Mathematical Physics III

Lab Assignment #6

Ishmeet Singh

College Roll No	:	2020PHY1221
University Roll No	:	20068567051
Unique Paper Code	:	32221401
Paper Title	:	Mathematical physics III Lab
Course and Semester	:	B.Sc.(Hons.) Physics, Sem IV
Due Date	:	14 March 2022
Date of Submission	:	14 March 2022
Lab Report File Name	:	2020PHY1221_A6.pdf
Submitted to	:	Dr. Mamta Dahiya

Team Member :

Name	:	Sarthak Jain
Roll Number	:	2020PHY1201

Shri Guru Tegh Bahadur Khalsa College, University of Delhi
New Delhi-110007, India.

Contents

1 Theory

1.1 Principle of Maximum Likelihood

Maximum likelihood estimation is a statistical method for estimating the parameters of a model. In maximum likelihood estimation, the parameters are chosen to maximize the likelihood that the assumed model results in the observed data.
This implies that in order to implement maximum likelihood estimation we must:

1. Assume a model, also known as a data generating process, for our data.

2. Be able to derive the likelihood function for our data, given our assumed model.

Once the likelihood function is derived, maximum likelihood estimation is nothing more than a simple optimization problem.

1.1.1 Maximum Likelihood Estimation and Least Squares Fitting

Consider a data set of $\{(x_i, y_i)\}$ where x_i are known exactly and y_i are measured each time with some uncertainty.
Let $y = f(x; \overrightarrow{a})$, $\overrightarrow{a}$ is set of parameters to be estimated.
Central Limit Theorem implies that the distribution of measured y-values about their ideal values is Gaussian and probability of a particular y_i for a given x_i is

$$P(y_i; a) = \frac{1}{\sigma_i\sqrt{2\pi}} exp\left[-\frac{[y_i - f(x_i; a)]^2}{2\sigma_i^2}\right]$$

Maximising the log likelihood function, ln (L) is equivalent to minimising

$$\sum_i \left[\frac{y_i - f(x_i; a)}{\sigma_i}\right]^2$$

And so we make the weighted sum of squared deviations (from predicted values) least – as small as possible by varying the parameters.
This shows the relation between Maximum likelihood and Least squares as we define Chi-squared,

$$\chi^2 = \sum_i \left[\frac{y_i - f(x_i; a)}{\sigma_i}\right]^2$$

Minimising χ^2 means to make the least squares model more accurate which is what is done by also using the Maximum likelihood method.

1.2 Method of Weighted Least Squares

Weighted Least Square Fitting (WLS) is a generalisation of ordinary least squares in which the knowledge of variables' variance is incorporated into the regression.
In weighted least squares method we give each data point its proper amount of influence over the parameter estimates. In WLS all of the observables have their quality taken in account for better estimation.
WLS method is used in the situations in which the data points are of varying quality.

Consider a data set $\{(x_i; [y_i])\}$ where x_i are known exactly and σ_i is the value of error in $\bar{y}_i$ and $\bar{y}_i$ is the mean of $[y_{ij}]$ for each i.
Let $y = f(x : m, c)$ are set of parameters to be estimated.
Then from central limit theorem, distribution of measured 'y' values about their ideal values is gaussian, and the probability of a particular y_i for a given x_i is,

$$P(y_i; m, c) = \frac{1}{\sigma_i\sqrt{2\pi}} exp\left[-\frac{[y_i - f(x_i; a)]^2}{2\sigma_i^2}\right]$$

Then, maximising maximum likelihood function of the estimates $\widehat{m}$ and $\widehat{c}$ is similar to minimising,

$$\sum_i \left[\frac{y_i - f(x : m, c)}{\sigma_i}\right]^2$$

This term is called χ^2
Therefore,

$$\chi^2 = \sum_i \left[\frac{y_i - f(x : m, c)}{\sigma_i}\right]^2$$

Then, we will minimise χ^2 and not m and c.
So, in linear WLS where $y = mx + c$ we will use χ^2 to find estimates for m and c.
i.e.,

$$\chi^2 = \sum w_i(y_i - mx_i - c)^2$$

Here,

$$w_i = \frac{1}{\sigma_i^2}$$

Thus,

$$\frac{\partial(\chi^2)}{\partial m} = 0$$

$$\begin{aligned}
\implies \sum \frac{\partial}{\partial m}[w_i(y_i - mx_i - c)^2] &= 0 \\
\implies -2\sum w_i x_i(y_i - mx_i - c) &= 0 \\
\implies \sum w_i x_i y_i - m\sum w_i x_i^2 - c\sum w_i x_i &= 0
\end{aligned}$$

Labelling the above equation as 1.
When

$$\frac{\partial(\chi^2)}{\partial c} = 0$$

$$\begin{aligned}
\implies -2\sum w_i(y_i - mx_i - c) &= 0 \\
\implies \sum w_i y_i - m\sum w_i x_i - c\sum w_i &= 0
\end{aligned}$$

$$\implies \boxed{c = \frac{\sum w_i y_i - m\sum w_i x_i}{\sum w_i}}$$

$$\implies \boxed{c = \bar{y} - m\bar{x}}$$

Here, $\bar{y} = \frac{\sum w_i y_i}{\sum w_i}$ and $\bar{x} = \frac{\sum w_i x_i}{\sum w_i}$
Putting c in 1

$$\sum w_i x_i y_i - m\sum w_i x_i^2 - \bar{y} - m\bar{x}\sum w_i x_i = 0$$

$$\implies \boxed{m = \frac{\sum w_i x_i y_i - \bar{y}\sum w_i x_i}{\sum w_i x_i^2 - \bar{x}\sum w_i x_i}}$$

$$m = \frac{S_x y - \bar{y}S_x}{S_x^2 - \bar{x}S_x} = \frac{\sum w_i(x_i - \bar{x})(y_i - \bar{y})}{\sum w_i(x_i - \bar{x})^2}$$

where,

$$\begin{aligned}
S_{xy} &= \sum w_i x_i y_i \\
S_x &= \sum w_i x_i \\
S_x^2 &= \sum w_i x_i^2
\end{aligned}$$

So, we can finally write,

$$\boxed{m = \frac{\sum w_i \sum w_i x_i y_i - \sum w_i x_i \sum w_i y_i}{\sum w_i \sum w_i x_i^2 - (\sum w_i x_i)^2}}$$

and,

$$\boxed{c = \frac{\sum w_i x_i^2 \sum w_i y_i - \sum w_i x_i \sum w_i x_i y_i}{\sum w_i \sum w_i x_i^2 - (\sum w_i x_i)^2}}$$

Let $\Delta = \sum w_i \sum w_i x_i^2 - (\sum w_i x_i)^2$

Since, this m depends on x_i and y_i but only y_i has error.

Therefore, by propagation of error,

$$\sigma_m^2 = \sum \left(\frac{\partial m}{\partial y_i} \sigma_i \right)^2$$

i.e.,

$$\frac{\partial m}{\partial y_i} = \frac{(\sum w_i) w_i x_i - (\sum w_i x_i) w_i}{\Delta}$$

Then,

$$\left(\frac{\partial m}{\partial y_i} \right) \sigma_i = \frac{\frac{(\sum w_i) x_i}{\sigma_i} - \frac{(\sum w_i x_i)}{\sigma_i}}{\Delta}$$

$$\begin{aligned}
\implies \sigma_m^2 &= \sum \frac{\left(\frac{(\sum w_i) x_i}{\sigma_i} - \frac{(\sum w_i x_i)}{\sigma_i} \right)^2}{\Delta^2} \\
\sigma_m^2 &= \sum \frac{\left[\frac{(\sum w_i)^2 x_i^2}{\sigma_i^2} - \frac{(\sum w_i x_i)^2}{\sigma_i^2} - 2 \frac{(\sum w_i \sum w_i x_i) x_i}{\sigma_i^2} \right]}{\Delta^2} \\
\sigma_m^2 &= \sum w_i \frac{[(\sum w_i)^2 x_i^2 + (\sum w_i x_i)^2 - 2 (\sum w_i \sum w_i x_i) x_i]}{\Delta^2} \\
\sigma_m^2 &= \sum w_i \frac{\left[(\sum w_i)^2 x_i^2 + [(\sum w_i) \bar{x}]^2 - 2 [(\sum w_i)^2 x_i] \right]}{\Delta^2} \\
\sigma_m^2 &= \sum w_i \frac{[(\sum w_i)^2 (x_i - \bar{x})^2]}{\Delta^2} \\
\sigma_m^2 &= \frac{(\sum w_i)^2 \sum w_i (x_i - \bar{x})^2}{\Delta^2} \\
\sigma_m^2 &= \frac{(\sum w_i)^2 \sum w_i (x_i - \bar{x})^2}{\Delta [\sum w_i (\sum w_i x_i^2 - \bar{x} \sum w_i x_i]}
\end{aligned}$$

$$\sigma_m^2 = \frac{\sum w_i}{\Delta}$$

$$\boxed{\sigma_m = \sqrt{\frac{\sum w_i}{\Delta}}}$$

Similarly,

$$\sigma_c^2 = \sum \left(\frac{\partial c}{\partial y_i} \sigma_i \right)^2$$

Here,

$$\frac{\partial c}{\partial y} = \frac{(\sum w_i x_i^2) w_i - (\sum w_i x_i) w_i x_i}{\Delta}$$

Using similar steps as we did previously, we get

$$\sigma_c^2 = \frac{\sum w_i x_i^2}{\Delta} = \frac{S_x^2}{\Delta}$$

$$\boxed{\sigma_c = \sqrt{\frac{S_x^2}{\Delta}}}$$

1.2.1 Reduction to Ordinary Least Square Fitting

When our data points are equally distributed along their best estimates, i.e.,

$$\sigma_i = \sigma \qquad \forall \;\; i$$

Then, our data points provide equally precise information about deterministic part of the process, i.e., for our all the values of explanatory variables standard deviation is constant. i.e., w being constant can be written out of the summation then, our parameters for best values of y becomes

$$m = \frac{w \sum (x_i - \bar{x})(y_i - \bar{y}}{w \sum (x_i - \bar{x})^2}$$

$$\boxed{m = \frac{\sum (x_i - \bar{x})(y_i - \bar{y}}{\sum (x_i - \bar{x})^2}}$$

$$\boxed{c = \bar{y} - m\bar{x}}$$

1.3 Correlation Coefficient

The Correlation Coefficient is the specific measure that quantifies the strength of the linear relationship between two variables in a correlation analysis.
The Adjusted Correlation Coefficient is an adjustment for the correlation coefficient that takes into account the no. of variables in a data set, and is optimised for getting closer to the "time" model by weighing the variables according to the error in them.

2 Programming

2.1 2020PHY1221_A6.py

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Name => Ishmeet Singh   Roll no. => 2020PHY1221
# Patner's Name => Sarthak Jain   Roll no. => 2020PHY1201

def Mylsf(x,y):
    n = len(x)
    # SLOPE
    slope = (n*np.sum(x*y) - (np.sum(x)*np.sum(y)))/(n*np.sum((x**2)) - (
    np.sum(x))**2)

    #INTERCEPT
    intercept = (((np.sum(np.power(x,2)))*np.sum(y)) - (np.sum(x*y)*np.sum
    (x)))/((n*np.sum(np.power(x,2))) - (np.power(np.sum(x),2)))

    # Y CALCULATED
    y_cal = []
    for i in range(n):
        y_cal.append(slope*x[i] + intercept)
    y_cal = np.array(y_cal)

    # SUM OF RESIDUAL
    res = []
    for j in range(n):
        res.append(y[j] - y_cal[j])
    res = np.array(res)
```

```
    res_s = np.sum(res)

    # SQUARE OF SUM oF RESIDUAL
    res_1 = []
    for j in range(n):
        res_1.append((y[j] - y_cal[j])**2)
    res_1 = np.array(res_1)
    res_ss = np.sum(res_1)

    # STANDARD ERROR IN SLOPE
    s = np.sqrt(res_ss/(n-2))
    SS_xx = ((np.sum(np.power(x,2))) - (np.power(np.sum(x),2)/n))
    Serr_slope = s/(np.sqrt(SS_xx))

    # STANDARD ERROR IN INTERCEPT
    Serr_intercept = (Serr_slope * ((np.sqrt(np.sum(np.power(x,2))))/n))

    # COFFICIENT OF DETERMINATION
    diff_l = []
    y_mean = np.mean(y)
    for k in range(n):
        diff_l.append((y[k] - y_mean)**2)
    diff_l = np.array(diff_l)
    tss = np.sum(diff_l)
    R_sqr = (tss - res_ss)/tss # cofficient of determination

    # CORRELATION COFFICIENT
    cc = np.sqrt(R_sqr) # cofficient of correlation

    return slope,intercept,y_cal,res_s,res_ss,Serr_slope,Serr_intercept,cc

def MyWlsf(x,y,w):
    n = len (x)
    wxy = []
    wx = []
    wy = []
    wx2 = []
    for i in range (n):
       wxy.append(w[i]* x[i]* y[i])
       wx.append(w[i]* x[i])
       wy.append(w[i]* y[i])
       wx2.append(w[i]* x[i]* x[i])
```

```
    # SLOPE
    slope_w = ( np.sum (w)* np.sum ( wxy ) - np.sum ( wx )* np.sum ( wy )) /( np.sum (w) *np.sum ( wx2 ) - np.sum( wx )* np.sum ( wx ))
    slope_w = np.array(slope_w)

    # INTERCEPT
    intercept_w = (np.sum ( wx2 )* np.sum ( wy ) - np.sum( wx )* np.sum ( wxy )) /( np.sum( w)* np.sum ( wx2 ) - np.sum ( wx )* np.sum ( wx ))
    intercept_w = np.array(intercept_w)

    # Y CALCULATED
    y_cal_w = []
    for i in range(n):
        y_cal_w.append(slope_w*x[i] + intercept_w)
    y_cal_w = np.array(y_cal_w)

    # SUM OF RESIDUAL
    res_w = []
    for j in range(n):
        res_w.append(y[j] - y_cal_w[j])
    res_w = np.array(res_w)
    res_s_w = np.sum(res_w)

    # SQUARE OF SUM oF RESIDUAL
    res_1_w = []
    for j in range(n):
        res_1_w.append((y[j] - y_cal_w[j])**2)
    res_1_w = np.array(res_1_w)
    res_ss_w = np.sum(res_1_w)

    # ERROR IN SLOPE
    slope_err_wls = np.sqrt(( np.sum (w)) /( np.sum(w )* np.sum( wx2 ) - np.sum( wx )* np.sum( wx )))

    # ERROR IN INTERCEPT
    intercept_err_wls = np.sqrt(( np.sum ( wx2 )) /( np.sum(w)* np.sum( wx2 ) - np.sum( wx )* np.sum( wx ) ))

    # COFFICIENT OF DETERMINATION
    diff_l = []
    y_mean = np.mean(y)
```

```
    for k in range(n):
        diff_l.append((y[k] - y_mean)**2)
    diff_l = np.array(diff_l)
    tss = np.sum(diff_l)
    R_sqr_wls = (tss - res_ss_w)/tss # cofficient of determination

    # CORRELATION COFFICIENT
    cc_wls = np.sqrt(R_sqr_wls) # cofficient of correlation

    return slope_w,intercept_w,y_cal_w,res_s_w,res_ss_w,slope_err_wls,
   intercept_err_wls,R_sqr_wls,cc_wls


if __name__ == "__main__":

    print("\nName: Sarthak Jain\tRoll no.: 2020PHY1201\nPatner's Name:
   Ishmeet Singh\tRoll no.: 2020PHY1221")

    x_vals = pd.read_csv(r"C:\Users\parmm\OneDrive\Desktop\wlsf\data.csv",
     usecols = [1])
    x_vals = (x_vals.to_numpy()).flatten()
    df = pd.read_csv(r"C:\Users\parmm\OneDrive\Desktop\wlsf\data.csv",
   usecols = range(2, 12))
    df = df.to_numpy()


    y_mean = np.array([])
    y_std_error = np.array([])
    for i in range(len(df[0])):
        mean = np.mean(df[i])
        y_mean = np.append(y_mean, mean)
        var = np.var(df[i])
        std_error = (4*mean**2)*var/len(df[0])
        y_std_error = np.append(y_std_error, std_error)

    x = x_vals.reshape(-1, 1)
    y = (y_mean**2).reshape(-1, 1)
    w = 1/y_std_error

    slope,intercept,y_cal,res_s,res_ss,Serr_slope,Serr_intercept,cc =
   Mylsf(x,y)
    slope_w,intercept_w,y_cal_w,res_s_w,res_ss_w,slope_err_wls,
```

```
    intercept_err_wls,R_sqr_wls,cc_wls = MyWlsf(x,y,w)

    data = np.column_stack([x,y,w])
    np.savetxt (" 1201. txt ",data, header = "xi , yi , wi")

    print("\nFITTING PARAMETERS (OLSF):")
    print("\nSlope: ",slope)
    print("\nError in Slope: ",Serr_slope)
    print("\nIntercept: ",intercept)
    print("\nError in Intercept: ",Serr_intercept)
    print("\nSum of residulas: ",res_s)
    print("\nSum of square of residulas: ",res_ss)
    print("\nCofficient of Correlation: ",cc)

    print("\n
    ----------------------------------------------------------------\n")

    print("\nFITTING PARAMETERS (WLSF):")
    print("\nSlope: ",slope_w)
    print("\nError in Slope: ",slope_err_wls)
    print("\nIntercept: ",intercept_w)
    print("\nError in Intercept: ",intercept_err_wls)
    print("\nSum of residulas: ",res_s_w)
    print("\nSum of square of residulas: ",res_ss_w)
    print("\nCofficient of Correlation: ",cc_wls)

    k = (4*np.pi**2)/slope_w
    m = intercept_w*k/(4*np.pi**2)
    error_k = slope_err_wls*k/slope_w
    error_m = (intercept_err_wls/intercept_w + error_k/k)*m

    print("\n
    -----------------------------------------------------------------\n")

    print("\nValue of k: ",k)
    print("\nValue of m: ",m)
    print("\nError in k: ",error_k)
    print("\nError in m: ",error_m)

    # Plots and Scatters
    plt.scatter(x, y, marker = 'o')
    plt.plot(x, y_cal, linestyle = 'dashed', linewidth = 1, label = "OLS
```

```
    Fitted Line",c = "red")
    plt.plot(x, y_cal_w, linewidth = 1, label = "WLS Fitted Line",c = "
    green")
    plt.title("IshmeetSingh and Sarthak Jain\nLinear Regression for Spring
     Constant")        #NAME OF EXPERIMENT
    plt.ylabel("Time Period - $T^2$ ($s^2$)\n")          #Y LABEL
    plt.xlabel("Mass (g.)\n")                            #X LABEL
    plt.grid(ls = "--")

    plt.legend()
    plt.show()
```

3 Results

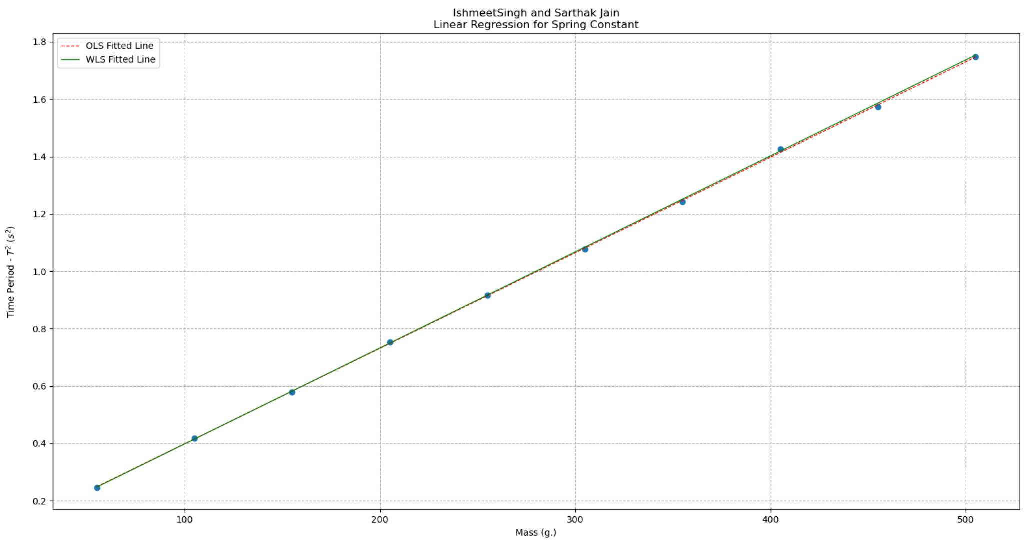

Figure 1: $T^2 vs M$

```
FITTING PARAMETERS (OLSF):

Slope:  0.0033288099478787875

Error in Slope:  1.2747360352292707e-05

Intercept:  0.06574134959393921

Error in Intercept:  0.0012685064753113395

Sum of residulas:  3.58046925441613e-15

Sum of square of residulas:  0.0002681170733194866

Cofficient of Correlation:  0.9999413478000884

-----------------------------------------------------------------


FITTING PARAMETERS (WLSF):

Slope:  0.003347233933131483

Error in Slope:  3.207700002558663e-05

Intercept:  0.06334311929953243

Error in Intercept:  0.0099676507969

Sum of residulas:  -0.027604855763475822

Sum of square of residulas:  0.0004143300462137248

Cofficient of Correlation:  0.9999093613954779

------------------------------------------------------------------


Value of k:  11794.340758078912

Value of m:  18.924019224515984

Error in k:  113.02677863471982

Error in m:  3.1592284176433205
```

Figure 2: Uncertainty Values