = trueVal.reshape(len(trueVal),1)
    return msq(fk-trueVal)

#for loading data
def loadData():
    data = loadtxt('fitting.dat' if len(argv)==1 else argv[1])
    global t
    global yy
    t,yy = hsplit(data,np.array([1]))# t and yy are vectors t.shape == (N

def plot1():
    plt.figure(1)
    # t is column matrix
    plt.plot(t,yy)
    plt.plot(t,g(t,A0,B0),c='black')
    legendList = [r'$\sigma_{0}={1:0.3f}$'.format(i+1,scl0[i]) for i in r
    plt.legend(legendList,loc='upper right')
    plt.xlabel(r'$t\;\longrightarrow$')
    plt.ylabel(r'$f(t)+noise\;\longrightarrow$')
    plt.title('Q4:Data to be fitted to theory')

def plot2():
    plt.figure(2)
    plt.errorbar(t[::5],yy[:,0][::5],scl0[0],fmt='ro',label='errorbar')
    plt.plot(t,g(t,A0,B0),label=r'$f(t)$')
    plt.legend()
    plt.xlabel(r'$t\rightarrow$')
    plt.ylabel(r'yet to be made')
    plt.title(r'Q5: Data Points for $\sigma$ = 0.10 along with exact func
#6
def verifyMPG(y,tolerance=1e-8):
    #print(all(M.dot(p)==y))
    if y.ndim==1:
        y = y.reshape(len(y),1)
    if allclose(M.dot(p),y,atol=tolerance):
        print('M.p␣and␣y␣are␣equal')
    else:
        print('Not␣equal')

#7
def Eij_s():
    for fkIndex in (0,1):
        print('column␣{}'.format(fkIndex+1))
        for i,j in zip(linspace(0,2,21),linspace(-0.2,0,21)):
            print("A={:0.2f},B={:0.2f},\tE_ij={:0.7f}".format(i,j,msqErro

def plot3():
    X, Y = meshgrid(linspace(0,2,30),linspace(-0.2,0,30))
    # x,y should be same shape
    #contour plot for first row(maximum noise data)
    Z =np.array([msqError(yy[:,0],g(t,i,j)) for i,j in zip(X.flatten('F')
    plt.figure(3)
    plt.scatter(A0,B0,c='r')
    plt.annotate(r"Exact␣Location",xy=(A0,B0),xycoords='data',
        xytext=(-20,-15),textcoords='offset␣points',
        arrowprops = dict(arrowstyle="->"))

    levels = [0] + [0.025*i if i<5 else 0.028*i for i in range(1,16)]
    c = plt.contour(X,Y,Z,levels,linewidths=1)#,setting levels for contou
    plt.clabel(c,c.levels[:4],inline=1)
    plt.title(r'Q8␣:␣contour␣plot␣of␣$\epsilon_{ij}$')
    plt.xlabel(r'$A\;\;\longrightarrow$')
    plt.ylabel(r'$B\;\;\longrightarrow$')


#9,10,11
def errorEstimation():
    estimatedA,estimatedB = zeros(k),zeros(k)
    for i in range(k):
        estimatedA[i],estimatedB[i] = sp.linalg.lstsq(M,yy[:,i])[0]
    return abs(estimatedA-A0),abs(estimatedB-B0)
```

```
def avgErrorEstimation(T=25):
    tempA  ,tempB = np.zeros((T,k)),np.zeros((T,k))
    for i in range(T):
        generateNewData();loadData()
        for j in range(k):
            tempA[i,j],tempB[i,j] = sp.linalg.lstsq(M,yy[:,j])[0]
    avgA,avgB = np.zeros(k),np.zeros(k)
    for i in range(k):
        avgA[i],avgB[i] = msq(tempA[:,i]-A0),msq(tempB[:,i]-B0)
    return avgA,avgB

def plot45(errorA,errorB,MSerrorA,MSerrorB):
    def plotABerror(errorA,errorB,i=4):
        plt.figure(i)
        plt.plot(scl0,errorA,'o--',c='red',label='Aerr')
        plt.plot(scl0,errorB,'--o',c='green',label='Berr')
        plt.title('Q10:␣Variation␣of␣error␣with␣noise')
        plt.xlabel(r'$Noise\;standard\;deviation\;\longrightarrow$')
        plt.ylabel(r'$MS\;error\;\;\longrightarrow$')
        plt.legend()
        plt.grid(True)

    plotABerror(errorA,errorB)

    plt.figure(5)
    loglog()
    errorbar(scl0,errorA,yerr=scl0,fmt='ro',label='Aerr')
    errorbar(scl0,errorB,yerr=scl0,fmt='go',label='Berr')
    title('Q11:␣Variation␣of␣error␣with␣noise')
    xlabel(r'$\sigma_{n}\;\longrightarrow$')
    ylabel(r'$MSerror\;\longrightarrow$')
    legend()
    grid(True)

    plotABerror(MSerrorA,MSerrorB,6)

    plt.figure(7)
    loglog()
    stem(scl0,errorA,'r',label='Aerr')
    stem(scl0,errorB,'g',label='Berr')
    title('Q11:␣Variation␣of␣error␣with␣noise')
    xlabel(r'$\sigma_{n}\;\longrightarrow$')
    ylabel(r'$MSerror\;\longrightarrow$')
    legend()
    grid(True)

def savefigs():
    for i in range(1,8):
        plt.figure(i)
        #manager = plt.get_current_fig_manager()
        #manager.resize(*manager.window.maxsize())
        savefig('plot__{}.png'.format(i))

if __name__ == '__main__':
    generateNewData()
    loadData()
    N,k = 101,9
    A0,B0=1.05,-0.105
    scl0 = logspace(-1,-3,k)

    M = c_[jn(2,t),t]# both are already column vectors
    p = c_[array([A0,B0])]
    y = g(t,A0,B0)

    plot1()
    plot2()
    verifyMPG(y)# to verify if M.p is approx equal to g())
    Eij_s()     #msq error for different signal noise and
    plot3()
    plot45(*errorEstimation(),*avgErrorEstimation())
    #savefigs()
    show()
```