Computer Simulation assignment 3- Least Square Fits

Professor Stefan Hutzler

Alexandra Mulholland 17336557 23/10/19

Abstract

The purpose of this investigation was to fit a data set of number of radioactive decays versus time using least square fitting in Python. There are 12 points in the data set - comparatively smaller to usual radioactivity samples, and so is subject to scattering effects. This is evident in Figure 1 in the results section, where there are irregularities in the exponential decay curve described by equation [1.0]. And again, when plotting the natural logarithm of the number of decays, which should be a straight line of negative gradient (equation [2]). Values for the intercept and gradient of this plot were approximated by eye aided by a line of best fit and found to be 3.4 and 0.0256 respectively. Fit parameters a and b were found by linear equations ([3.1] and [3.2]). The value of a was 3.340898909967487 and that of b was -0.026703363333904266. It was then that the linear regression fit was plotted with the natural logarithmic scale of the data (figure 4) using these values for the parameters, followed by a linear-scaled plot of the data (figure 5). This is the exponential radioactive decay function, and so a third parameter, c (representing N_o), was introduced in order to define this function. This was found to accurately describe the data trend, however, a non-linear least square fit of the data using scipy was more representative of the results- emphasising a clear exponential decay. The values for the parameters b and c were initialised to zero, and the values at which they described the decay accurately were b=-0.03252979329724948 and c=35.36563360300477.

Mathematical Background

The equation for radioactive decay is as follows:

$$N(t) = N_o e^{-\lambda t} \qquad [1.0]$$

Where $N(t)$ is the number of radioactive decays at a time t, N_o is the number of decays initially and λ is the decay constant.

Within this investigation, the number of measurements was 12, which, for radioactive decay, is relatively small if one were to get an accurate representation of the exponential function. Therefore, there will be scattering which significantly affect the graph. Linear regression allows the modelling of the relationship between the dependent and independent variable, correcting for the scattering which may occur due to experimentation.

Computing the sum of the squared residuals for the total number of data points N (not to be confused with the number of radioactive decays), which is the sum of the 'functional guess' subtract the actual data point, squared. It is here that fit parameters a and b are chosen in order to minimise the value of this sum, denoted by S:

$$S = \sum_{i=1}^{N} (f(x_i)_{p1,\dots,pk} - y_i)^2$$

The minimisation of this sum requires $\frac{\partial S}{\partial a}$ and $\frac{\partial S}{\partial b}$ to equal zero. Once this and the second derivatives of each are computed (in order to confirm these are minimum values) we are left with 2 linear equations with 2 unknowns: fit parameters a and b. These equations are shown below:

$$aN + b\sum_{i=1}^{N} x_i = \sum_{i=1}^{N} y_i \qquad [1.1]$$

$$a\sum_{i=1}^{N} x_i + b\sum_{i=1}^{N} {x_i}^2 = \sum_{i=1}^{N} x_i y_i \qquad [1.2]$$

In our case, our functional guess assumes a linear relationship of the form $f(t) = a + bt$. Therefore, solving equation [1] to compute a linear relationship dependent on t, we get:

$$lnN(t) = \ln(N_o) - \lambda t \qquad [2]$$

Here, fit parameters a and b take the form $\ln(N_o)$ and $-\lambda$ respectively. Thus equations [1.1] and [1.2] become:

$$(\ln N_o)\mathrm{N} - \lambda\sum_{i=1}^{N} t_i = \sum_{i=1}^{N} \ln N_i \qquad [1.1i]$$

$$(\ln N_o)\sum_{i=1}^{N} t_i - \lambda\sum_{i=1}^{N} {t_i}^2 = \sum_{i=1}^{N} (\ln N_i) t_i \qquad [1.2i]$$

Solving for $(\ln N_o)$ and λ, labelling them as a and b for simplicity, we get:

$$a = \frac{\sum_{i=1}^{N} (\ln N_i) t_i \sum_{i=1}^{N} t_i - \sum_{i=1}^{N} (\ln N_i) \sum_{i=1}^{N} {t_i}^2}{\sum_{i=1}^{N} t_i \sum_{i=1}^{N} t_i - N\sum_{i=1}^{N} {t_i}^2} \qquad [3.1]$$

$$b = \frac{\sum_{i=1}^{N} \ln(N_i) - aN}{\sum_{i=1}^{N} t_i} \qquad [3.2]$$

These fit parameters were used in part 3 onwards in order to carry out a least square fit of the data.

Method

Part 1 of the investigation involved plotting the data set provided. This plot was number of decays (the dependent variable) versus time (the independent variable), and hence represents the exponential function of equation [1]. The data was presented in python as shown:

```
time = np.array([5,15,25,35,45,55,65,75,85,95,105,115])
Number_of_decays = np.array([32,17,21,7,8,6,5,3,4,1,5,1])
```

Next, in part 2, the data was replotted using a log-lin scale, whereby the natural logarithm of the number of decays was plotted against time, done so by coding `np.log(Number_of_decays)` for the graph. This results in a negative-gradient linear plot represented by equation [2]. By plotting a line of best fit, it was made easier to estimate by eye the intercept and the gradient values, which in this case are $\ln(N_o)$ and $-\lambda$ respectively (i.e. a and b).

Part 3 consisted of carrying out a least square fit of the data. The summations were defined, as shown below, and used to define the parameters a and b in the form of equations [3.1] and [3.2].

```
A = np.sum(time)
B = np.sum(np.square(time))
C = np.sum(np.log(Number_of_decays))
D= np.sum(np.multiply(np.log(Number_of_decays),time))
#defining number of data points N
N=12

a= (D*A-(B*C))/((A*A)-(N*B))
b= (C-(a*N))/(A)
print('Paramater a:',a)
print('Parameter b:',b)

def function(time):
        f = b*time + a
        return f
```

In part 4, the straight line corresponding to the linear regression fit was defined and plotted atop the data plotted in part 2 (The code defining this is shown above). The values printed for the fit parameters were used to compare those estimated in part 2. Then the value for the decay constant λ could be deduced from the magnitude of the gradient.

In part 5, the values for a and b were used to plot a linear-scaled function for the decay, represented by equation [1.0]. Here, a third parameter c was defined, which would take the value N_o. The function was plotted atop the decay function plotted in part 1.

```
(np.exp(3.340898909967487))   where 3.34… is the value of a
c = 28.244504564174118
def function2(time):
        g= c*np.exp(b*time)
        return g
```

Finally, in part 6, a second non-linear least square fit was defined and plotted with the function in part 1, using the scipy library. Initial guesses of the fit parameters b and c were made (0 and 0).

```
def function3(time,c,b):
        h = c*np.exp(b*time)
        return h
```

The fit parameters were then found and printed, then compared with those found using the code previously in the linear regression analysis.

Results

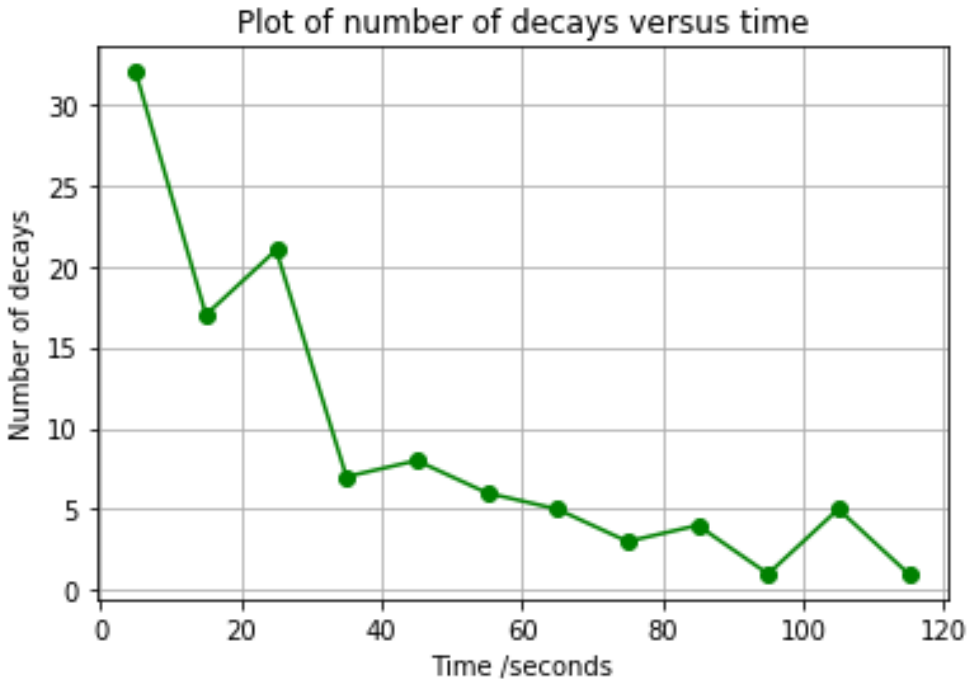

Figure 1

Plot of the given data of number of decays versus the time in seconds.

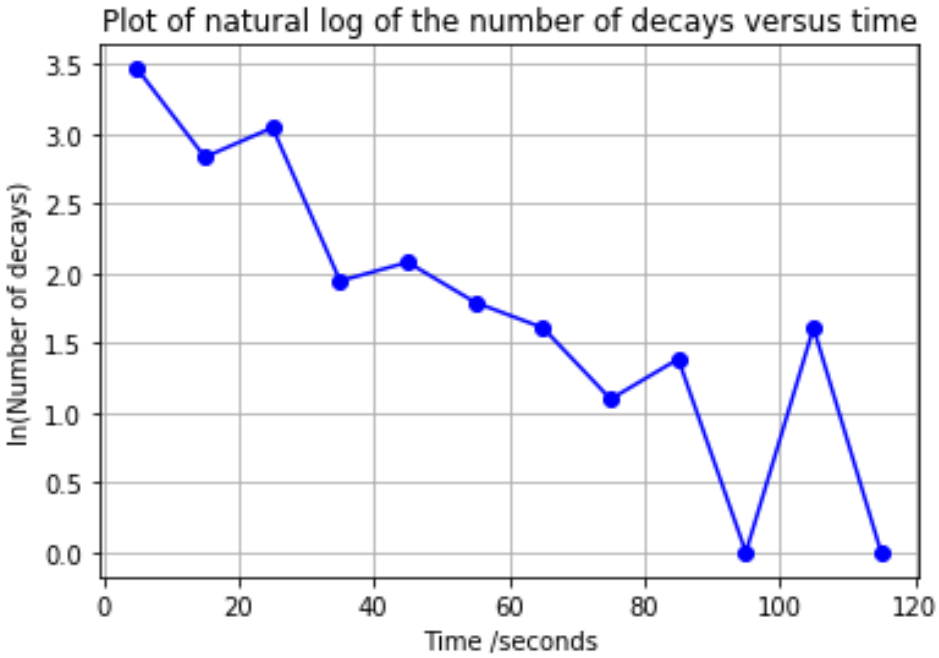

Figure 2

Plot of the natural logarithm of the number of decays versus the time in seconds

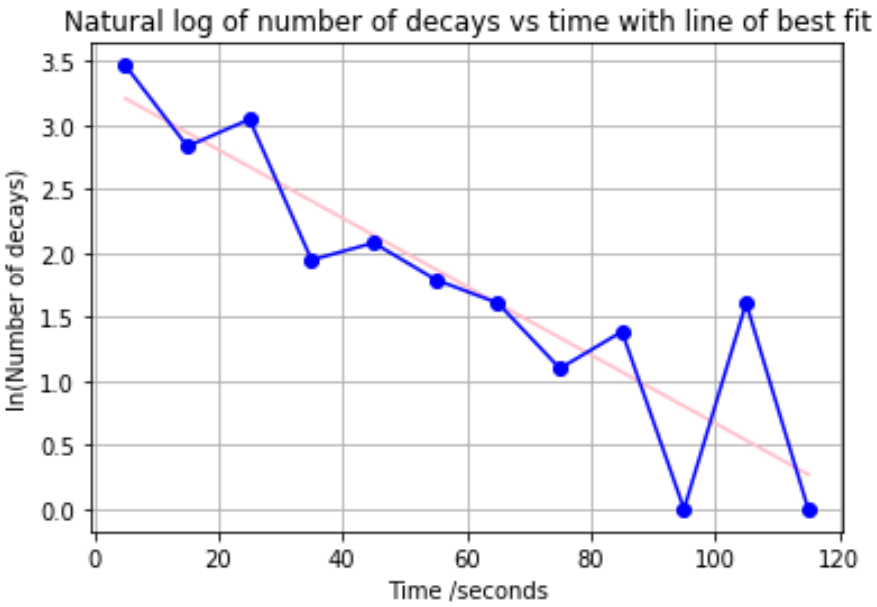

Figure 2.1

Plot of the natural log of the number of decays versus time, with a line of best fit plotted in pink.

Estimated quantity	Value
Gradient	-0.025555555555555554
Intercept	3.4

Figure 2.2

Table showing the estimated values for the gradient and intercept by eye from the line of best fit

Parameter	Value
a	3.340898909967487
b	-0.026703363333904266

Figure 3

Table showing the printed results for the values of the two fit parameters

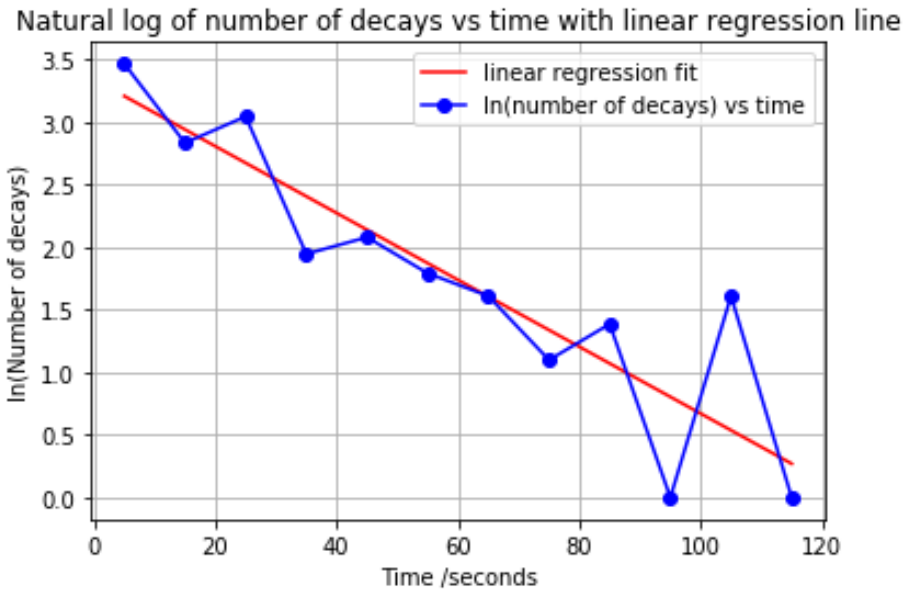

Figure 4

Graph showing the linear regression fit corresponding to the least square fit calculated in part 3.

Parameter	Value
b	-0.03252979329724948
c	35.36563360300477

Figure 5.1

Table showing the values of the parameters b and c found in order to plot figure 5.2 and 6.2

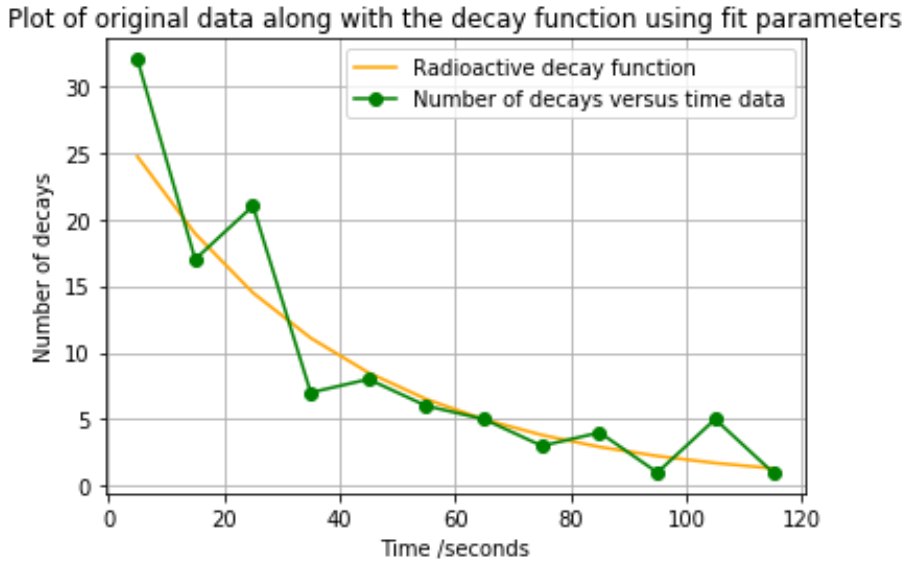

Figure 5.2

Graph showing the original data of number of decays versus time plotted with the exponential decay function corresponding to the values b and c

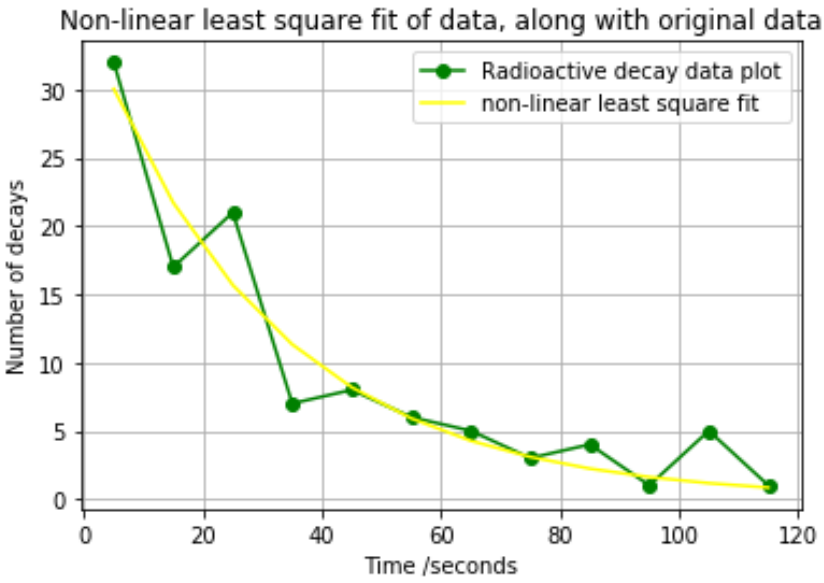

Figure 6.2

Plot of the original data atop a non-linear square fit, graphed using scipy. Again, plotted using parameters b and c

Discussion

Figure 1, showing the graph of the data, is as expected- whereby the relatively small sample size and perhaps experimental error, leads to a large amount of scattering. However, one can see the exponential decay trend described by $N(t) = N_o e^{-\lambda t}$.

Taking the natural logarithm of equation [1.0] and plotting ln $N(t)$ versus time, leads to a graph described by equation [2]. Again, the anomalies are evident, but a straight line of negative gradient describes this plot.

This is shown in figure 2.1, whereby a line of best fit was plotted. From this, the gradient and intercept were able to be estimated by eye. As shown in figure 2.2, the value for the y-axis intercept was estimated as 3.4, and the gradient (-λ) $\frac{3-0.7}{10-100} \approx -0.0256$.

Figure 3 shows the values of fit parameters a and b which, from equations [1.1i] and [1.2i], take the values of -λ and $\ln(N_o)$ respectively. These values are relatively close to the estimated values of the gradient and intercept from part 2. Thus, owing to the reliability of the linear regression method. The code shown in part 3 of 'Method', defining the summations, allows for the simplicity of the calculation of a and b, when inserted into equations [3.1] and [3.2]. The value of the decay constant from this code is the magnitude of b. Therefore $\lambda = 0.026703363333904266 s^{-1}$.

Figure 4 implemented the values of a and b in order to plot the logarithmic function described by equation [2]. Again, the similarity between this and the line of best fit is evident. From this, we can see that the linear regression fit has eliminated the scattering effects and depicted an accurate result for the logarithmic radioactive decay. The code for this function is shown below:

```
def function(time):
        f = b*time + a
        return f
```

Figure 5 depicts the exponential radioactive decay described by equation [1]. Here, a third parameter 'c' was defined, which represents N_o in the equation- i.e. $c = e^a$, with a being the constant in figure 3. The trend is clearly exponential and describes the decay appropriately, however, figure 6 conveys a more representative result.

Figure 6 utilises the scipy library to perform a non-linear least square fit of the data- i.e. a graph which represents equation [1]. The exponential function was defined in terms of b, c and time and the fit performed using the line of code: fitparameter=optimization.curve_fit(function3,time, Number_of_decays, guess)[0]

The values for b and c for which this least-square fitting curve was optimally plotted are shown in Figure 6.1. The value of the initial number of counts, c, was about 35 and so agrees with the trend of the data points, where the initial value was 32.

The values of b calculated in parts 3 and 6 of the investigation are -0.026703363333904266 and -0.03252979329724948 respectively. Differing by approximately 5.83×10^{-3} units, they are also in agreement, where the decay constant can be deduced to have an approximate value of $0.03 s^{-1}$.

Conclusion

Evidently, the least-squared method is effective in correcting for experimental error in data collection. The scattering effect produced is conspicuous in parts 1 and 2. The exponential trend one would expect in part 1 is not entirely prominent, nor in part 2 where one would expect a negatively sloped straight line. The estimated value for the gradient was around -0.0256 $\left(\frac{3-0.7}{10-100}\right)$ and for the intercept was 3.4. Both were deduced through plotting a line of best fit and approximating by eye.

Writing a code in parts 3 and 4 to implement the linear regression of the data, allowed the non-linear fit to be represented more accurately, showing the expected negative straight-line slope. The fit parameter a (i.e. $\ln(N_o)$) was found to be 3.340898909967487, and b -0.026703363333904266. This results in a decay constant value of around $0.026 s^{-1}$.

In part 5, a new parameter was introduced to allow the exponential decay function to be plotted. The value of this c was found to be around 28, using c=e^a indicating a similar value to the initial value of 32 in the original data. Parts 3, 4 and 5 therefore verify the validity of the method, evident by their consistencies.

Part 6 implemented a more sophisticated code to carry out a Least-square fit of the data. Using scipy, the values of the parameters b and c were found to be -0.03252979329724948 and 35.36563360300477. From the 2 methods, the average value of the decay constant is $0.029616578315576875 s^{-1}$. The plot for this method showed a more defined exponential decay, indicating it may be more effective. The results of this investigation verify the effectiveness of the Least-square fits method to represent data; data which may not describe a trend as expected due to experimentational error.