Weighted Least Square Fitting
Lab Report for Assignment No. 6

PAWANPREET KAUR
(2020PHY1092)

KABIR SETHI
(2020PHY1097)

S.G.T.B. Khalsa College, University of Delhi, Delhi-110007, India.

March 14, 2022

Submitted to Dr. Mamta
"32221401 - MATHEMATICAL PHYSICS III"

Contents

1 Theory

1.1 Maximum likelihood and Least squares

Maximum likelihood estimation is a method that determines values for the parameters of a model. The parameter values are found such that they maximise the likelihood that the process described by the model produced the data that were actually observed.
The method of least squares can be derived from the **maximum likelihood** with the help of central limit theorem, which simply states that the point that we observe which is a dependent quantity is from a gaussian distribution with the help of standard error,σ_i

We simply assume that the distribution that we have is Gaussian Probability Distribution. The calculated y coordinate from the best fit line, is the most probable value.
And we know that the slope (m) and intercept (c) are a function of the pobability density function which helps calculating the most probable value of y_i using these m,c and the central limit theorem.
We assume that the data set has a uncertainity of σ_i in y_i.
Let $y = f(x : m, c)$ be the parameters to be estimated.Then from the central limit theoram,distribution of measured y values about their ideal value is gaussian.
Then probability for a particular y_i for given x_i is-

$$P(y_i : m, c) = \frac{1}{\sigma_i\sqrt{2\pi}} exp\left(\frac{-(y_i - f(x_i : m, c)}{\sigma_i}\right) \tag{1}$$

Also, Maximizing the maximum likelihood function of the estimators $\hat{m}$ and $\hat{c}$ is similar to minimizing -

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

Using this χ^2 we will determine the $\hat{m}$ and $\hat{c}$ for a function : $y = mx + c$.

$$\chi^2 = \Sigma\left[\frac{(y_i - mx_i - c)}{\sigma_i}\right]^2$$
$$\chi^2 = \Sigma w_i(y_i - mx_i - c)^2, w_i = \frac{1}{\sigma_i^2}$$

1.2 Weighted Mean and error in mean

Suppose the result for one experiment is $x_i \pm \alpha_i$ and that of another be $x_j \pm \alpha_j$. If the two results have similar error, than the mean of the results would be the average of them as they have equal importance to both the data. And it is given by:

$$\bar{x}_{i,j} = \frac{1}{2}(x_i + x_j)$$

The error in the weighted mean is given by:

$$\frac{1}{\alpha_{(\bar{x}_i,j)}} = \sqrt{\frac{1}{\alpha_i^2} + \frac{1}{\alpha_j^2}}$$

The best combined estimate, x_{CE} , is the sum of the weighted means, divided by the sum of the weightings:

$$x_{CE} = \frac{\sum_i w_i x_i}{\sum_i w_i}$$

Let there be N measurement of the same quantity, x_i, with $i = 1, 2, 3, ...N$
and thek σ to be the error in these measurements. As the error is the same for each x_i , each measurement carries the same weight,

$$\therefore x_{CE} = \frac{\sum_i^N w_i x_i}{\sum_i^N w_i}$$

$$x_{CE} = \frac{\sum_i^N w_i}{N} = \bar{x}$$

This gives the expected result that the combined estimate is the mean of the measurements. In a similar manner we can calculate the standard error of the weighted mean to be

$$\frac{1}{\alpha_{CE}^2} = \sum \frac{1}{\sigma^2} = \frac{N}{\sigma^2}$$

$$\boxed{\alpha_{CE} = \frac{\sigma}{\sqrt{N}}}$$

1.3 Derivation of Slope and Intercept

We have to minimize χ^2 w.r.t m and c then.

$$\frac{\partial \chi^2}{\partial m} = 0 \tag{3}$$

$$\sum \frac{\partial [w_i(y_i - mx_i - c)^2]}{\partial m} = 0 \tag{4}$$

$$-2\sum [w_i x_i (y_i - mx_i - c)] = 0 \tag{5}$$

$$\Sigma w_i x_i y_i - m\Sigma w_i x_i^2 - c\Sigma w_i x_i = 0 \tag{6}$$

$$\frac{\partial \chi^2}{\partial c} = 0 \tag{7}$$

$$-2\sum [w_i (y_i - mx_i - c)] = 0 \tag{8}$$

$$\Sigma w_i y_i - m\Sigma w_i x_i - c\Sigma w_i = 0 \tag{9}$$

$$\tag{10}$$

$$\implies c = \frac{\Sigma w_i y_i - m\Sigma w_i x_i}{\Sigma w_i} \tag{11}$$

$$\boxed{c = \overline{Y} - m\overline{X}} \tag{12}$$

$$\overline{Y} = \frac{\Sigma w_i y_i}{w_i}, \overline{X} = \frac{\Sigma w_i x_i}{w_i} \tag{13}$$

Putting c in eq 8.

$$\Sigma w_i x_i y_i - m\Sigma w_i x_i^2 - (\overline{Y} - m\overline{X})\Sigma w_i x_i = 0 \tag{14}$$

$$m = \frac{\Sigma w_i x_i y_i - \overline{Y}\Sigma x_i}{\Sigma w_i x_i^2 - \overline{X}\Sigma w_i x_i} \tag{15}$$

$$m = \frac{\Sigma w_i \Sigma w_i x_i y_i - \Sigma w_i y_i \Sigma x_i}{\Sigma w_i \Sigma w_i x_i^2 - (\Sigma w_i x_i)^2} \tag{16}$$

$$c = \frac{\Sigma w_i x_i^2 \Sigma w_i y_i - \Sigma w_i x_i \Sigma w_i x_i y_i}{\Sigma w_i \Sigma w_i x_i^2 - (\Sigma w_i x_i)^2} \tag{17}$$

$$\Delta = \Sigma w_i \Sigma w_i x_i^2 - (\Sigma w_i x_i)^2, S_{xy} = \sum w_i x_i y_i \tag{18}$$

$$S_x = \sum w_i x_i, S_{x^2} = \sum w_i x_i^2, S_y = \sum w_i y_i \tag{19}$$

1.4 Derivation of Error in Slope and Intercept

Since,the constant m depends on x_i and y_i but only y_i have uncertainity around therefore,by propagation of errors.

$$\sigma_m^2 = \sum \left(\frac{\partial m}{\partial y_i}\right)^2 \sigma_i^2 \tag{20}$$

$$\frac{\partial m}{\partial y_i} = \frac{(\Sigma w_i) w_i x_i - (\Sigma w_i x_i) w_i}{\Delta} \tag{21}$$

$$\left(\frac{\partial m}{\partial y_i}\right) \sigma_i = \frac{(\Sigma w_i)\frac{x_i}{\sigma_i} - \frac{(\Sigma w_i x_i)}{\sigma_i}}{\Delta} \tag{22}$$

$$\sigma_m^2 = \sum \frac{\left[(\Sigma w_i)\frac{x_i}{\sigma_i} - \frac{(\Sigma w_i x_i)}{\sigma_i}\right]^2}{\Delta^2} \tag{23}$$

$$\sigma_m^2 = \sum \frac{w_i \left[(\Sigma w_i)^2 x_i^2 + (\Sigma w_i x_i)^2 - 2(\Sigma w_i \Sigma w_i x_i) x_i\right]}{\Delta^2} \tag{24}$$

$$\sigma_m^2 = \sum \frac{w_i \left[(\Sigma w_i)^2 x_i^2 + (\Sigma w_i \overline{X})^2 - 2(\Sigma w_i)^2 \overline{X} x_i\right]}{\Delta^2} \tag{25}$$

$$\sigma_m^2 = \sum \frac{w_i \left[(\Sigma w_i)^2 (x_i^2 + \overline{X}^2 - 2\overline{X} x_i)\right]}{\Delta^2} \tag{26}$$

$$\sigma_m^2 = \sum \frac{w_i \left[(\Sigma w_i)^2 (x_i - \overline{X})^2\right]}{\Delta^2} \tag{27}$$

$$\sigma_m^2 = \frac{(\Sigma w_i)^2 \Sigma w_i (x_i - \overline{X})^2}{\Delta^2} \tag{28}$$

$$\sigma_m^2 = \frac{\Sigma w_i}{\Delta} \tag{29}$$

$$\sigma_m = \sqrt{\frac{\Sigma w_i}{\Delta}} \tag{30}$$

Similarly for intercept we can write.

$$\sigma_c^2 = \sum \left(\frac{\partial c}{\partial y_i}\right)^2 \sigma_i^2 \tag{31}$$

$$\frac{\partial c}{\partial y_i} = \frac{(\Sigma w_i x_i^2) w_i - (\Sigma w_i x_i) w_i x_i}{\Delta} \tag{32}$$

$$\left(\frac{\partial c}{\partial y_i}\right) \sigma_i = \frac{\frac{\Sigma w_i x_i^2}{\sigma_i} - \frac{(\Sigma w_i x_i) x_i}{\sigma_i}}{\Delta} \tag{33}$$

$$\sigma_c^2 = \sum \frac{\left[\frac{\Sigma w_i x_i^2}{\sigma_i} - \frac{(\Sigma w_i x_i) x_i}{\sigma_i}\right]^2}{\Delta^2} \tag{34}$$

$$\sigma_c^2 = \sum \frac{w_i \left[(\Sigma w_i x_i^2)^2 + (\Sigma w_i x_i)^2 x_i^2 - 2(\Sigma w_i x_i^2)(\Sigma w_i x_i) x_i\right]}{\Delta^2} \tag{35}$$

$$\sigma_c^2 = \sum \frac{w_i \left[(\Sigma w_i x_i^2)^2 - (\Sigma w_i x_i^2)(\Sigma w_i x_i) x_i + (\Sigma w_i x_i)^2 x_i^2 - (\Sigma w_i x_i^2)(\Sigma w_i x_i) x_i\right]}{\Delta^2} \tag{36}$$

$$\sigma_c^2 = \sum \frac{w_i \left[\Sigma w_i x_i^2 (\Sigma w_i x_i^2 - (\Sigma w_i x_i) x_i) + \Sigma w_i x_i ((\Sigma w_i x_i) x_i^2 - (\Sigma w_i x_i^2) x_i)\right]}{\Delta^2} \tag{37}$$

$$\sigma_c^2 = \sum \frac{\left[\Sigma w_i x_i^2 (\Sigma w_i \Sigma w_i x_i^2 - (\Sigma w_i x_i)^2) + \Sigma w_i x_i ((\Sigma w_i x_i) \Sigma w_i x_i^2 - (\Sigma w_i x_i^2) \Sigma w_i x_i)\right]}{\Delta^2} \tag{38}$$

$$\sigma_c^2 = \frac{\Sigma w_i x_i^2}{\Delta} \tag{39}$$

$$\sigma_c = \sqrt{\frac{\Sigma w_i x_i^2}{\Delta}} \tag{40}$$

$$\tag{41}$$

Correlation coefficient

$$r = \frac{\Sigma w_i (x_i - \overline{X})(y_i - \overline{Y})}{\sqrt{\Sigma w_i (x_i - \overline{X})^2 \Sigma w_i (y_i - \overline{Y})^2}}$$

To sum up

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

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

$$\boxed{\Delta = \Sigma w_i \Sigma w_i x_i^2 - (\Sigma w_i x_i)^2, S_{xy} = \sum w_i x_i y_i}$$

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

$$\boxed{\sigma_c = \sqrt{\frac{\Sigma w_i x_i^2}{\Delta}}}$$

$$\boxed{r = \frac{\Sigma w_i (x_i - \overline{X})(y_i - \overline{Y})}{\sqrt{\Sigma w_i (x_i - \overline{X})^2 \Sigma w_i (y_i - \overline{Y})^2}}}$$

2 RESULTS

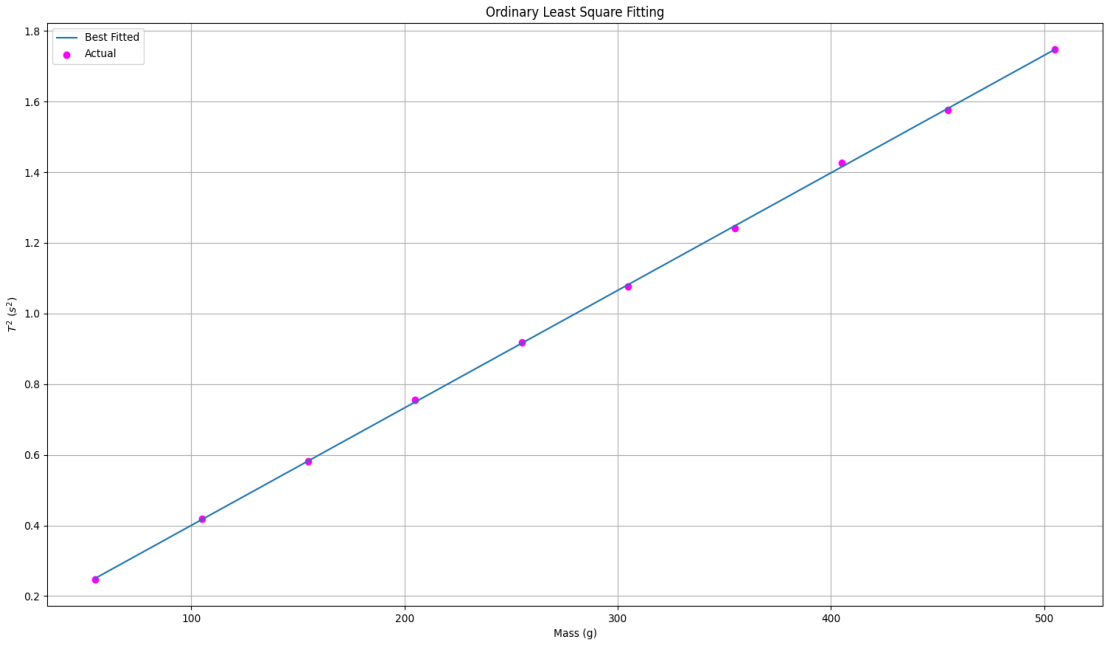

Figure 1: *Ordinary Least Squares Fitting*

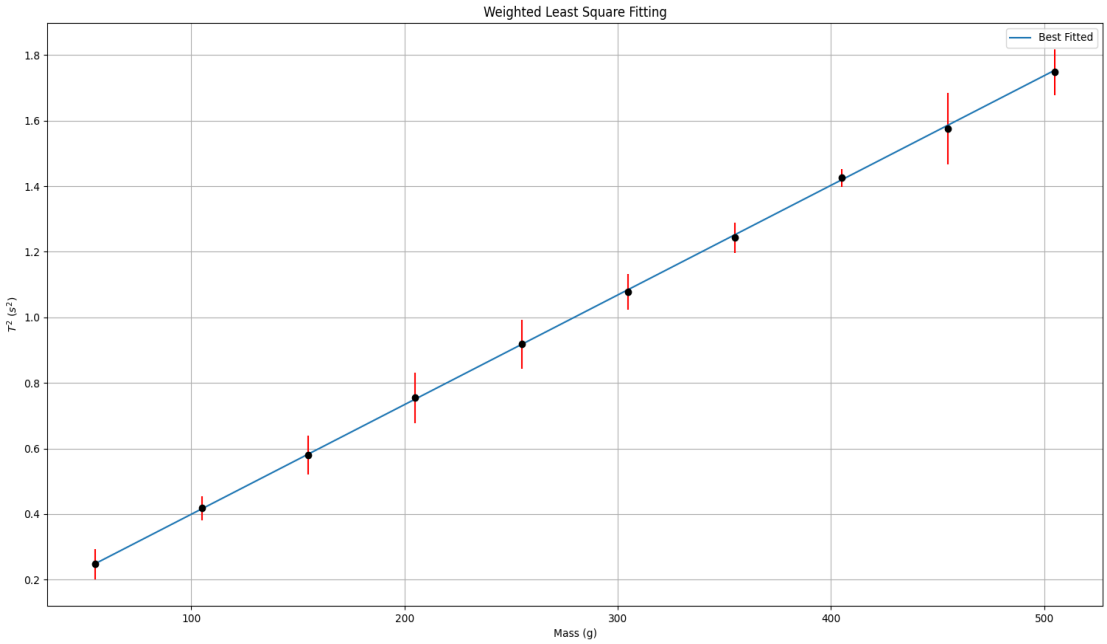

Figure 2: *Weighted Least Squares Fitting*

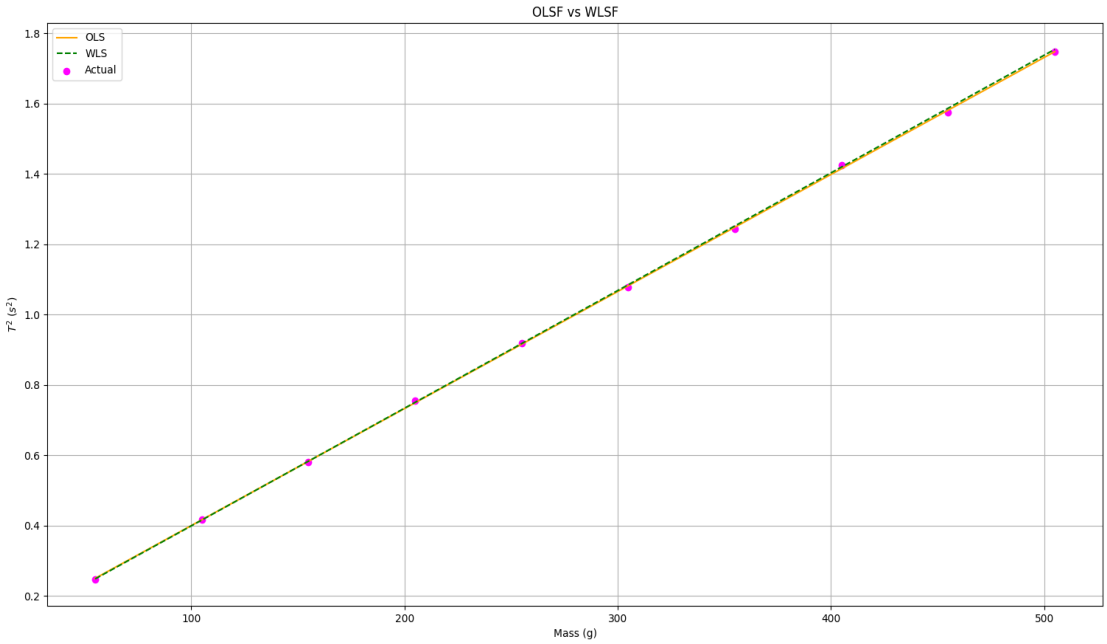

Figure 3: *Comparison of Ordinary Least Squares Fitting and Weighted Least Squares Fitting*

	PARAMTER	OLSF	WLSF	lingress fn
0	SLOPE	3.326653e-03+0.000000e+00j	0.003345	0.003327
1	INTERCEPT	6.742589e-02+0.000000e+00j	0.064898	0.067426
2	ERROR IN SLOPE	1.233610e-05+0.000000e+00j	0.000107	
3	ERROR IN INTERCEPT	3.881952e-03+0.000000e+00j	0.032921	
4	CORRELATION COEFFICIENT	9.996035e-01+0.000000e+00j	0.999933	0.999945
5	SUM OF RESIDUALS	-2.164935e-15+0.000000e+00j	-0.024732	0.0
6	SUM OF RESIDUAL SQUARES	2.510958e-04+0.000000e+00j	0.000378	0.000251

Figure 4: *Results Obtained for different parameters*

Parameter	OLSF	WLSF	linregress fn
k	11867.309570533085	11803.930677874625	11867.309570533078
Error in k	(44.00708444701446+0j)	376.01594233547087	
m	20.26838853676536	20.160142601678164	20.268388536765094
Error in m	(1.2420849603046908+0j)	10.868822258767166	

Figure 5: *Value of k,m obtained from OLSF, WLSF and linegress along with errors.*

- The value of slope and intercept obtained from my function lsqf and inbuilt linegress are matching.
- The value of slope and intercept obtained from ordinary least square fitting and weighted least squares fitting are almost same.
- The value of k,m obtained from ordinary least square fitting and weighted least squares fitting are shown in table[2]. Although the values of k and m obtained from both methods are same, the error in m and error in k is more from weighted least squares fitting than ordinary least squares fitting.

3 Program

```
'''
NAME - KABIR SETHI
ROLL NO - 2020PHY1097

PARTNER:
NAME - PAWANPREET KAUR
ROLL NO - 2020PHY1092
'''

import numpy as np
import matplotlib.pyplot as plt
import cmath
import csv
from scipy.stats import linregress
from statistics import stdev,variance
import pandas as pd
from prettytable import PrettyTable

print("Name - Kabir Sethi \n Roll No. 2020PHY1097")
#2 PROGRAMMING
#(a)
#Ordinary Least squares fitting
def lsqf(x,y):
    n1=len(x)
    n2=len(y)
    slope=0
    intercept=0
    if n1==n2:
        sigma_xi=0
        sigma_yi=0
        sigma_xi_yi=0
        sigma_xisq=0
        sigma_yisq=0
        sse=0
        Rs=0
        count=0
        SSxx=0
        while count<n1:
            sigma_xi=sigma_xi+x[count]                              #SUM OF ALL X
   ELEMENTS
            sigma_yi=sigma_yi+y[count]                              #SUM OF ALL Y
   ELEMENTS
            sigma_xi_yi=sigma_xi_yi+x[count]*y[count]               #SUM OF X*Y ELEMENTS
            sigma_xisq=sigma_xisq+x[count]**2                       #SUM OF X**2
            sigma_yisq=sigma_yisq+y[count]**2                       #SUM OF Y**2
            count=count+1
        slope=(sigma_xi*sigma_yi-n1*sigma_xi_yi)/(sigma_xi**2-n1*sigma_xisq)
          #CALCULATES SLOPE
        intercept=(sigma_xisq*sigma_yi-sigma_xi*sigma_xi_yi)/(n1*sigma_xisq-sigma_xi
   **2)    #CALCULATESINTERCEPT
        r=cmath.sqrt(((sigma_xi_yi)**2)/(sigma_xisq*sigma_yisq)) #COEFFICIENT OF
   CORRELATION
        c=np.array([intercept]*n1)
        xm=np.dot(slope,x)
        y_calc=xm+c
        x_bar=np.mean(x)        #MEAN OF X
        y_bar=np.mean(y)        #MEAN OF Y
        err_y=y-y_calc
        for i in range(len(y)):
            sse+=(y[i]-y_calc[i])**2  #SUM OF RESIDUAL SQUARES
```

```
            Rs+=(y[i]-y_calc[i])       #SUM OF RESIDUALS
        SSxx = sigma_xisq - ((sigma_xi)**2/n1)
        std_slope = cmath.sqrt((sse)/(SSxx * (n1-2)))                    #standard
    deviation of slope
        std_intercept = np.sqrt(((std_slope)**2)*(sigma_xisq/n1))        #standard
    deviation of intercept
        return y_calc,[slope,intercept,std_slope,std_intercept,r,Rs,sse]
        # y_calc,slope,intercept,std_slope,std_intercept,r=#COEFFICIENT OF
    CORRELATION,Rs=SUM OF RESIDUALS,sse=SUM OF RESIDUAL SQUARES

#Weighted Least Squares Fitting
def wlsf(x,y,w):
    n1=len(x)
    n2=len(y)
    slope=0                          #slope
    intercept=0                 #intercept
    e_s=0                            #error in slope
    e_i=0                            #error in intercept
    i=0
    r = 0                            # correlation  coefficient
    sse=0     #SUM OF RESIDUAL SQUARES
    Rs=0      #SUM OF RESIDUALS
    if n1==n2 and n1>3:
        S_wi_xi=0
        S_wi_yi=0
        S_wi_xi_yi=0
        S_wi_xisq=0
        S_wi=0
        x_mean = 0
        y_mean = 0
        Sxx =0
        Syy=0
        Sw=sum(w)
        while i < n1:
                    x_mean += x[i]*w[i]/Sw
                    y_mean += y[i]*w[i]/Sw
                    i = i+1
        count=0
        while count<n1:
            S_wi_xi=S_wi_xi+ x[count]*w[count]
            S_wi_yi= S_wi_yi+y[count]*w[count]
            S_wi_xi_yi=S_wi_xi_yi+ x[count]*y[count]*w[count]
            S_wi_xisq=S_wi_xisq+(x[count]**2)*w[count]
            S_wi=S_wi+w[count]
            Sxx = Sxx + w[count]*(x[count]- x_mean)**2
            Syy =  Syy +w[count]*(y[count] - y_mean)**2
            count+=1


        # SLOPE , INTERCEPT AND CORRELATION COEFFICIENT
        intercept=(S_wi_xisq*S_wi_yi-S_wi_xi*S_wi_xi_yi)/(S_wi*S_wi_xisq-S_wi_xi**2)
        slope=(S_wi*S_wi_xi_yi-S_wi_xi*S_wi_yi)/(S_wi*S_wi_xisq-S_wi_xi**2)
        r = (slope*np.sqrt(Sxx))/(np.sqrt(Syy))


        #DEFINING CORRESPONDING BEST FITTED Y VALUE FOR X

        c=np.array([intercept]*n1)
        xm=np.dot(slope,x)
        y_calc=xm+c                      # Best Fitted y

        # ERROR IN SLOPE , INTERCEPT
```

```
115
116         e_s=((S_wi)/(S_wi*S_wi_xisq - S_wi_xi**2))**(1/2)
117         e_i=((S_wi_xisq)/(S_wi*S_wi_xisq-S_wi_xi**2))**(1/2)
118         for i in range(len(y)):
119             sse+=(y[i]-y_calc[i])**2  #SUM OF RESIDUAL SQUARES
120             Rs+=(y[i]-y_calc[i])       #SUM OF RESIDUALS
121
122
123         return y_calc,[slope,intercept,e_s,e_i,r,Rs,sse]
124         #y_calc,slope,intercept,e_s=error in slope,e_i=error in intercept,r=
    correlation coefficient,Rs=sum of residuals,sse=sum of residual squares
125
126 #d
127 # (i)
128 m=[];t1=[];t2=[];t3=[];t4=[];t5=[];t6=[];t7=[];t8=[];t9=[];t10=[]
129 with open('1097.csv','r') as csv_file:
130     csv_reader = csv.reader(csv_file)       #reading the values from csv file and
    making list of them.
131     next(csv_reader)
132     for line in csv_reader:
133         m.append(line[1])
134         t1.append(line[2])
135         t2.append(line[3])
136         t3.append(line[4])
137         t4.append(line[5])
138         t5.append(line[6])
139         t6.append(line[7])
140         t7.append(line[8])
141         t8.append(line[9])
142         t9.append(line[10])
143         t10.append(line[11])
144
145 M=[];T1=[];T2=[];T3=[];T4=[];T5=[];T6=[];T7=[];T8=[];T9=[];T10=[]
146 for i in range(len(m)):             #to convert elements of list from str into float
147     M.append(float(m[i]))
148     T1.append((float(t1[i])**2))
149     T2.append((float(t2[i])**2))
150     T3.append((float(t3[i])**2))
151     T4.append((float(t4[i])**2))
152     T5.append((float(t5[i])**2))
153     T6.append((float(t6[i])**2))
154     T7.append((float(t7[i])**2))
155     T8.append((float(t8[i])**2))
156     T9.append((float(t9[i])**2))
157     T10.append((float(t10[i])**2))
158
159 T=[]
160
161 for i in range(0,10):
162     s1=np.mean([T1[i],T2[i],T3[i],T4[i],T5[i],T6[i],T7[i],T8[i],T9[i],T10[i]])
163     T.append(s1)
164
165 T=np.array(T)
166
167 hh=np.array([np.array(T1),np.array(T2),np.array(T3),np.array(T4),np.array(T5),np.
    array(T6),np.array(T7),np.array(T8),np.array(T9),np.array(T10)]).reshape(10,10)
168 dd=hh.T
169 w = []
170 err = []
171 for i in range(0,10):
172   w.append(1/stdev(dd[i])**2)
173   err.append(stdev(dd[i]))
```

```
s_s=np.column_stack((M,T,w))
np.savetxt("1097.txt", s_s,header = "xi, yi, wi")

#(ii)
pr1=lsqf(M,T)
pr=wlsf(M,T,w)

#(iii)
plt.scatter(M,T,label="Actual",c="magenta")
plt.plot(M,pr1[0],label="Best Fitted")
plt.xlabel("Mass (g)")
plt.ylabel("$T^2$ ($s^2$)")
plt.title("Ordinary Least Square Fitting")
plt.legend()
plt.grid()
plt.savefig("1097_OLSF.pdf")
plt.show()


plt.errorbar(M,T,yerr=err,xerr=None,fmt='o',ecolor = 'red',color='black')
plt.plot(M,pr[0],label="Best Fitted")
plt.xlabel("Mass (g)")
plt.ylabel("$T^2$ ($s^2$)")
plt.title("Weighted Least Square Fitting")
plt.legend()
plt.grid()
plt.savefig("1097_WLSF.pdf")
plt.show()

plt.scatter(M,T,label="Actual",c="magenta")
plt.plot(M,pr1[0],label="OLS",c="orange")
plt.plot(M,pr[0],label="WLS",linestyle="dashed",c="green")
plt.xlabel("Mass (g)")
plt.ylabel("$T^2$ ($s^2$)")
plt.title("OLSF vs WLSF")
plt.legend()
plt.grid()
plt.savefig("1097_WLSF_vs_OLSF.pdf")
plt.show()
sl, intec, r, p, se =linregress(M, T)

#(e)
k_ols = 4*np.pi*np.pi/pr1[1][0]
k_wls = 4*np.pi*np.pi/pr[1][0]
k_linregress = 4*np.pi*np.pi/sl
m_ols = (k_ols*pr1[1][1])/(4*np.pi*np.pi)
m_wls = (k_wls*pr1[1][1])/(4*np.pi*np.pi)
m_linregress = (k_linregress*intec)/(4*np.pi*np.pi)
err_k_ols=(pr1[1][2]/pr1[1][0])*k_ols
err_k_wls=(pr[1][2]/pr[1][0])*k_wls
err_m_ols=((pr1[1][3]/pr1[1][1])+(err_k_ols/k_ols))*m_ols
err_m_wls=((pr[1][3]/pr[1][1])+(err_k_wls/k_wls))*m_wls
header = ["Parameter", "OLSF", "WLSF", "linregress fn"]
myTable = PrettyTable(header)
myTable.add_row(["k",k_ols,k_wls,k_linregress ])
myTable.add_row(["Error in k",err_k_ols,err_k_wls, ""])
myTable.add_row(["m",m_ols,m_wls,m_linregress])
myTable.add_row(["Error in m",err_m_ols,err_m_wls,""])
print(myTable)

#d(iv),(f)

```

```
y_calc_inbuilt=[]
for i in range(0,10):
    y_calc_inbuilt.append(sl*M[i] +intec)
S_RS=0;S_R=0
for i in range(10):
    S_RS+=(T[i]-y_calc_inbuilt[i])**2  #SUM OF RESIDUAL SQUARES
    S_R+=(T[i]-y_calc_inbuilt[i])       #SUM OF RESIDUALS
array_3=[sl,intec,"","",r,S_R,S_RS]


para=["SLOPE","INTERCEPT","ERROR IN SLOPE","ERROR IN INTERCEPT","CORRELATION
   COEFFICIENT","SUM OF RESIDUALS","SUM OF RESIDUAL SQUARES"]
DATA={"PARAMTER":para,"OLSF":pr1[1],"WLSF":pr[1],"lingress fn":array_3}
print(pd.DataFrame(DATA))
```