EE2703 ASSIGNMENT 3

Nandha Varman
EE19B043

1 Objectives:

Reading data from files and parsing them

- Analysing the data to extract information
- Study the effect of noise on the fitting process
- Plotting graphs

2 Summary and Equations:

The data consists of 10 columns. The first column is time, while the remaining columns are data. The data columns correspond to the function

$$f(t) = 1.05J_2(t) - 0.105t + n(t)$$

The noise is given to be normally distributed, i.e., its probability distribution is given by

$$Pr(n(t)|\sigma) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{n(t)^2}{2\sigma^2}\right)$$

with σ given by

$$\sigma = logspace(-1, -3, 9)$$

We want to fit a function to this data. The function has the same general shape as the data but with unknown coefficients:

$$g(t; A, B) = AJ_2(t) + Bt$$

For our assignment, the values of t are discrete and known (from the datafile). We can obtain $g(t, A, B)$ as a column vector by creating a matrix equation:

$$g(t, A, B) = \begin{pmatrix} J_2(t_1) & t_1 \\ ... & ... \\ J_2(t_m) & t_m \end{pmatrix} \begin{pmatrix} A \\ B \end{pmatrix} \equiv M.p$$

For $A = 0, 0.1, ..., 2$ and $B = -0.2, -0.19, ..., 0$, for the data given in the columns, the **mean squared error** between the data (f_k) and the assumed model is given by

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

Consider data given in the form $(t, x) = \{t_i, x_i\}_{i=1}^{N}$ and a fitting model that is linear in parameters,i.e

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

where $F_i(t)$ are arbitrary functions of time. The fitting problem can be expressed as a matrix problem as below:

$$\begin{pmatrix} F_1(t_1) & F_2(t_1) & ... & F_N(t_1) \\ F_1(t_2) & F_2(t_2) & ... & F_N(t_2) \\ ... & ... & ... & ... \\ F_1(t_M) & F_2(t_M) & ... & F_N(t_M) \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ ... \\ p_N \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ ... \\ x_M \end{pmatrix}$$

Clearly the above equation cannot be exactly satisfied in the presence of noise, since the rank of F is N whereas the number of observations is $M \gg N$. We also make a very important assumption, namely that the noise added to each observation x_i, namely n_i , is "independent" of the noise added to any other observation. We wish to get the "best" guess for $\vec{p}$. For us, this means that we need to minimize the L_2 norm of the error. The error is given by

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

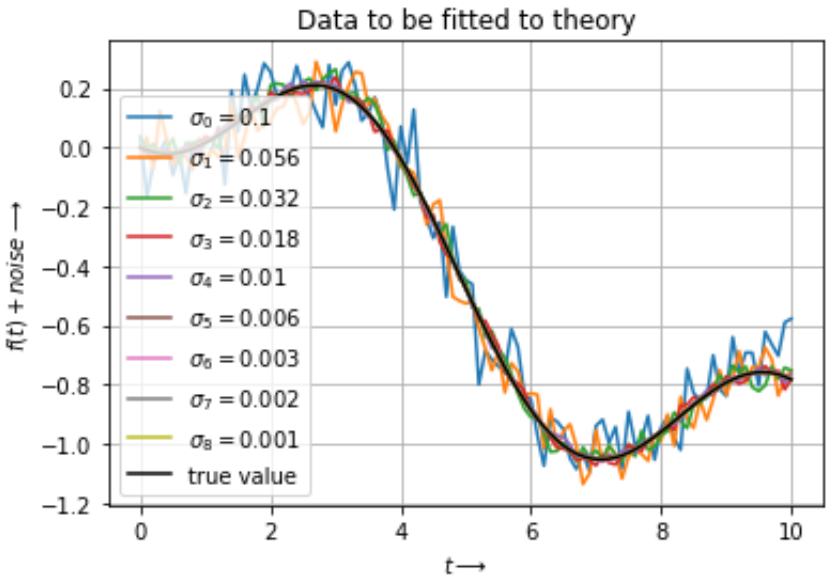
By using least squares approximation we get the parameters for best fit $\vec{p_0}$ as

$$\vec{p_0} = \left(F^T F\right)^{-1} F^T \vec{x}$$

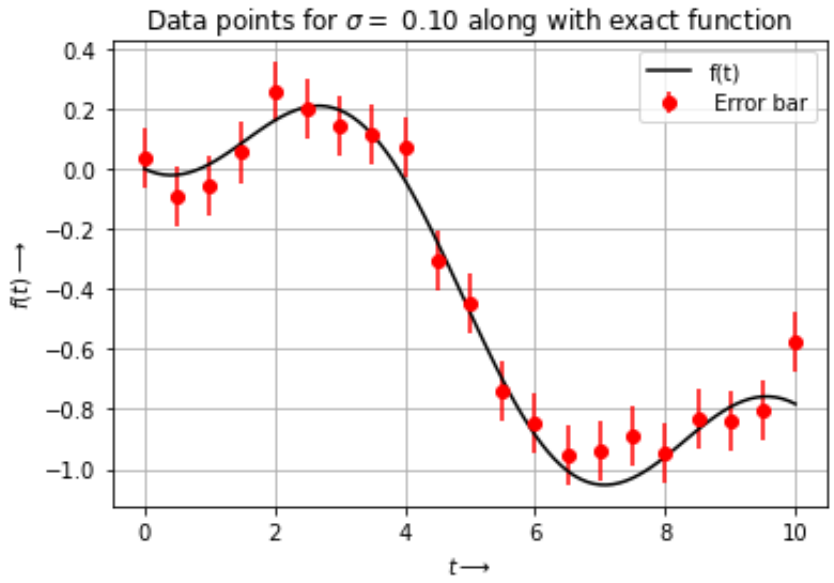
3 Code and Plots:

```
# Importing modules
from pylab import *
import scipy.special as sp
import sys

# PART 1:
# Run generate_data.py program

# PART 2:
try:
    input_data = np.loadtxt("fitting.dat", usecols =range(1,10))
except OSError:
    sys.exit("fitting.dat not found!!")
# Extracting data
data_columns = [[],[],[],[],[],[],[],[],[]]
for i in range(len(input_data)):
    for j in range(len(input_data[0])):
        data_columns[j].append(input_data[i][j])

# PART 3:
t = linspace(0,10,101)
sigma = logspace(-1,-3,9)
# Rounding off(3 places)
sigma = around(sigma,3)

# opening a new plot
figure(0)
# plotting the data for different sigma values
for i in range(len(data_columns)):
    plot(t,data_columns[i],label='$\sigma_{} = {}$'.format(i, sigma[i]))

# PART 4:
# defining a function with same general shape as data
def g(t, A, B):
    return A*sp.jn(2,t) + B*t
A = 1.05
B = -0.105
fitting_fn = g(t, A, B)
# Plotting
plot(t, fitting_fn, label='true value', color='#000000')
```

```
xlabel('$t\longrightarrow$')
ylabel('$f(t)+noise\longrightarrow$')
title('Data to be fitted to theory')
legend()
grid()
show()
```

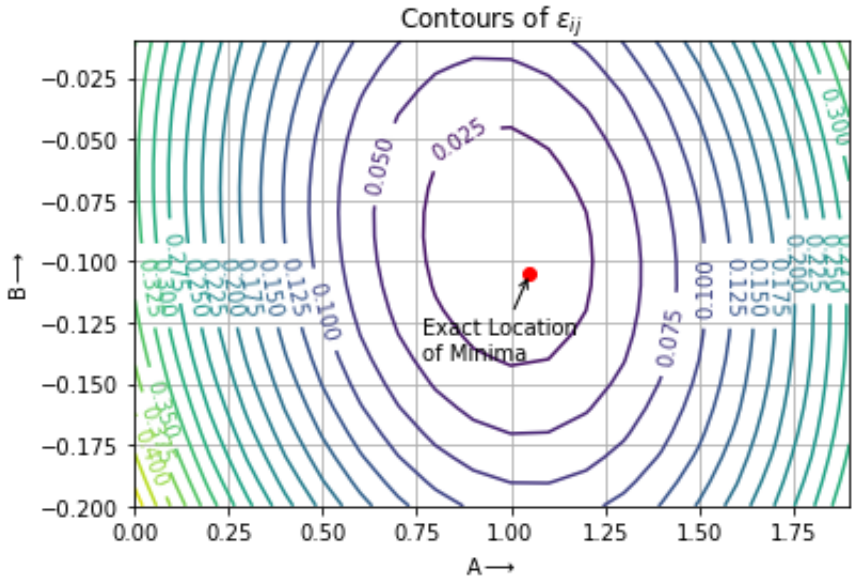



```
# PART 5:
# opening a new plot
figure(1)
xlabel('$t\longrightarrow$')
ylabel('$f(t)\longrightarrow$')
title('Data points for $\sigma =$ 0.10 along with exact function')
plot(t, fitting_fn, label='f(t)', color='#000000')
# Errorbar plot
errorbar(t[::5], data_columns[0][::5], 0.1, fmt='ro', label=' Error bar')
legend()
grid()
show()
```



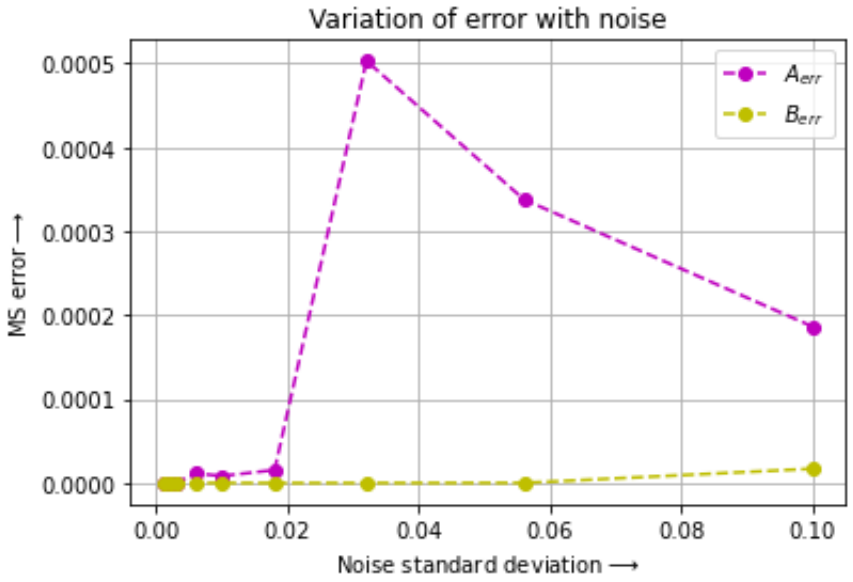
```
# PART 6:
# Creating column vector for least-squares estimation
jColumn = sp.jn(2,t)
M = c_[jColumn, t]
p = array([A, B])
# Constructiong matrix out of the column vectors
actual = c_[t,fitting_fn]

# PART 7:
# Calculating the fitting error for different combinations of A and B
A = arange(0,2,0.1)
B = arange(-0.2,0,0.01)
epsilon = zeros((len(A), len(B)))
for i in range(len(A)):
    for j in range(len(B)):
            epsilon[i][j] = mean(square(data_columns[0][:] - g(t[:], A[i], B[j])))

# PART 8:
# opening a new plot
figure(2)
# Contour plot of epsilon with A and B as axes
contour_plot=contour(A,B,epsilon,levels=20)
xlabel("A$\longrightarrow$")
```

```
ylabel("B$\longrightarrow$")
title("Contours of $\epsilon_{ij}$")
clabel(contour_plot, inline=1, fontsize=10)
# Annotating the graph with exact location of minima
plot([1.05], [-0.105], 'ro')
grid()
annotate("Exact Location\nof Minima", (1.05, -0.105), xytext=(-50, -40), textcoords="offset po
show()
```



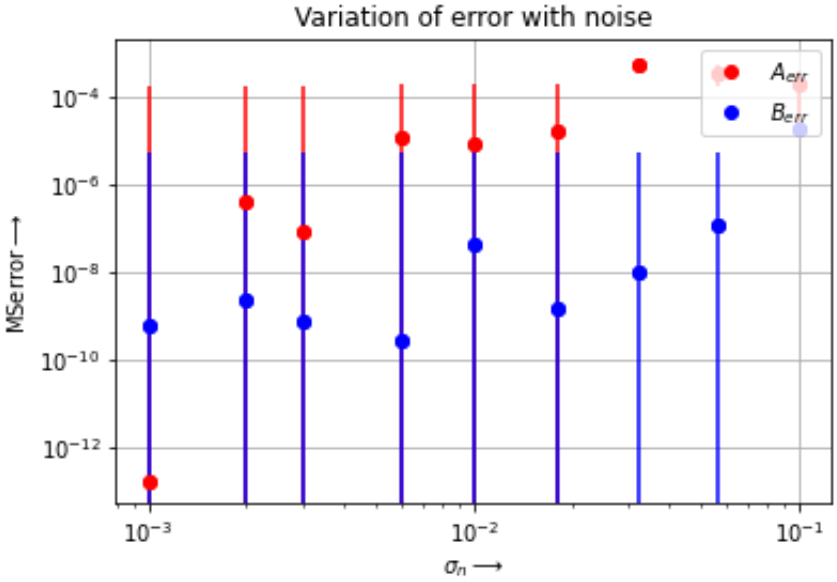

```
# PART 9:
# Least squares estimation
p= lstsq(M,fitting_fn,rcond=None)[0]

# PART 10:
# opening a new plot
figure(3)
perr= zeros((9, 2))
# Least square estimation taking different columns
for k in range(len(data_columns)):
    perr[k], *rest = lstsq(M, data_columns[k], rcond=None)
# Calculating Aerr and Berr for each least square estimation
Aerr = array([square(x[0]-p[0]) for x in perr])
```

```
Berr = array([square(x[1]-p[1]) for x in perr])
plot(sigma, Aerr, 'mo--', label='$A_{err}$')
plot(sigma, Berr, 'yo--', label='$B_{err}$')
xlabel("Noise standard deviation$\longrightarrow$")
title("Variation of error with noise")
ylabel("MS error$\longrightarrow$")
legend()
grid()
show()
```




```
# PART 11:
# opening a new plot
figure(4)
# Plotting Aerr and Berr vs. sigma in a log-log scale
loglog(sigma, Aerr, 'ro', label="$A_{err}$")
loglog(sigma, Berr, 'bo', label="$B_{err}$")
legend()
errorbar(sigma, Aerr, std(Aerr), fmt='ro')
errorbar(sigma, Berr, std(Berr), fmt='bo')
xlabel("$\sigma_{n}\longrightarrow$")
title("Variation of error with noise")
ylabel("MSerror$\longrightarrow$")
```

```
legend(loc='upper right')
grid()
show()
```



4 Observations

- The contour plot of ϵ_{ij} is ***inverted dome*** shaped. It therefore has one absolute minima.
- The plot of error in fitting versus σ of noise is less interesting. The log-log plot however shows linear behavior.

5 Inference:

The logarithm of error in fitting by least squares estimation is **linearly dependent** on the logarithm of standard deviation of noise.