Computer simulation: Numerical Methods, assignment 2

Name: Rohan Kadam

Student number: 20334092

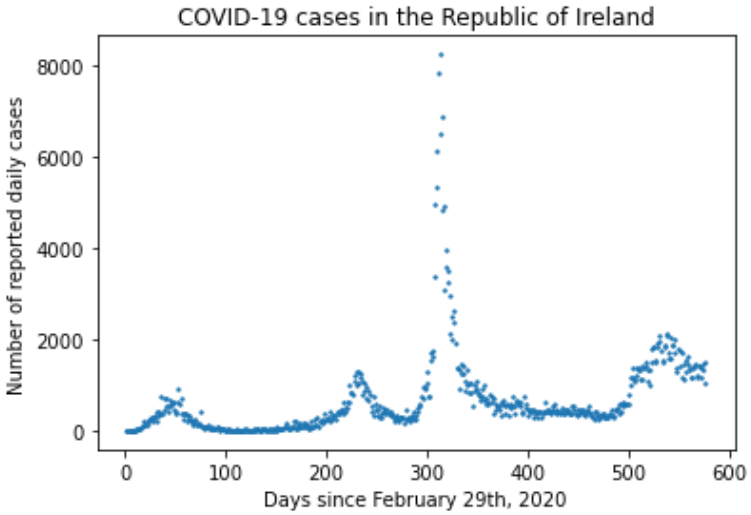
Introduction

In this laboratory we are given a data set which lays the relation between the numbers of daily Covid-19 cases in the Republic of Ireland against the number of days past February the 29th 2020, i.e. we begin counting the days on the 29th February and examine how many Covid-19 cases occurred daily. The code was written on Jupyter notebook but could have been carried out on another platform such as visual studio code. There are tasks that needed to take place before we start manipulating the data and carrying out our assignment. Firstly to import the data we download the file and store it on our device in a known directory. We need to create pairs of arrays for each of our data sets, one for the number of days and one for the number of cases recorded. We skip the first row of data as we do not need to read this. This is done by writing [input file = "C:\Rohan\COVIDData.txt" days, cases = np.loadtxt(input file, skiprows=1, unpack=True)] Just to make sure that there is a 1:1 relationship between the two variables (make sure day 1 corresponds to exactly x number of daily cases) we print the length of each variable for example [print(len(days))]. This is also done for cases. We find out that both variables yield a results 576, which is what we expected. The data set is printed so we can have a convenient way of examining it. Numerous packages were also installed and loaded at the start of the assignment such as but not limiting to, scipy, matplotlib, numpy, os, plotly.express, and pandas. Each of these had their specific role to play and when combined, allowed me to carry out the tasks at hand. Sample pseudocode examples will be given throughout the report to describe how I attempted to carry out the tasks.

Task 1

In the first task we are asked to plot the two sets of data against each other to reproduce the figure given in the question. We create a scatter plot of the number of daily cases against the days passed since February the 29th 2020. This is done simply by using the matplotlib package we introduced at the start and by using the plt function in this manner [plt.scatter(days, cases, s = 2)]. The graph produced is shown below. The exponentially rising part of each wave is described by the equation below, where n(t) = daily number of cases, t_o = 0 days passed since February 29th 2020, and λ is the constant found from a least square fit.

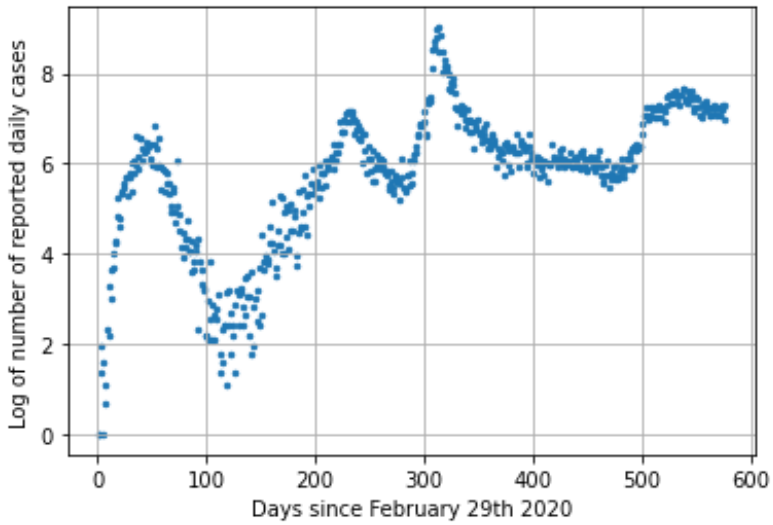
Equation 1: $n(t) = n_0 exp[\lambda(t - t_0)]$



Task2

In this question we are asked to plot a scatter plot of the same variables as in task 1 but this time we plot the log of the number of cases against the days past since February 29th 2020. Why we should we do this we should ask ourselves. Firstly let us examine the graph produced above. It seems as though there are 4 waves of people catching covid-19, with the first one occurring somewhat between days 1 and lowering back down at 100. This data is telling us that there are increasing and decreasing number of covid-19 cases for certain periods of time. There is not a linear relation between daily cases and days passed (for example as the number of days passing by increases the number of covid-19 cases increase) there are also many straight lines observed near but not on the line y=0. This tells us for a certain period of time the number of covid-19 cases recorded is near to a constant. This makes sense as covid-19 is a viral contagious illness that once one person is exposed, he/she is more likely to pass it on to others and the spread occurs exponentially. However are we able to find a relation between each wave and how they may differ? This is where we need a logarithmic scale to further investigate our data. A logarithmic scale is usually used when we are investigating a large range of quantities. It is useful when creating graphs to compress the scale and make the data easier to comprehend. It ensures that when studying data and potentially plotting relationship lines that the outliers have a less meaningful impact than what they would have without the logarithmic axis. We are analysing the covid-19 cases recorded over a range of days so it is understandable why we use this axis, for example on certain days there could potentially be an outlier in terms of the number of cases recorded.

Now to plot the graph but with a log scale for the y-axis, we use the log function in numpy like so: [plt.scatter(days, np.log(cases), s = 6)]. The graph plotted is shown below.



As we can see there are four potential waves, with the first 3 describing an n type shape. As the plotting is manipulated we can now see the waves could potentially be closely described as straight (linear) lines. For example the first linear line would be the first waves left hand side with a positive slope, and similarly the second linear line would be the line going from the peak of the first wave to somewhere at the start of the second wave.

Task3

We now need to carry our linear regression fits. The equation below describes the graph above, with the 'beginning' and 'end' 'part' of the waves being determined by this equation.

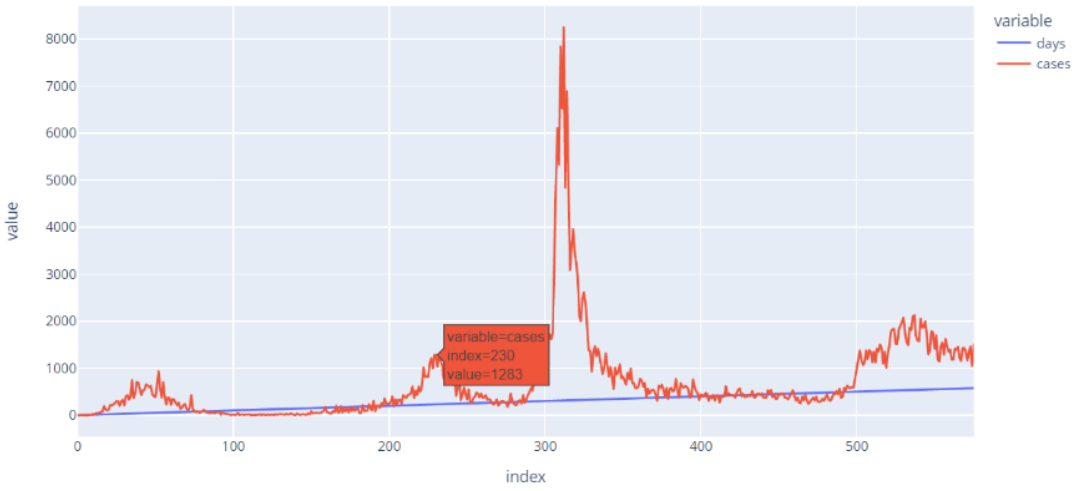
$$\text{Equation 2: } \log[n(t)] = \ln[n_0] + \lambda(t - t_0)$$

If we let [a = $\ln[n_o]$ - λt_o] and [b = λ] then we can rewrite the equation above as

$$\text{Equation 3: } \ln[n(t)] = a + bt$$

This is the functional form of $\ln[n(t)]$ as a straight line in the form y = mx +c. Now we need to find the values for a and b. However how are we supposed to know the range of values to measure from for each wave? One could estimate this range from observation of the graph or create a function between the two estimates and subsequently, find the maximum and minimum points, however this is tedious and not convenient. I took a different approach. I plotted an interactive graph of the number of daily reported cases against the days passed since the 29th of February 2020 using two packages not in built in Jupyter. I downloaded plotly and pandas packages using the anaconda command prompt and created the code below the use the interactive graph. This will allow me to accurately detect the range of values each of my linear fit lines should include, i.e. from the minimum start point of the wave to the maximum and subsequently the maximum of the same wave to the minimum of the next.

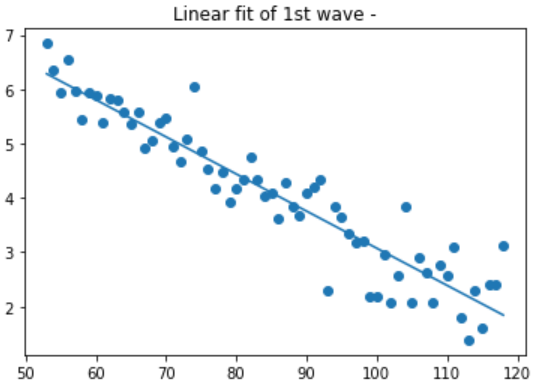
```
In [18]: import plotly.express as px
         import pandas as pd
         norm_data = pd.DataFrame({'days':days, 'cases':cases})
         fig = px.line(norm_data)
         fig.show()
```

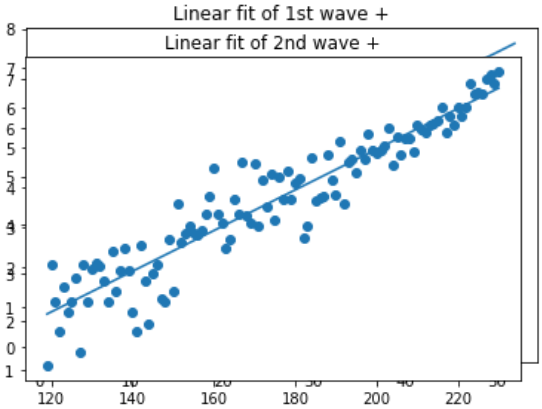


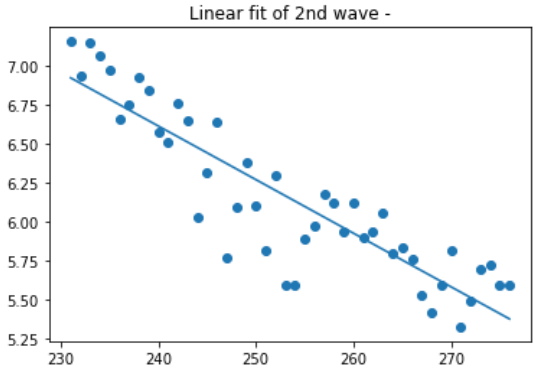
9 linear fits in total were created. To create a linear fit the scipy package was used in particular, the linregress function. A linear regression line function was created, as well as the function for the 'a' and 'b' values from equation 3. This was done in this manner [linearfit1_function = stats.linregress(days[0:52], np.log(cases[0:52]))] [linearfit1 = (linearfit1_function.slope*days[0:52] + linearfit1_function.intercept)].

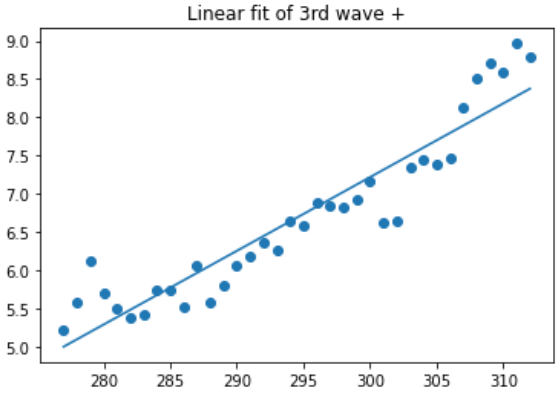
The slope and intercept values were defined as follows
[slope1 = linearfit1_function.slope]
[intercept1 = linearfit1_function.intercept]

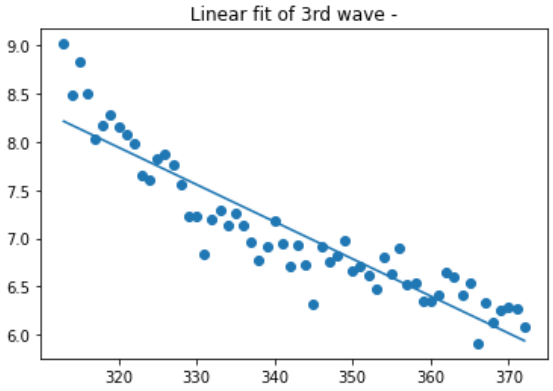
The values for the slope and intercept were printed and the graph is plotted using familiar methods from the matplotlib library. The graphs for all 9 fit lines are shown below (1st wave + signifies a linear fit of the first wave, in particular the line with positive slope going from minimum to maximum)

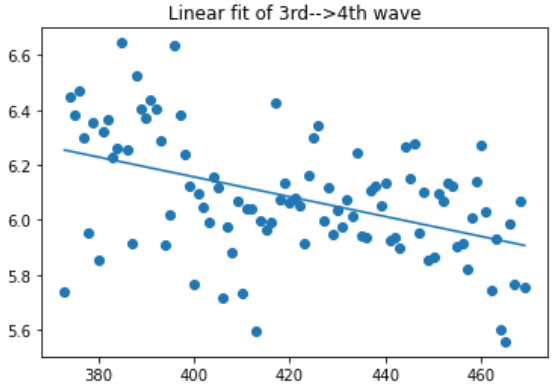


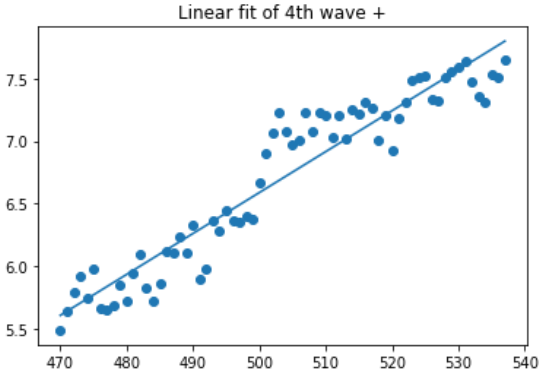

Linear fit of 1st wave +
Linear fit of 2nd wave +

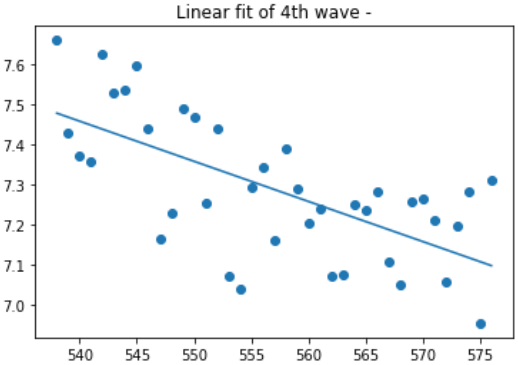

Linear fit of 2nd wave -


Linear fit of 3rd wave +


Linear fit of 3rd wave -


Linear fit of 3rd-->4th wave


Linear fit of 4th wave +


Linear fit of 4th wave -

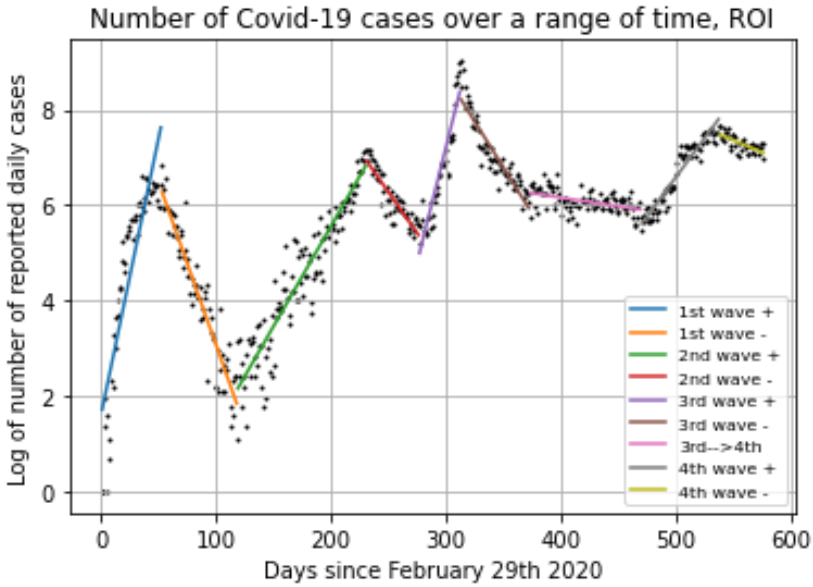
After plotting each of the linear fits we can see there is a strong correlation between the rise of each wave and the decline of each wave with respect to the two variables (days and cases). Plotting the linear fit on the logarithmic scale allowed us to see this more clearly, with less room for skewed data. The graph of the linear fit labelled “Linear fit of 3rd-->4th wave” was plotted due to concerns whether it was reasonable to make a linear fit from the minimum of wave 3 to the maximum point of wave 4, as there had been a plateau in the number of covid-19 cases between approximately 373 and 490 days. Instead a linear fit was made from the days where a plateau occurred. The values ‘a’ and ‘b’ were found from equation 3 above. The values are given to 3 decimal places to keep consistency. The range over which line fit is also provided. After observation of the results the increase and decrease in λ is almost similar for each wave, possibly suggesting a quadratic like figure.

	‘a’ value	‘b’ value (λ)
Linear fit of wave 1+ [0:52]	1.602	0.116
Linear fit of wave 1- [52:118]	9.900	-0.068
Linear fit of wave 2+ [118:230]	-2.833	0.042
Linear fit of wave 2- [230:276]	14.859	-0.034
Linear fit of wave 3+ [276:312]	-21.736	0.096
Linear fit of wave 3- [312:372]	20.291	-0.039
Linear fit of Linear fit of 3rd-->4th wave [372:469]	7.602	-0.004
Linear fit of wave 4+ [496:537]	-9.758	0.0327
Linear fit of wave 4- [537:576]	12.862	-0.010

Task 4

Now we need to combine the linear fit plots and the original scatter plot of we carried out into one plot depicting all the information. To do this we plot the linear fit followed by the scatter plot of “Log of number of reported daily cases” against "Days since February 29th 2020". We do this by coding the plot of the linear fit with defined linear fit function from task 3 by [plt.plot(days[0:52], linearfit1, label='1st wave +')]. This is done for every range. The scatter plot is also done by taking the whole range we found out for both number of days and the log of the number of cases as shown by: [plt.scatter(days[0:576], np.log(cases[0:576]), color ='black', s = 1.1)].

Adjustments were made to the graph for clear viewing. The graph is shown below.

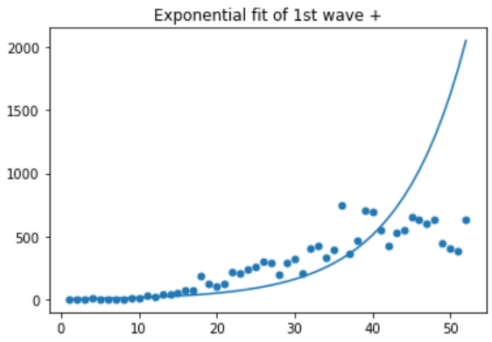


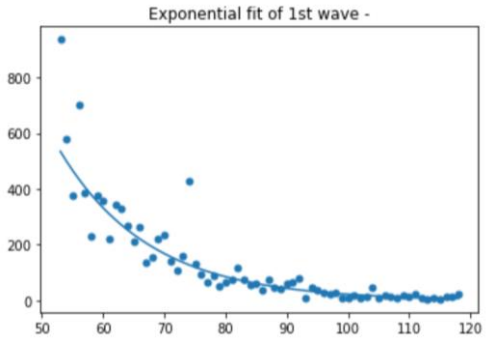
Task 5

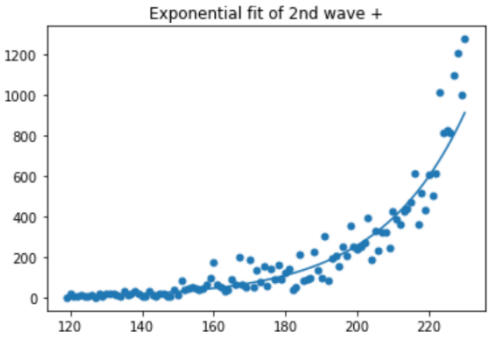
After finding the values for the slope and y-intercept for all of the linear fit lines, we now wish to find the values for n_o as per equation 1. As the intercept and slope of each line fit was already defined this is a quick process. The following is an example of how n_o was calculated for the 1st wave rising (+). [no1 = np.e**(intercept1 + slope1*0)] and print[no]. The following table was made for n_o for each value of 'a' and 'b'.

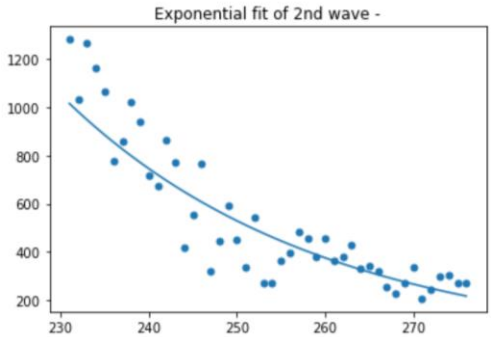
	n_o value
Linear fit of wave 1+ [0:52]	4.961
Linear fit of wave 1- [52:118]	572.193
Linear fit of wave 2+ [118:230]	8.307
Linear fit of wave 2- [230:276]	1051.871
Linear fit of wave 3+ [276:312]	134.301
Linear fit of wave 3- [312:372]	3839.292
Linear fit of Linear fit of 3rd->4th wave [372:469]	522.016
Linear fit of wave 4+ [496:537]	263.724
Linear fit of wave 4- [537:576]	1789.047

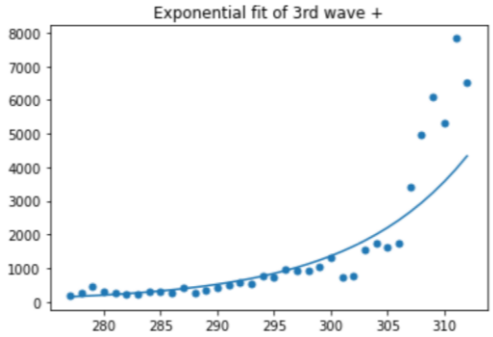
Now an exponential fit is applied to the original graph plotted at the very start of the report, (the number of cases plotted against the number of days without any adjustments made to the scales). We do this by defining the exponential fit equation which is equation 1 from page 1 of the report. This is done when we are calculating n_o. We then define the range over which we want the fit to lie in which is the same range applied as the linear fit plots. This is done by [exponentialfit1 = no1*np.e**((slope1)*(days[0:52] - 0))]. A scatter plot is made of the cases against days from the range desired. An exponential fit is required as no adjustments were made to the axis. The plots are shown below.

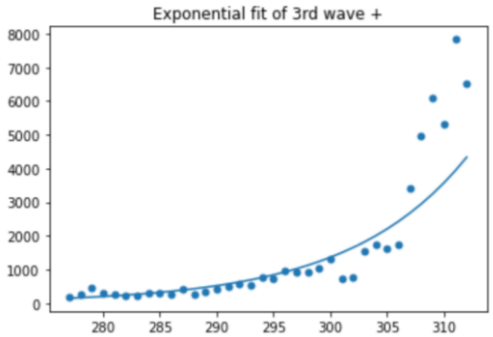


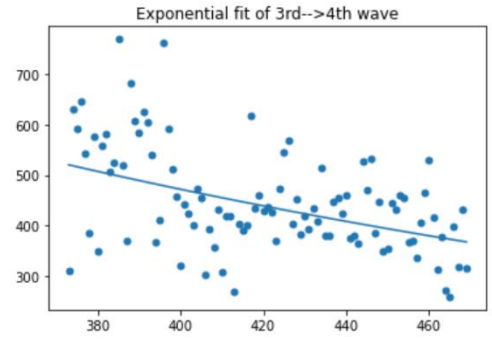


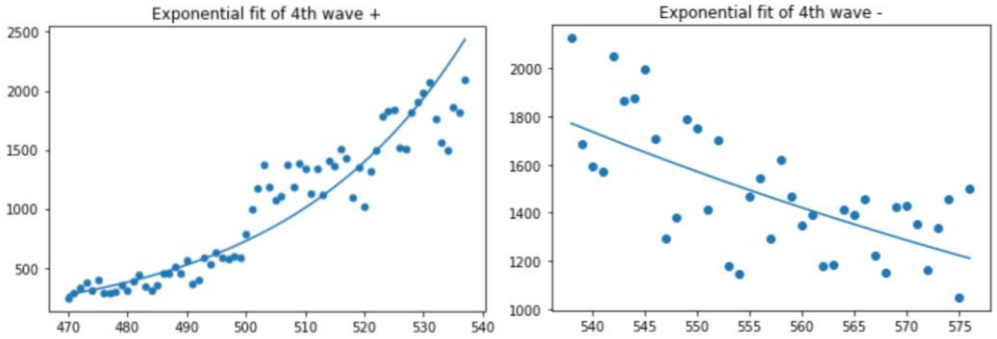






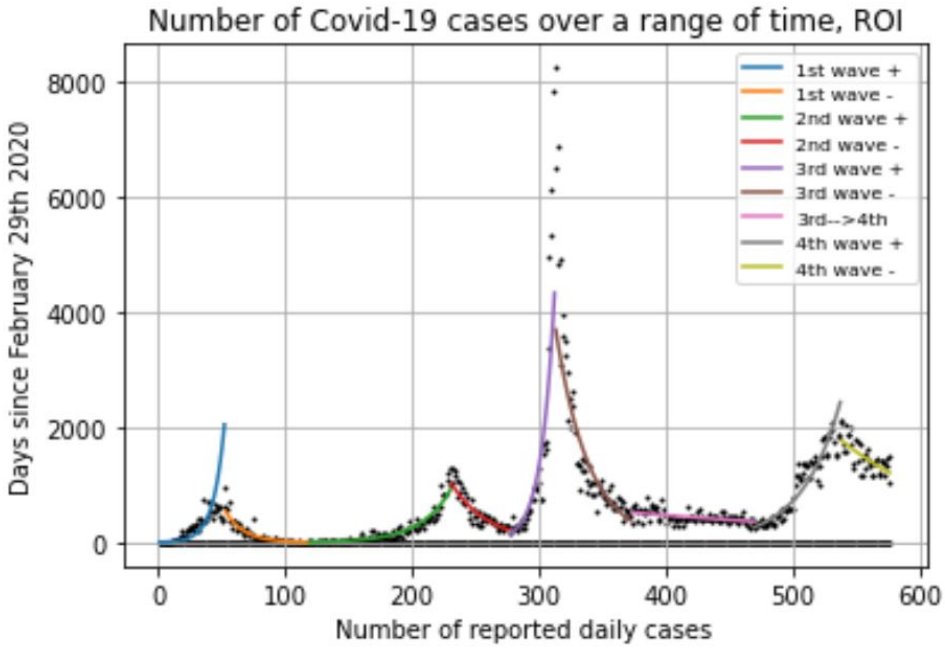






The exponential fit plots were now combined with the original cases vs days plot by [plt.scatter(days, cases, color = 'black', s = 1.1)]
[plt.plot(days[0:52], exponentialfit1, label='1st wave +')].

The graph below is reproduced after making adjustments for a clear view of different plot fits.



Task 6

Wave 4 appears to be different than the rest. In the sense that the first three waves have an ‘n’ like shape almost symbolizing a negative quadratic function. And the fourth wave stays constant for a while after increasing similarly to the first three, but not having the almost same rate of declining/decaying like them. This could be due to a kerb in the rate people are being infected with the disease, possibly through the vaccine. An exponential fit is not the most accurate fit we could possibly apply. A possible $\frac{1}{2}$sin fit would have been more accurate to plot. Perhaps a polynomial with a leading co-efficient of a negative number and with an odd degree. A fit of this polynomial could possibly be perfect.