Predicting women's ovulation day using Linear Regression

1. Introduction

The menstrual cycle is one of the most critical indicators of a woman's health. The process starts with the first day of one period and ends with the first day of the next period. A healthy woman will have a 22-35 day cycle, depending on their own body. During this cycle, the fertile window, which generally lasts for seven days, including five days before ovulation day, the ovulation day and the day after, is the time woman can easily get pregnant. It varies among different people. Therefore, accurately predicting ovulation day is essential for family planning and women's health.

In this project, I will build a model to predict ovulation day based on the length of the menstrual cycle using the Linear Regression model. Session 2 discusses the problem formulation and explores the dataset. Next, I go deep into the actual methods in session three and present the result in session 4.

2. Problem formulation

The ovulation day generally happens in the middle of the cycle length. However, the menstrual cycle length and the fertile window period differ among women and time phases. It is reasonable to build a machine learning model based on its linear characteristic to forecast ovulation day.

The dataset contains crucial information about the menstrual cycles, including lengths, cycle peaks, estimated ovulation day, luteal phases, etc. All data inputs are integer as it describes the date in a cycle.

3. Method

3.1. Dataset:

The dataset is collected from Menstrual Cycle Data by Richard J. Fehring, Marquette University (cited in the References below). They were published in

2012 under the grand title Randomized Comparison of Two Internet–Supported Natural Family Planning Methods.

Each row in the dataset describes the menstrual cycle information from multiple women in multiple periods. Ovulation day generally occurs in the middle of the cycle length and at the end of the fertility window. Additionally, a woman can self-track her period length and fertility period based on the changes in the body. Therefore, the ovulation day depends on the cycle length and fertility length. I would choose the cycle length and the total fertility days as *features* and the ovulation day as *labels*.

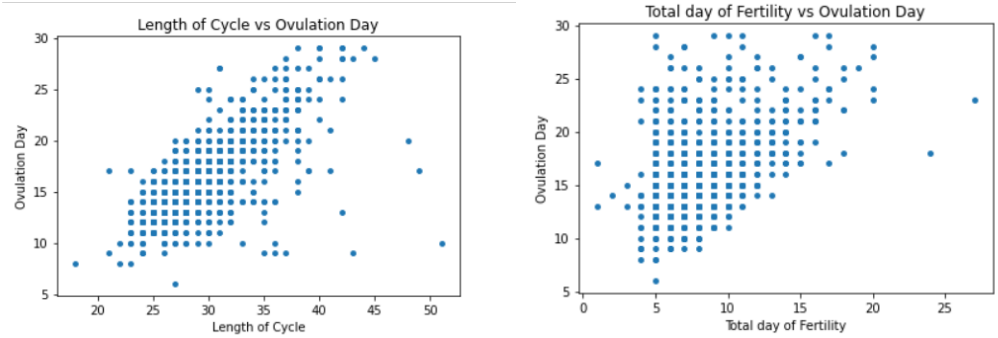


Fig1. Linearity check between features and labels

The raw dataset contains some Null (not given) datapoints in the columns 'OvulationDay'. After processing to clean the data frame, I removed all the Null data points which have missing values on the OvulationDay and the negative LengthOfCycle. As a result, the data frame contains 1491 data points. The data type is integer.

```
Data columns (total 3 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   LengthofCycle           1491 non-null   int64
 1   TotalDaysofFertility    1491 non-null   int64
 2   EstimatedDayofOvulation 1491 non-null   int64
dtypes: int64(3)
```

The dataset is split into a training set, a validation set and a test set according to the ratio of 5:3:2. With this ratio, the model still has enough data to be built, and the validation set is not too small to avoid the overfitting problem.

3.2 Linear regression model:

As can be seen in the data point visualisation above, there is a linear relationship between the length of the menstrual cycle and the ovulation day. So I choose the linear regression model, a supervised learning model.

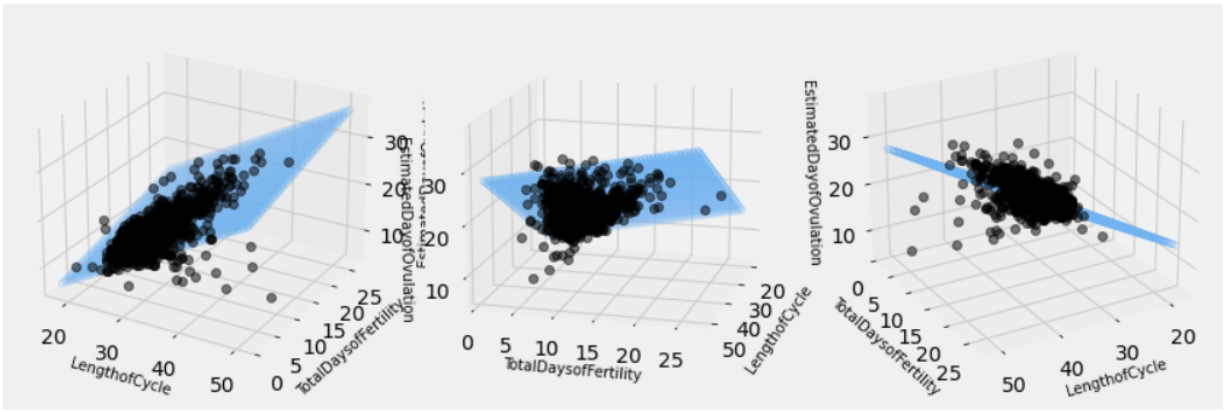


Fig2. Visualise the trained linear regression model.

I believe that the data input (length cycle and fertility cycle) are normally distributed around the mean value as there is a common route for a certain person these days. The mean square error is adapted for the model loss function.

3.3 Polynomial regression model:

A menstrual cycle differs among women, with an average of 22-35 days. Moreover, the cycle length and fertility length can be distinct among different months of a certain person due to stress, unhealthy eating habit, etc. This gap is quite big for a linear model to work well. Let us try the polynomial regression model, which will be fitted by polynomial features and then apply the linear regression model to them. Different degrees of polynomial features are tried to find the best fit model.

The same loss function is used for this model, the mean square error because the linear regression will be applied to the model in the later stage.

4. Result:

The training error and validation error of the model mentioned above are presented in table 1.

Models (with degree)	Training error	Validation error	Test error
Linear regression	4.7535	6.0066	5.9629
Polynomial regression (2)	4.8906	4.6418	5.5301
Polynomial regression (3)	4.6105	4.6438	4.9540
Polynomial regression (4)	4.5951	4.6374	5.0756
Polynomial regression (5)	4.5510	4.6342	15.1434
Polynomial regression (6)	4.5194	5.7299	11.2264

Table 1. Training error and validation error on model

As can be seen from this table, the higher the degree of the polynomial regression, the lower the training error and the higher validation and test error. This is a sign of overfitting in a higher degree (from degree 5 onward).

The most reasonable model for the problem would be the polynomial regression with degree 3, as it performs the best result among all the models I have tried. Its error is quite low compared to other results, especially the test error.

5. Conclusions:

This report is aimed to build a supervised machine-learning model to predict women's ovulation day *(label)* based on the menstrual cycle length and total days of fertility *(features)*. After trying two different models, linear regression and polynomial regression, I believe the polynomial regression with degree 3 fits the problem. Currently, the model works quite well. As the future directions of this problem, I consider choosing more features like age range, health condition, etc., to increase the model's accuracy.

References

Atha, R. (2020, November 20). *Multi-Linear Regression Using Python | by Rafi Atha | The Startup*. Medium. https://medium.com/swlh/multi-linear-regression-using-python-44bd0d10082d

Blake, K., & Rapp, A. (n.d.). *Relationship Between the Menstrual Cycle and Timing of Ovulation Revealed by New Protocols: Analysis of Data from a Self-Tracking Health App*. NCBI. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5725625/

Clark, Karen; Jain, Mridul; Messa, Araya; Le, Vinh; and Larson, Eric C. (n.d.). Open Cycle: Forecasting Ovulation for Family Planning. *SMU Data Science Review*, *1*. https://scholar.smu.edu/datasciencereview/vol1/iss1/2

Fehring, R. J. (n.d.). *"Menstrual Cycle Data"*. e-Publications@Marquette. https://epublications.marquette.edu/data_nfp/7/

Johns Hopkins Medicine. (n.d.). *Menstrual Cycle: An Overview*. Johns Hopkins Medicine. Retrieved September 23, 2022, from https://www.hopkinsmedicine.org/health/wellness-and-prevention/menstrual-cycle-an-overview

Jung, A. (2022). *Machine Learning: The Basics*. Springer Nature Singapore.

Appendix

October 9, 2022

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.preprocessing import PolynomialFeatures    # function to generate␣
 ↪polynomial and interaction features
from sklearn.linear_model import LinearRegression    # classes providing Linear␣
 ↪Regression with ordinary squared error loss and Huber loss, respectively
from sklearn.metrics import mean_squared_error    # function to calculate mean␣
 ↪squared error
```

Cleaning the dataset

[2]:
```
periodRawData = pd.read_csv('CycleData.csv')
periodRawData.head()
```

[2]:

	ClientID	CycleNumber	Group	CycleWithPeakorNot	ReproductiveCategory	\
0	nfp8122	1	0	1	0	
1	nfp8122	2	0	1	0	
2	nfp8122	3	0	1	0	
3	nfp8122	4	0	1	0	
4	nfp8122	5	0	1	0	

	LengthofCycle	MeanCycleLength	EstimatedDayofOvulation	LengthofLutealPhase	\
0	29	27.33	17	12	
1	27		15	12	
2	29		15	14	
3	27		15	12	
4	28		16	12	

	FirstDayofHigh	…	Method	Prevmethod	Methoddate	Whychart	Nextpreg	\
0	12	…	9			2	7	
1	13	…						
2		…						
3	13	…						
4	12	…						

	NextpregM	Spousesame	SpousesameM	Timeattemptpreg	BMI

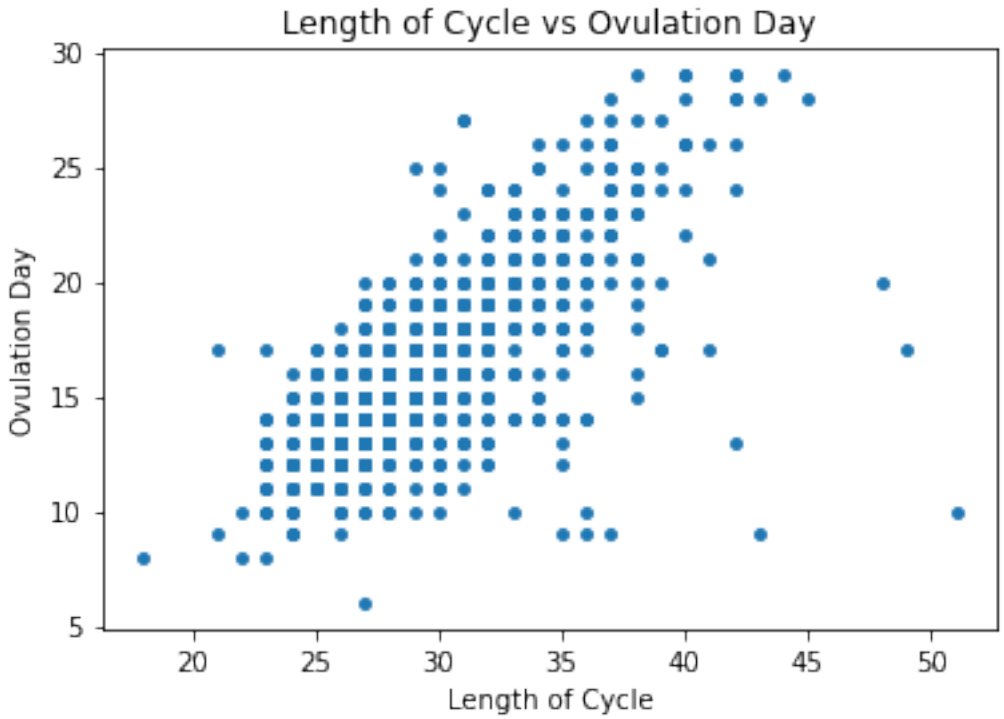
```
0         7          1           1                0  21.254724111867
1
2
3
4

[5 rows x 80 columns]
```

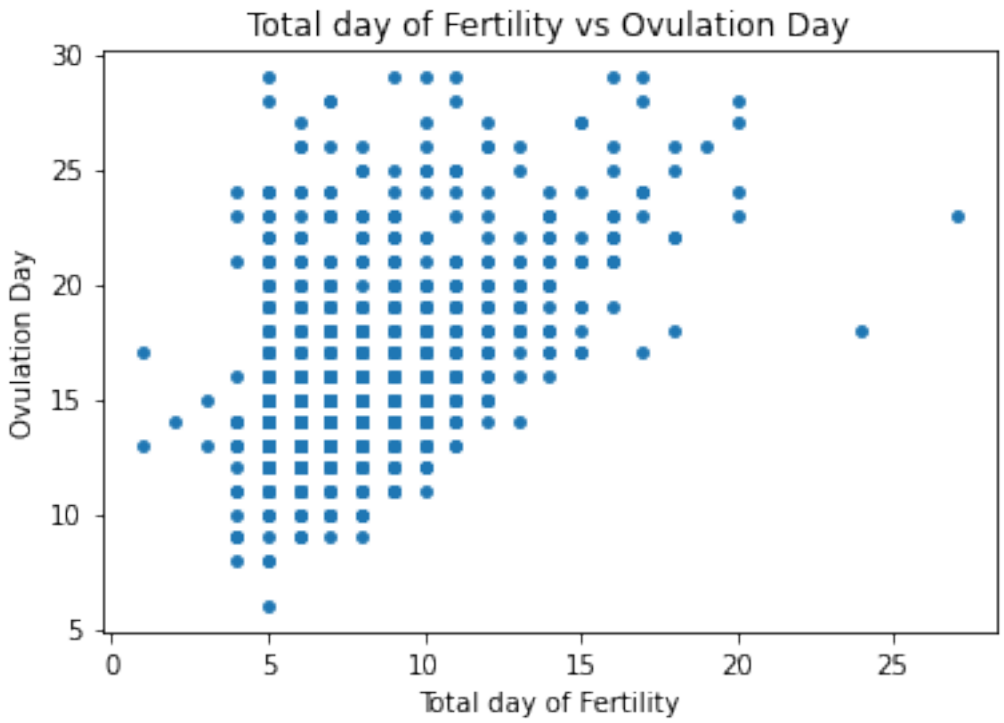
[3]:
```
# filtering Null datapoints
data = periodRawData[periodRawData['LengthofCycle'] > 0]
data = data[data['EstimatedDayofOvulation'] != ' ']
data = data[data['TotalDaysofFertility'] != ' ']
data = data[['LengthofCycle', 'TotalDaysofFertility',␣
 ↪'EstimatedDayofOvulation']]
data = data.astype(int)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1491 entries, 0 to 1662
Data columns (total 3 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   LengthofCycle            1491 non-null   int64
 1   TotalDaysofFertility     1491 non-null   int64
 2   EstimatedDayofOvulation  1491 non-null   int64
dtypes: int64(3)
memory usage: 46.6 KB
```

[4]:
```
#Check the linearity between
## The cycle length and the ovulation day
## the total days of fertility and the ovulationday

x1 = data['LengthofCycle']
x2 = data['TotalDaysofFertility']
y = data['EstimatedDayofOvulation']

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.scatter(x1,y, s=15)
ax.set_xlabel('Length of Cycle')
ax.set_ylabel('Ovulation Day') #
ax.set_title('Length of Cycle vs Ovulation Day')

fig = plt.figure()
ax2 = fig.add_subplot(1, 1, 1)
ax2.scatter(x2,y, s=15)
ax2.set_xlabel('Total day of Fertility')
```

```
ax2.set_ylabel('Ovulation Day') #
ax2.set_title('Total day of Fertility vs Ovulation Day')

plt.show()
```





[5]:
```
#Generate datapoints with feature X = 'LengthOfCycle' and label y =␣
 ↪'EstimatedDayOfOvulation'
X = data[['LengthofCycle', 'TotalDaysofFertility']].to_numpy().reshape(-1,2)
y = data['EstimatedDayofOvulation'].to_numpy()
print(X.shape)
print(X)
```

```
(1491, 2)
[[29  9]
 [27  6]
 [29  5]
 …
 [29 10]
 [28  9]
 [28  9]]
```

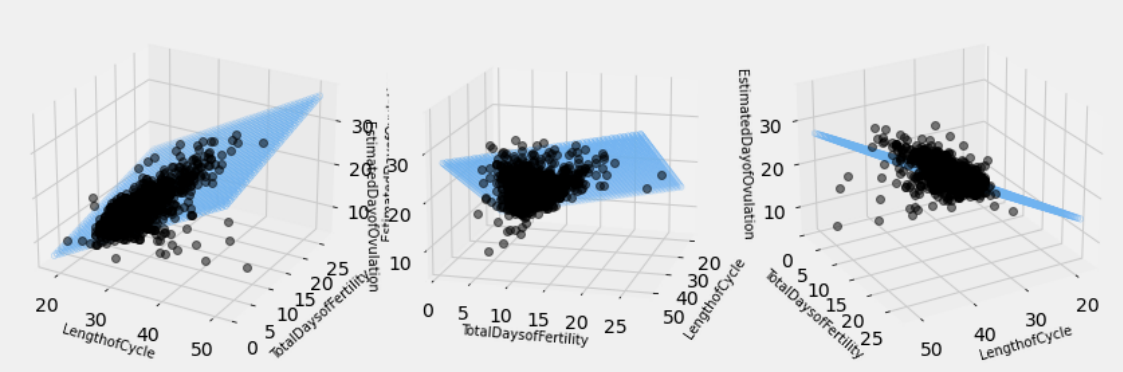
[6]:
```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split

X_rest, X_test, y_rest, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=0)
X_train, X_val, y_train, y_val = train_test_split(X_rest, y_rest, test_size = 0.
 ↪375, random_state = 0)
```

[7]:
```
#Linear Regression
regr = LinearRegression()
regr.fit(X_train,y_train)

#predict label values based on features and calculate the training error
y_pred = regr.predict(X_train)
tr_error = mean_squared_error(y_train, y_pred)

y_pred_val = regr.predict(X_val)
val_error = mean_squared_error(y_val, y_pred_val)

y_pred_test = regr.predict(X_test)
test_error = mean_squared_error(y_test, y_pred_test)

print('The training error is: ', tr_error)
print('The validation error is: ', val_error)
print('The test error is: ', test_error)

print("w1 = ", regr.coef_)   # print the learnt w1
print("w0 = ",regr.intercept_) # print the learnt w0
```

```
The training error is:  4.955716823991903
The validation error is:  4.627561516110277
The test error is:  5.962948863296241
w1 =  [0.63999172 0.32469795]
w0 =  -5.293805661889895
```

[8]:
```
#VISUALIZE THE LINEAR REGRESSION MODEL

# Create range for each dimension
x1 = X[:, 0]
x2 = X[:, 1]
z = y

x1_pred = np.linspace(x1.min(), x1.max())  # range of price values
x2_pred = np.linspace(x2.min(), x2.max())  # range of advertising values
xx1_pred, xx2_pred = np.meshgrid(x1_pred, x2_pred)
model_viz = np.array([xx1_pred.flatten(), xx2_pred.flatten()]).T

# Predict using model built on previous step
predicted = regr.predict(model_viz)


# Plot model visualization
plt.style.use('fivethirtyeight')

fig = plt.figure(figsize=(12, 4))
```

```
ax1 = fig.add_subplot(131, projection='3d')
ax2 = fig.add_subplot(132, projection='3d')
ax3 = fig.add_subplot(133, projection='3d')

axes = [ax1, ax2, ax3]

for ax in axes:
    ax.plot(x1, x2, z, color='black', zorder=15, linestyle='none', marker='o',␣
 ↪alpha=0.5)
    ax.scatter(xx1_pred.flatten(), xx2_pred.flatten(), predicted,␣
 ↪facecolor=(0,0,0,0), s=20, edgecolor='#70b3f0')
    ax.set_xlabel('LengthofCycle', fontsize=10)
    ax.set_ylabel('TotalDaysofFertility', fontsize=10)
    ax.set_zlabel('EstimatedDayofOvulation', fontsize=10)
    ax.locator_params(nbins=4, axis='x')
    ax.locator_params(nbins=5, axis='x')

ax1.view_init(elev=25, azim=-60)
ax2.view_init(elev=15, azim=15)
ax3.view_init(elev=25, azim=60)


fig.tight_layout()
```



[9]:
```
#polynomial regression
for i in range(2,7):
    poly = PolynomialFeatures(i)
    X_train_poly = poly.fit_transform(X_train)
    lingr = LinearRegression()
    lingr.fit(X_train_poly, y_train)
    y_pred_poly = lingr.predict(X_train_poly)
    tr_error_poly = mean_squared_error(y_train, y_pred_poly)
```

```
    X_val_poly = poly.fit_transform(X_val)
    y_pred_val_poly = lingr.predict(X_val_poly)
    val_error_poly = mean_squared_error(y_val, y_pred_val_poly)

    X_test_poly = poly.fit_transform(X_test)
    y_pred_test_poly = lingr.predict(X_test_poly)
    test_error_poly = mean_squared_error(y_test, y_pred_test_poly)

    print('Current polynomial degree is: ', i)
    print('The training error is: ', tr_error_poly)
    print('The validation error is: ', val_error_poly)
    print('The test error is: ', test_error_poly)
    print("w1 = ", lingr.coef_)   # print the learnt w1
    print("w0 = ", lingr.intercept_) # print the learnt w0
    print('')
```

```
Current polynomial degree is:  2
The training error is:  4.89059966414253
The validation error is:  4.641876918457889
The test error is:  5.530061283001103
w1 =  [ 0.          1.09972034 -0.03619537 -0.00883192  0.01300533 -0.00290778]
w0 =  -11.042830197977263

Current polynomial degree is:  3
The training error is:  4.610519329132766
The validation error is:  4.643837327978654
The test error is:  4.954031758818262
w1 =  [ 0.00000000e+00 -5.56705076e+00  1.85720707e+00  2.15701816e-01
 -1.58106961e-01  1.05513291e-01 -2.43214710e-03  2.68509849e-03
 -1.09816670e-03 -2.07340478e-03]
w0 =  55.0524616662161

Current polynomial degree is:  4
The training error is:  4.595077977356042
The validation error is:  4.637413077764491
The test error is:  5.075637871748076
w1 =  [ 0.00000000e+00  7.52177018e-01  4.93966164e+00 -6.31083445e-02
 -2.57789066e-01 -2.19092196e-01  3.21195054e-03  2.20724677e-03
  1.10750914e-02  6.08262961e-03 -4.25103520e-05  1.55048867e-05
 -6.38297939e-05 -2.32132150e-04 -2.64027845e-05]
w0 =  -1.6774153382923949

Current polynomial degree is:  5
The training error is:  4.551028174711112
The validation error is:  4.634197449064575
The test error is:  15.143448012530293
```

```
w1 =  [ 0.00000000e+00  3.06172717e+01  3.87401002e+01 -1.59957854e+00
 -4.89557111e+00  4.74096688e-01  3.79263322e-02  2.49020950e-01
 -1.00142222e-01  5.81455005e-02 -3.50864912e-04 -5.44634988e-03
  2.02852992e-03  3.07084526e-03 -5.37546415e-03  7.05596811e-07
  3.75161209e-05  3.06892710e-05 -1.75535378e-04  1.87447563e-04
 -2.85175704e-06]
w0 =  -219.09550271056096

Current polynomial degree is:  6
The training error is:  4.519427409141918
The validation error is:  4.729979500337141
The test error is:  11.226481612393805
w1 =  [-1.95712256e-03 -3.10352171e+02  3.60200996e+02  3.06348013e+01
 -3.94466734e+01 -2.60502175e+01 -1.43080940e+00  1.37490289e+00
  3.35690093e+00 -1.92645248e-01  3.45488862e-02 -9.61449015e-03
 -1.56122315e-01  1.02866872e-02  4.71116169e-03 -4.15026158e-04
 -3.14377266e-04  3.00698912e-03  1.88511411e-04 -5.85146031e-04
  9.44600116e-05  1.95454498e-06  4.21481302e-06 -1.96493691e-05
 -7.88388023e-06  9.17386543e-06  1.68887855e-06 -1.84448402e-06]
w0 =  1058.2775203158599
```