COVID-19 Data Analysis

Rishav Bhagat

April 26, 2020

Contents

1 Dataset

I used the dataset that was collected and used by John Hopkins University for this project. This dataset was collected from many international health organizations and all compiled into onto github repository. It contains information on daily reports (including new cases, deaths, and recoveries), and global trends.

Here is the link to the datasets I used:

https://github.com/CSSEGISandData/COVID-19

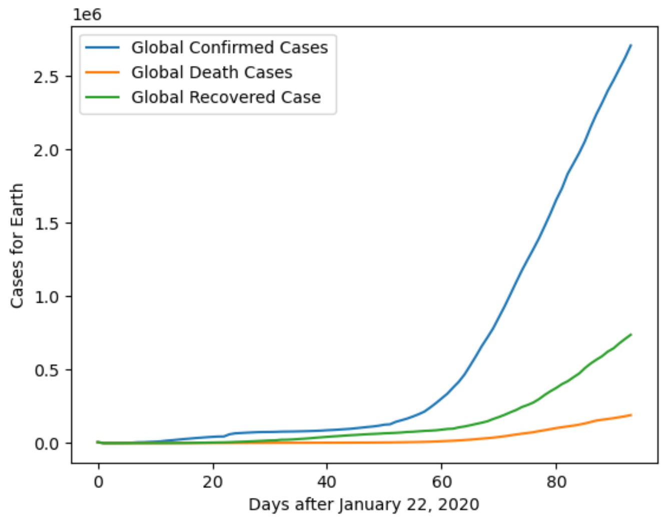
1.1 Downloading the Data

I created a batch script to download all this data using git, delete files that are not needed for my analysis, and move the files into more convienient locations.

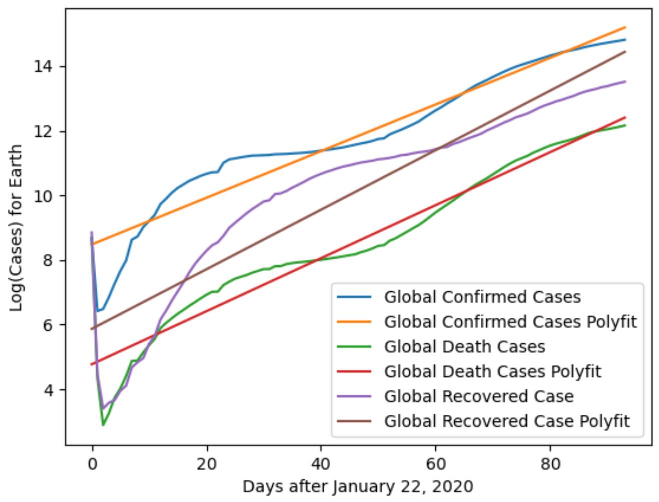
```
@echo off
rmdir data /s /q
git clone https://github.com/CSSEGISandData/COVID-19.git
rename COVID-19 data
cd data
rmdir .git /s /q
rmdir archived_data /s /q
rmdir who_covid_19_situation_reports /s /q
del README.md
mv csse_covid_19_data/* .
rmdir csse_covid_19_data /s /q
cd ..
git add data
git commit -m "updated data from John Hopkins github repo"
git push
```

1.2 Initial Glance

First, I plotted the global coronavirus data (cases vs days) in a similar fashion to the way it was plotting on the John Hopkins project. This is to reveal any obvious trends.



At a first glance, this looks like exponential growth, so I then plotting a logarithmic graph with a line of best-fit.



A linear relationship in the logarithmic graph corresponds to an exponential relationship in the original graph. As we can see, the logarithmic graph is approximately linear, which confirms that so far, the coronavirus is exponentially growing. But we also see that the curve is beginning to flatten out, corresponding to the true logistic growth of the original graph. The beginning of all logarithmic graphs appear to be exponential, which explains why it apears as it does.

Now, we know that logistic growth will eventually level out, and it is in our interest to see when and where it will do this globally. We can do this by tracking the growth rate of the graph, as we shall see in the next section.

2 Data Analysis and Implications

First let's define some variables:

- Let us refer to the region we are talking about as X. For now X refers to the entire globe.
- Let N^X be the final number of people who have COVID-19 within X. I may sometimes just use N, which will refer to the region in question at the time.
- Let N_i^X be the current number of people who have COVID-19 within X on the i-th day after January 22nd (which is when John Hopkins began collecting data).
- Let $\Delta N_i^X = N_i^X - N_{i-1}^X$ be the number of new cases on a given day, defined by i.
- In a similar way, define $D^X, D_i^X, \Delta D_i^X$ as the people who died due to COVID-19 in the respective time frames.
- In a similar way, define $R^X, R_i^X, \Delta R_i^X$ as the people who recovered from COVID-19 in the respective time frames.

While doing the analysis on the data, I will focus on the number of cases, but the number of deaths and recoveries also follow very similar patterns.

2.1 Growth Ratio

Another intrinsic value that can give us insight into the data is the growth ratio. Let's define this as

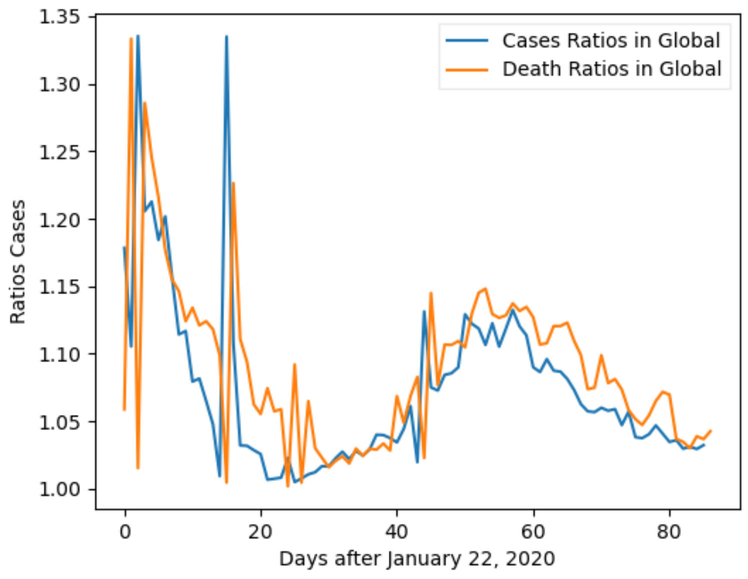
$$GR_i^X = \frac{N_i^X}{N_{i-1}^X} \tag{1}$$

For pure exponential growth the ratio should be a constant number throughout the entire domain, but since the ratio is decaying with time (as we see in the graph below), there is more evidence pointing to a more logistic approach. We see that the growth ratio is approaching one for both the cases and deaths, showing that eventually $N_i^X \to N^X$.

Another interesting observation that is made apparant from this graph is the relationship between the deaths and cases. In other graphs this relationship is also evident, but here they are both plotted on the same scale so it is even more clear. The deaths are following a similar pattern as the cases, but they are slightly time-shifted (forward in time). This makes sense since we would expect a ratio of cases to become death in a certain time.

We can also estimate the effectiveness of our treatment by checking if the death ratios start to become less that the cases ratios, but it is clear that this is not the cases currently, implying that we have not yet found a proper treatment.

A further and more rigorous analysis of this graph and more detailed data could also give us insight to how long it takes the coronavirus to cause death on average.

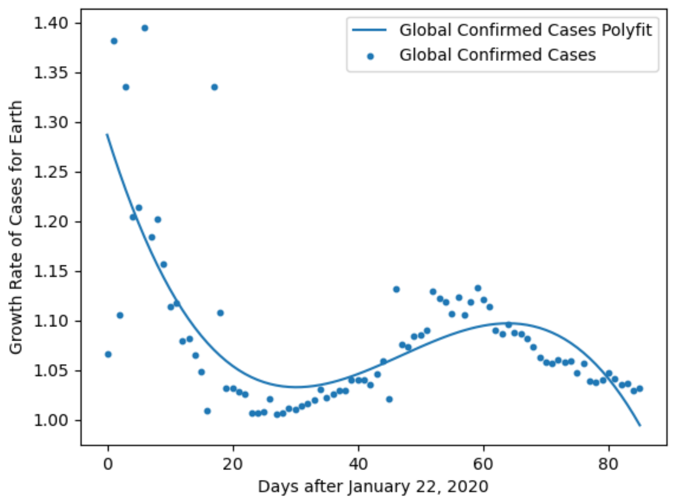


2.2 Growth Rate

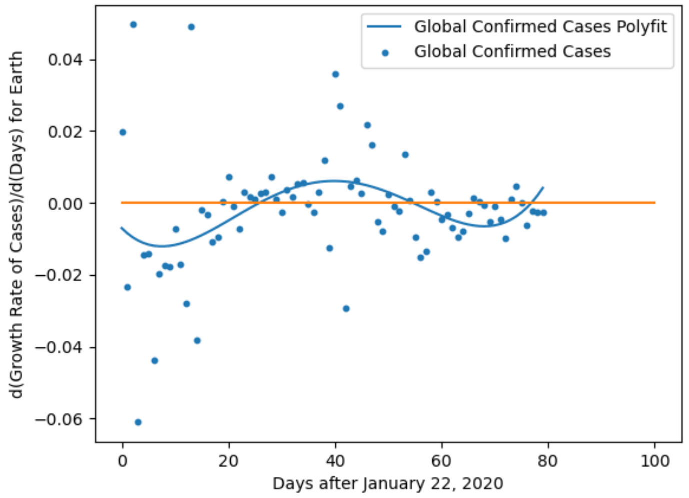
Now we can define our growth rate as

$$G_i^X = \frac{\Delta N_i^X}{\Delta N_{i-1}^X} \tag{2}$$

We are interested when this value crosses from $G > 1$ to $G < 1$, which corresponds to a point of infection. It is when the numbers of new cases each day begins to decrease rather than increase. Or in terms of calculus, when the second derivative becomes negative. The reason we do not just use the second derivative is that this data is noisy and approximating a high-order derivative would not work too well. Now plotting the growth rate:



If the graph continues to follow the polyfit curve drawn, then we expect the inflection point to occur soon, but we do not know if it will go back up as it did at around $i = 25$. In fact, we can expect it to go up, which is evident by plotting the approximated derivative [1] of the growth function.



[1]When I refer to the derivative I am using the central difference approximation give by:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

with $h = 1$ since we have a discrete input space (each day).

The derivative of the growth rate at the current time ($i \approx 80$) is following the same pattern as at $i = 25$ and it has gone above the zero-line, implying that the growth rate will increase again.

This is a dissapointing result since finding the true point of inflection will let us estimate where the number of cases will max out. If the point of inflection occurs at i, then it is reasonable to guess that the number of cases will max at $2 * N_i^X$ by the way logistic growth works.

2.2.1 Modelling with the Growth Rate

The curve generated by `np.polyfit` fits the growth rate pretty well, so it may be possible to create a model based on the growth rate. To do so, we just have to follow the definitions of the growth rate and the solve for N_i:

$$G_i^X = \frac{\Delta N_i^X}{\Delta N_{i-1}^X} = \frac{N_i^X - N_{i-1}^X}{N_{i-1}^X - N_{i-2}^X}$$
$$G_i^X(N_{i-1}^X - N_{i-2}^X) = N_i^X - N_{i-1}^X$$

and finally

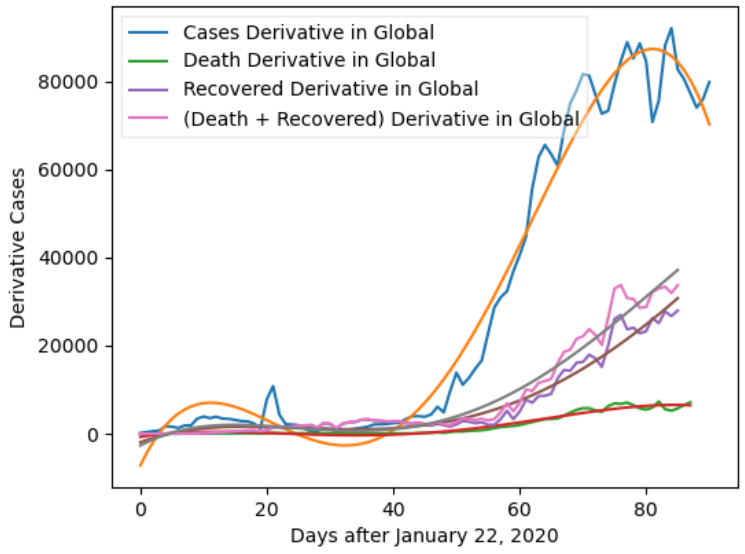
$$N_i^X = N_{i-1}^X + G_i^X(N_{i-1}^X - N_{i-2}^X) \tag{3}$$

But there are a lot of problems with a model like this. For one it is based on previous estimates, so any error in early estimates will propagate through to the later estimates, making the model extremely inaccurate for anything too far out of the domain of the data.

Also, as we showed before, the growth rate will likely not follow the polyfit curve outside the domain of the data.

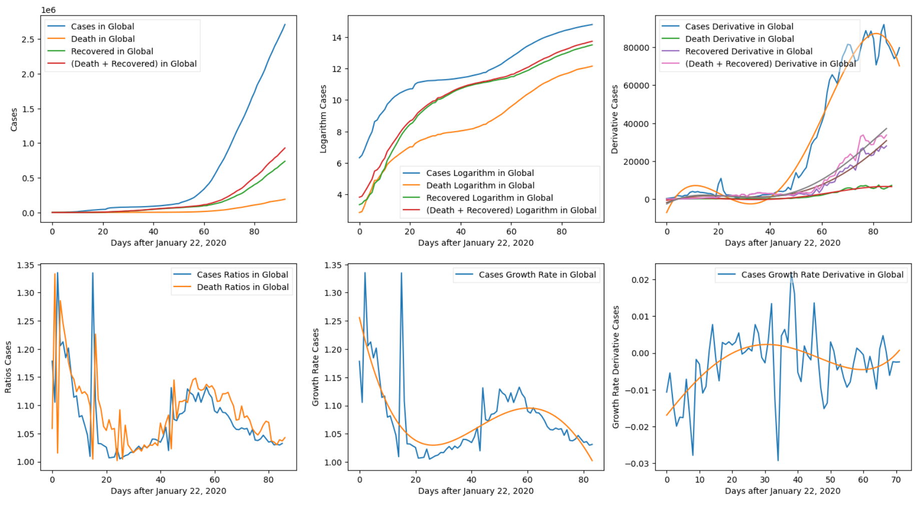
2.3 Derivative

The derivative (approximated) of N_i^X with respect to i, or $\frac{dN_i^X}{di}$, is an important value to look at since it can also help us classify the type of model the spread of coronavirus is following. In an exponential model, we would see $\frac{dN_i^X}{di} \propto N_i^X$, while in a logistic model we would expect $\frac{dN_i^X}{di} \propto (1 - N_i^X/N^X)N_i^X$. Now plotting the derivative:



As expected the derivative is never negative since that would meann the number of cases are going down, which is impossible (since the number of cases is defined as the number of all cases recorded and does not decrement from recoveries).

Putting this graph next to one of the number of cases, we can compare them to see proportionality and see what type of model the spread is following. I will do this with the Analysis Dashboard for Global Cases, which I talk about more in Section 2.5



The derivative is initially approximately proportion to the number of cases, which is seen by looking at the first and third graphs. But then the derivative starts to level off as expected by logistic growth. The derivative is starting to decrease implying a critical point. Now if we assume that the graph will continue to fall in this manner we can expect the derivative to go back to 0 at around $i = 120$, or about 4 months after January 22, which puts us at around the end of May. Once the derivative is back at zero there will be no new cases and the virus will have stopped spreading. But a crude prediction such as that one does not account for skew, since chances are that the graph will be right skewed, since that is how logistic graphs are. Accounting for this we can expect the curve to flatten out a bit later than the end of May.

Another option is that we can try to use an initial subset of the data to try to estimate the proportionality constant c by

$$\frac{dN_i^X}{di} = c(1 - N_i^X/N^X)N_i^X \tag{4}$$

where $(1 - N_i^X/N^X) \approx 1$ (in the initial subset), so $\frac{dN_i^X}{di} \approx cN_i^X$. After finding c, we can use the graph for the derivative to estimate N^X, assuming a perfectly logistic model. I will actually implement this in Section 2.6.

2.4 Fitting Polynomials to Subsets of the Data

The numpy module has a `polyfit` method that can take a bunch of data and fit a polynomial of a given degree to it. In theory, this can be used to create a model for the COVID-19 data, but there are many problems with this[2]. But we can still use a technique like this to gain insights into our data.

2.4.1 Finding the Best Degree for Polyfit

Just like any other model, we can measure how well the generated polynomial fits the data using some sort of loss function. I used mean squared error, defined by

$$MSE(\vec{p}, (N_i^X)) = \frac{1}{N}\sum_{i=0}^{N-1}(N_i^X - \vec{p}\cdot\vec{x}(i))^2 \tag{5}$$

for our polynomial model, where $\vec{p}$ is a vector containing the weights for our polynomial model, N is the number of samples. The function $\vec{x}(i)$ returns a vector defined by

$$x(i) = [i^m, i^{m-1}, i^{m-2}, \ldots, i^2, i, 1] \tag{6}$$

where m is the polynomial's degree. The vectors $\vec{p}$ and $\vec{x}(i)$ have length $m + 1$. Here x acts as a feature transformer.

Now to find the smallest m that has a good fit to our data, I track the ratio between the impovement since $m - 1$ and the original MSE with $m = 1$ and increment m until it is less than a theshold ϵ. Essentially I find the smallest m, where

$$\frac{MSE(\vec{p_m}, (N_i^X)) - MSE(\vec{p_{m-1}}, (N_i^X))}{MSE(\vec{p_1}, (N_i^X))} < \epsilon \tag{7}$$

[2] I go further in depth into these problems in Section 3.1

where the MSE for $m = 1$ is used as a scaling factor. After this we know that after $m - 1$ there is not going to be much improvement, so we do our polyfits on $m - 1$.

2.4.2 Correlations Between the Best Degree and Subsets of the Data

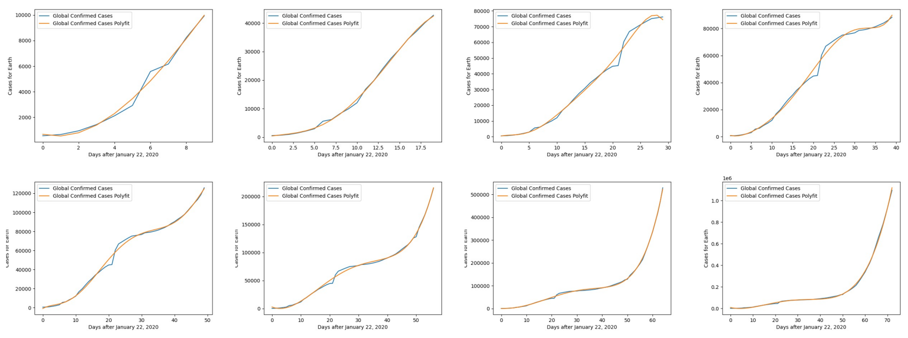
From now on, I will refer to m as the best degree to use `np.polyfit` with. I will use python subindexing, which is defined like this: If

$$x = [1, 2, 3, 4, 5]$$

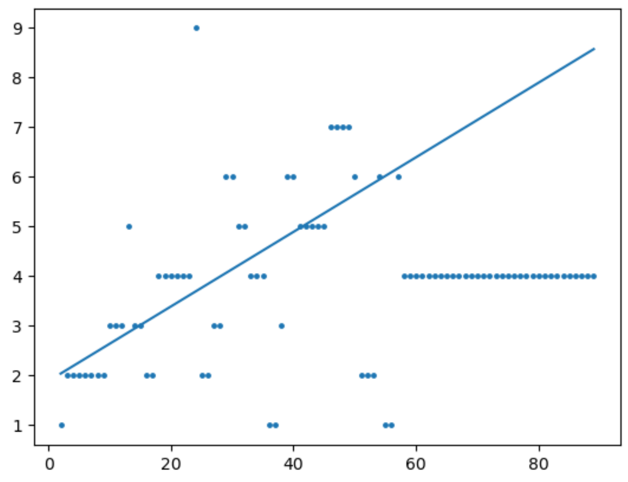
then

$$x[0:2] = [1, 2]$$

And it is [inclusive, exclusive]. Now define, $m(j) = \text{findBestDegree}((N_i^X)[0:j])$. Incrementing up j and calculating $m(j)$, we are essentially finding a best fit polynomial of optimal degree for more and more of the data. Here is some of the polyfits for some subsets (incrementing by 10):



Now, there are two main factors in play as we increase j: there is more data to model, and the rate of change keeps increasing more as we get further in (N_i^X), requiring a higher polynomial degree to model. Both of these require a higher degree to model as j increases.



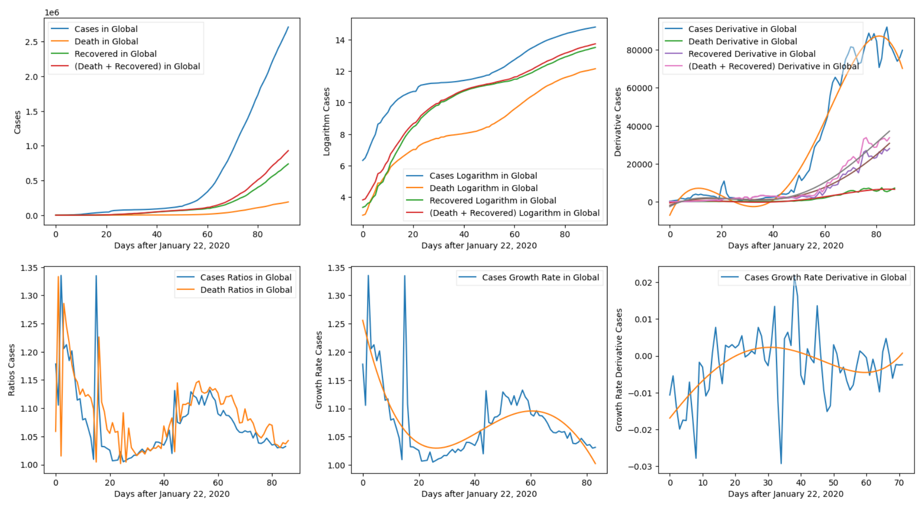
In this plot the linear regression line is drawn based on the only the $j < 50$. And the graph exihibits the predicted pattern while $j < 50$, increasing m with j, but after around $j = 55$, the m stays constant at 4. Cross-referencing this with our previous finding that there is a point of inflection around that point (since $dG_i^X/di = 0$), I am guessing that the rate of change of the function slowed down enough for a polynomial of order $m = 4$ to model the graph.

Another interesting thing to see is that the points on the graph are clustered, which makes sense since adding one point will likely not change m by at least one everytime, and the output space is discrete, causing clusters to from.

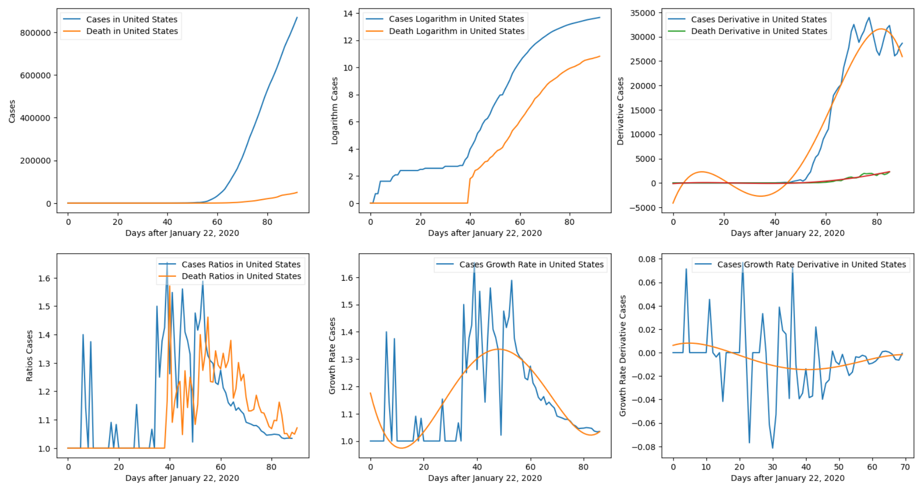
2.5 Country Analysis

To do an analysis of the countries, I will be using a ”dashboard” with a bunch of graphs that each convey different information. Many of the graphs are talked about in previous sections talking about the global data. But now we will focus in on other regions (different values for X).

Many of the observations I make will be in relation to the global dashboard since I have already done an in depth analysis of the global data. One of the difference that apply to all countries is that the graphs are scaled down from the global scale, but since we are analyzing trends, the amplitude does not matter too much. Another observation that applies to all the countries is that the data is noisier than on the global scale since there is less data.



2.5.1 United States

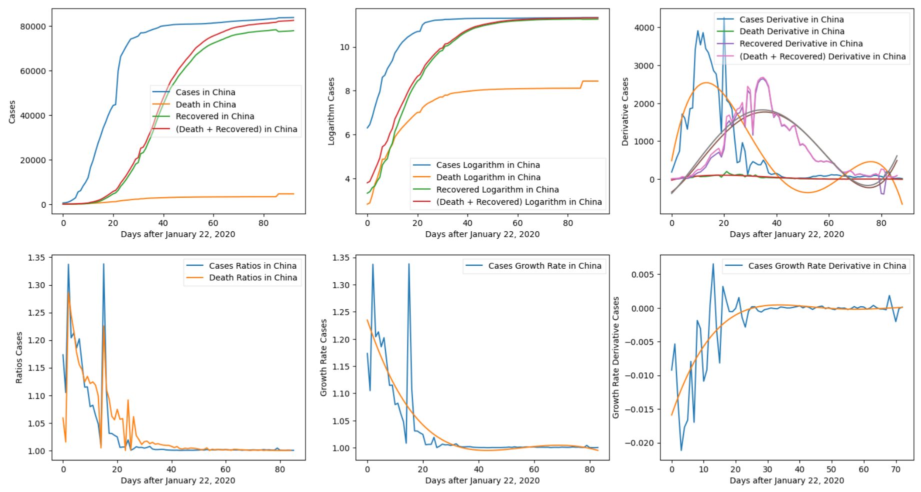


The trends that the United States graphs are following are similar to those of the global scale, except they are time shifted forward. This is a result of COVID-19 spreading to the United States later than it the average country. Up until around $i = 40$, the data is either near 0 or does not follow any clear pattern. This is because the number of cases in the United States before that point was near 0.

Also, the logarithmic graph has only recently started to flatten out, meaning that the United States has been experiencing exponential growth of the coronavirus until recently. The is likely due to the late start in taking action against the virus. Another point is that the curves of the United States are flattening at around the same time as the global curves. So, since COVID-19 hit the United

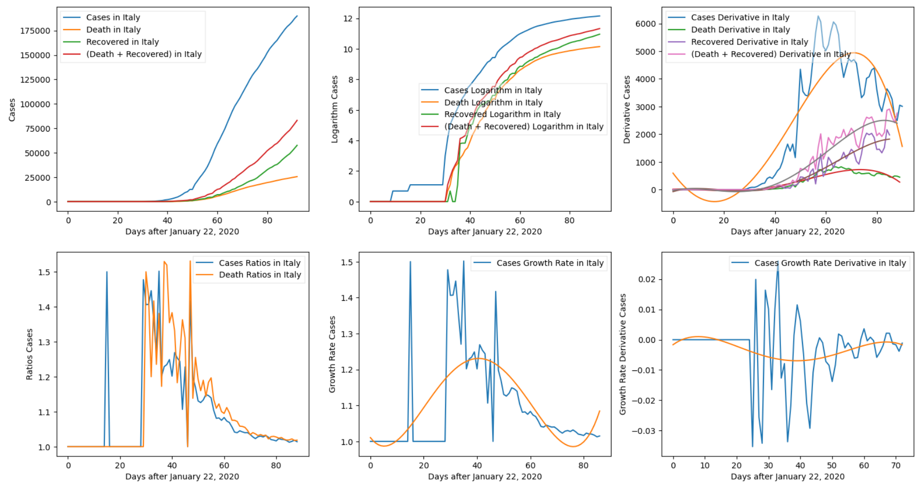
States pretty late, this shows that the United States is doing better than the globe as a whole.

2.5.2 China



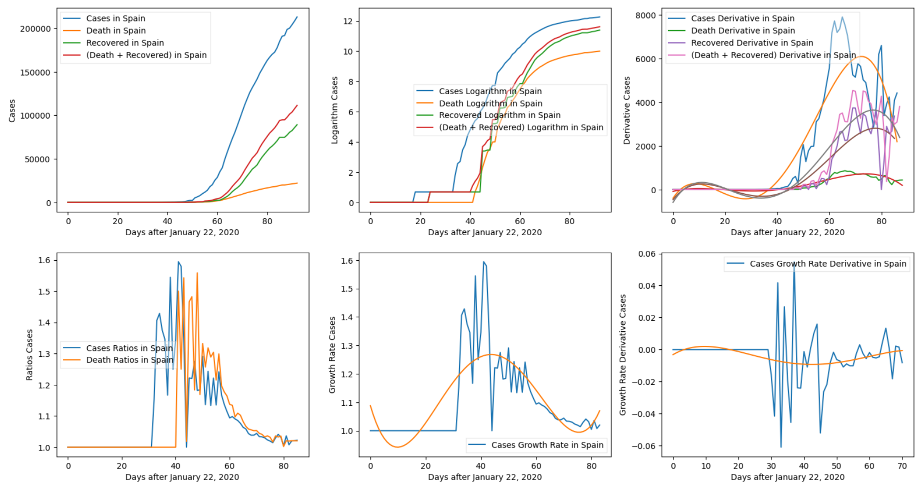
China's graphs are shifting backwards in time in relation to the global graphs, a result of the fact the virus started in China. Looking at the first graph we see that the difference between current cases and (death + recovered), or $N_i^{China} - (D_i^{China} + R_i^{China})$, is approaching 0. That difference is the number of non-dealt with cases. China is the ideal model for logistic growth since we have data on the tail end of the growth. The growth ratio GR_i^{China} and growth rate G_i^{China} have both converged to 1, as seen by the graphs. Also, the graph of the cases follows a logistic growth curve. This flattening of the curve shows that China's strict policies are working in stopping the spread of the virus.

2.5.3 Italy



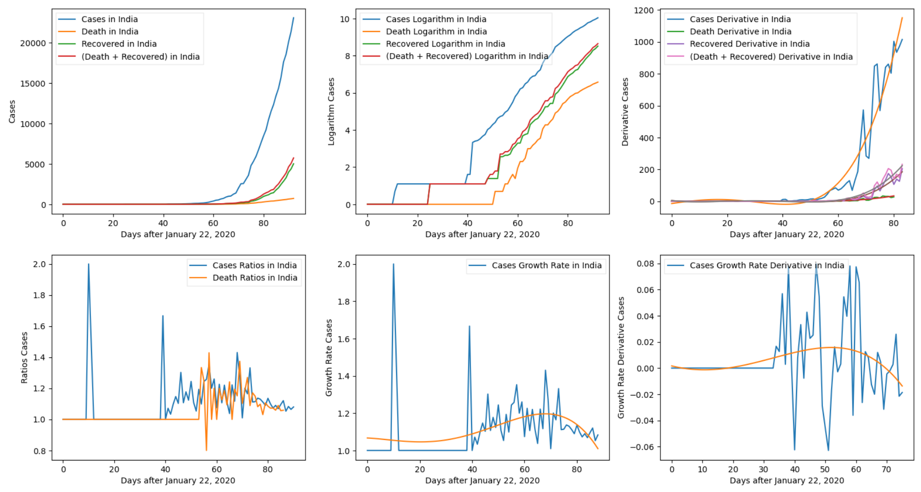
Italy's graphs are time-shifted forward by about 10 less than the US graphs. Taking a look at the logarithmic graph, it is almost imediately flattening. This is unlike the previous couple countries, in which there was a longer time where the logarithmic graph was near linear. This correlated to Italy being well prepared for the virus and taking proper action against it.

2.5.4 Spain



The time shift of Spain is similar to that of the United States, but the behavior of the graphs are similar to Italy. This shows that the difference between United States and Italy or Spain was not due to the ten day difference, but in the action against coronavirus.

2.5.5 India

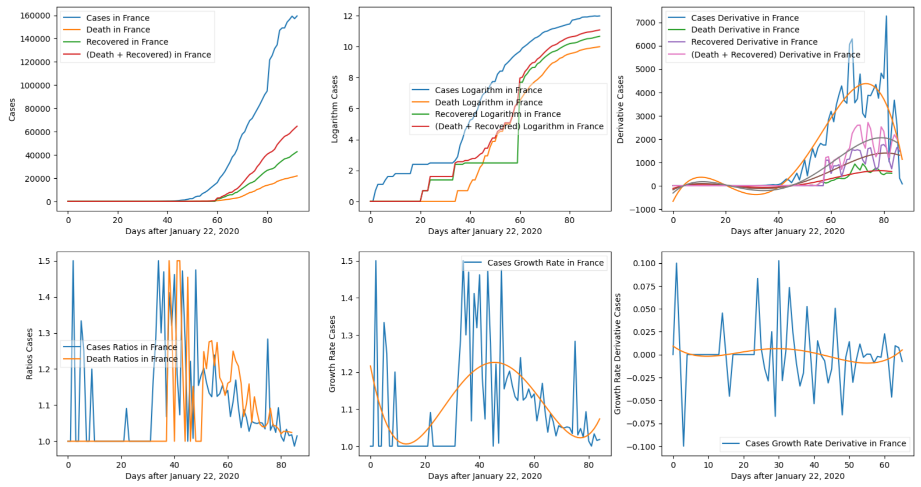


While India's graph of the cases may look similar to the graphs of the other countries, they are very different. This shows the importance of looking at these different transforms of the data (log, derivative, growth rate, ratios, etc.).

India's data is a prime example for exponential growth, which can be seen by taking a look at their logarithmic graph. This graph is almost perfectly linear, with only a slight curve at the very end. India's time shift is similar to that of the United States and Spain, so that it not the reason for them still being in the exponential portion of the logarithmic curve. India's action against the virus is not working to flatten the curve. Looking at the graph of the derivative, we see that it is proportional to the graph of the cases, as expected for exponential growth.

The growth ratio of India is staying approximately constant at around 1.15. In a similar manner, the growth rate is not going down. The polyfit curve for dG_i^{India}/di is the opposite of that of the other countries. While this curve does not model dG_i^{India}/di that well, the fact that it is mostly postive indicated that the growth rate is not decreasing enough.

2.5.6 France



The data for France is very similar to that of the United States. France is time-shifted forward by about 10 less that the United States, making their graphs seem a bit better, but in this case it is very similar to the patterns of the United States.

2.5.7 Comparing Amplitudes

Most of the analysis I have done up to this point has been about looking at trends in the data but now I will take a look at the amplitudes by estimating the number of cases around $i = 80$ by the graphs for each country.

Amplitudes

Region	Amplitude
Global	2.5e6
United States	800,000
Spain	200,000
Italy	175,000
France	160,000
China	80,000
India	20,000

where I have sorted it from greatest to least. By amplitudes it seems that India is doing very well, but from our earlier analysis we know that this may not be true. It is important to take into account both of these factors before making a conclusion.

Another thing to consider while looking at amplitudes is the validity of them. Some countries may not report all of their cases, or they may not even know about them. For this reason, more developed countries and countries with a greater involvement with international health organizations will have larger numbers due to better recording of data. However, if we assume that the ratio of cases reported to true cases stays approximately constant as time passes, we can still look at the trends since we are ignoring amplitudes anyways.

2.6 Auto Analysis

This section will be apply using python to analyze the graphs we have created and use the data to find some intrinsic values and make predictions. These methods will work for any valid value of X, but I am going to mainly use the global data.

2.6.1 Finding the Parameters for a Differential Equation

The first thing I will do is assume the form of the growth to be logistic. This mean it will follow the differential equation

$$\frac{dN_i^X}{di} = c\big(1 - \frac{N_i^X}{N^X}\big)N_i^X \tag{8}$$

Just as a reminder, N^X is the final number of cases in region X, or the "carrying capacity" of the spread. Also,

$$(N_i^X) \to N^X$$

The first step is to split the data into two sections: one where $N_i^X << N^X$ and the rest. Or mathematically,

$$(N_i^X)_1 = \{N_i^X \in (N_i^X) : N_i^X << N^X\} \qquad (N_i^X)_2 = (N_i^X)\backslash(N_i^X)_1 \tag{9}$$

The reason for this splitting is that in (N_i^X), we can ignore the factor $(1 - N_i^X/N^X)$ since $N_i^X << N^X$ implies $N_i^X/N^X << 1$ so $(1 - N_i^X/N^X) \approx 1$. This makes our derivative

$$\frac{dN_i^X}{di} \approx cN_i^X \qquad N_i^X \in (N_i^X)_1$$

approximately follows the differential equaiton for exponential growth in $(N_i^X)_1$. Now, there is only one parameter left in this model for the derivative. In theory, I could use gradient descent to find it, but a much simplier way is to just solve for it $\forall N_i^X \in (N_i^X)_1$ and average the results:

$$c \approx \frac{dN_i^X/di}{N_i^X}$$

Now, taking the average, I set c to

$$c = \frac{1}{m} \sum_{N_i^X \in (N_i^X)_1} \frac{dN_i^X/di}{N_i^X} \tag{10}$$

where m is the length of $(N_i^X)_1$.

Now moving to $(\frac{dN_i^X/di}{N_i^X})_2$, where we have a logistic growth model again, but now we have c and only need to solve for N^X and then apply the same technique as we did on c.

Though some simple algebra

$$N^X = \left(1 - \frac{1}{cN_i^X}\right)^{-1} N_i^X$$

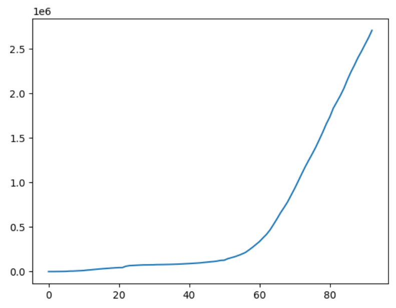
and taking the mean, I can set

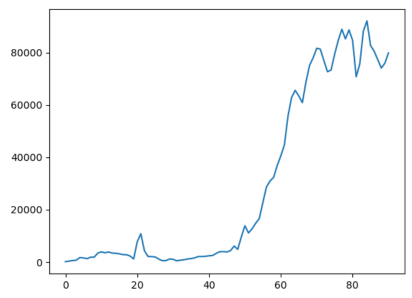
$$N^X = \frac{1}{m} \sum_{N_i^X \in (N_i^X)_2} \left(1 - \frac{1}{cN_i^X}\right)^{-1} N_i^X \tag{11}$$

Now both the scaling factor c and the limit/max value $N^X = \lim_{i\to\infty} N_i^X$ are both known

Implementation Details

To implement this, you must first look at the graphs and choose a place to split your data into the two subsets.





Eying out the two graphs (N_i^X and dN_i^X/di), I choose to split at $i = 60$. I did this by defining a variable `split_point = 60` and then splitting.

A model like this requires a more complete dataset since it is using a pretty basic algorithm to estimate the values for c and N^X. Hence, it works pretty will on the China dataset since China's dataset consists of almost the entire logistic curve. However as more data is collected for other countries and as time passes, models like this can be created for other countries as well.

China's output:

```
Finding Differential Equation Parameters - China
Enter the split point: 10
c: 0.4591556247467866
N^X: 83771.24238239802
```

2.6.2 Evaluating R-Squared on the Logarithmic Graph

Here I will use python to measure how linear the logarithmic of a few countries. The more linear this graph is, the more exponential the number of cases is, implying the action taken against COVID-19 has not been effective.

We can measure how well a linear model fits the logarithmic graph by measure the statistic R-squared, defined by

$$R^2 = (\frac{||\vec{y} - \hat{y}||}{||\vec{y} - \bar{y}||})^2 \tag{12}$$

where in this case y is the $\ln N_i^X$ and $\hat{y}$ is our linear regression model's prediction from the same input as $\vec{y}$. Also, vector - scalar is defined by elementwise scalar substraction.

As R^2 approaches 1, the fit of the linear model if worse, which is a good thing since that means the curve is flattening out. This is a way to numerically give each country a "rating" on how well they are flattening out the curve.

One important implementation detail is the I only created the linear regression model on the data collected after some growth began. This is to get the best representation of the data's growth rather than including extra data points that have no significance. I did this by only including points with $N_i^X > 10$.

Results

Country	R^2 Value
China	0.5470569248885822
Italy	0.17500288400296796
Spain	0.14825334735656834
Global	0.08059000856842549
France	0.06241867519925645
United States	0.05848466049484314
India	0.009644241280158 47

3 Modelling

3.1 Problems with Polynomial (Polyfit) Models

I first considered the idea of a polynomial model back in Section 2.4, but as I added more and more data, the number of degrees required kept increasing, which means that a polynomial model will to generalize well to anything outside of the domain it was trained on.

So using `np.polyfit` to create an model that will generalize well outside the dataset without just using a large number of degrees[3] is not possible.

[3] Using too many degrees is also a problem since the training time will become longer for the same result. We will be losing out on both time and space, the two ways to measure efficiency.

3.2 Logistic Growth Model

3.2.1 Solving the Differential Equation

3.2.2 Using Gradient Descent

TODO: fix the negative sign and redo entire thing

3.2.3 Transforming Features and Linear Regression

TODO: Code this (it should be quick)

3.3 Transfer Models

3.4 Markov Chain Model

3.4.1 Formulating the State Vector and Transition Matrix

Another model that can be made to represent the spread of COVID-19 is a markov chain model with a state vector:

$$\vec{S}_i^X = \begin{bmatrix} n \\ d \\ e \\ p \end{bmatrix} = \begin{bmatrix} N_i^X - D_i^X - R_i^X \\ D_i^X \\ R_i^X \\ P^X - N_i^X \end{bmatrix} \tag{13}$$

This vector contains the current number of cases (this time not including the ones resulting in death or recovery), the number of deaths, the number of recovered cases, and the rest of the population. Also, let P^X be the total population of the region X.

To complete a Markov Chain Model we need to create a transition matrix T, so that when it is multiplied by the state vector at i, we get the state vector at $i+1$. Or mathematically,

$$\vec{S}_{i+1}^X = T\vec{S}_i^X \tag{14}$$

Now to fill in the entries of T, we apply our knowledge of logistic growth and common sense. The change in number of deaths and number of recoveries are only based on the number of current cases. And for our purposes assume that someone who has recovered or died can not get corona again. This gives us the second and third rows of the matrix. We also know that anyone in p cannot go directly to d or e (assuming no vaccines and no other causes of death). So the number of new cases is equal to the number that the untouched population decreases.

Combining everything we get

$$T = \begin{bmatrix} r & 0 & 0 & s(i) \\ r_d & 1 & 0 & 0 \\ r_r & 0 & 1 & 0 \\ -(r-1) & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

where

- r is the rate of increase (of cases)
- r_d is the death rate
- r_r is the recovery rate
- $s(i)$ is some unknown function holding information on the coronavirus growth's dependence on the infectable population

We know s cannot be constant by the difference equation

$$\Delta N^X = c(1 - \frac{N_i^X}{N^X})N_i^X$$

where N^X is some proportion of the population P^X. Clearly there is not a linear relationship between these quantitiies, but there is a relationship nonetheless.

A model like this would work better on a smaller region X since there is a more constant contact throughout everyone and COVID-19 closer to being uniformly distributed across a smaller region. Our model does not account for a non-uniform spread of COVID-19.

3.4.2 Finding these parameters

Finding these parameters for a certain transition $\vec{S}_i \to \vec{S}_{i+1}$ is given by expanding the matrix multiplication and the reformating the equation into a form that linear algebra can solve.

For this specific transition, call it t_i, we can treat $s(i)$ as constant since this value of $s(i) = s_i$ does not need to hold true for any other transitions. We start with

$$\begin{bmatrix} n' \\ d' \\ e' \\ p' \end{bmatrix} = \begin{bmatrix} r & 0 & 0 & s_i \\ r_d & 1 & 0 & 0 \\ r_r & 0 & 1 & 0 \\ -(r-1) & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n \\ d \\ e \\ p \end{bmatrix}$$

where we are given all the values in the two states but need to find the parameters of the matrix r, r_d, r_r, s_i. To do this we want to restructure the equation so we have:

$$A \begin{bmatrix} r \\ r_d \\ r_r \\ s_i \end{bmatrix} = \vec{b}$$

which we can solving using `np.solve`. After expanding the matrix multiplication into 4 different linear equations and rearranging and repackaging back into matricies, we get

$$\begin{bmatrix} n & 0 & 0 & p \\ 0 & n & 0 & 0 \\ 0 & 0 & n & 0 \\ n & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} r \\ r_d \\ r_r \\ s_i \end{bmatrix} = \begin{bmatrix} n' \\ d' - d \\ e' - e \\ n + p - p' \end{bmatrix} \tag{16}$$

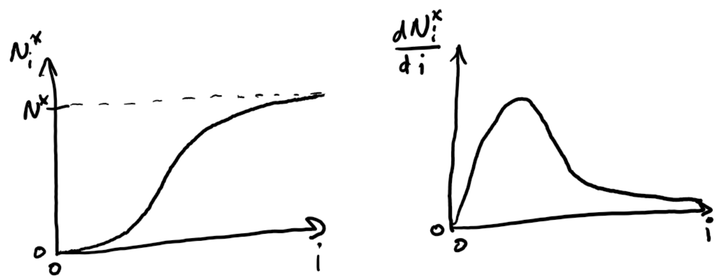
which is of the form we would like. Now, we can apply a similar method as what we did before with the logistic model analysis, by calculating the values for r, r_d, r_r, s_i for every transition t_i. Then for the constnats r, r_d, r_r we average all the values and for s_i we use `np.polyfit` to fit a curve to $s(i)$.

Since we would like to apply this model on a smaller region, I will choose do it by counties in the United States, but the John Hopkins dataset does not include any recovery cases for the United States. So this model will not be good at predicting the recovery rate, since I will just take both e' and e to be $0 \forall t_i$

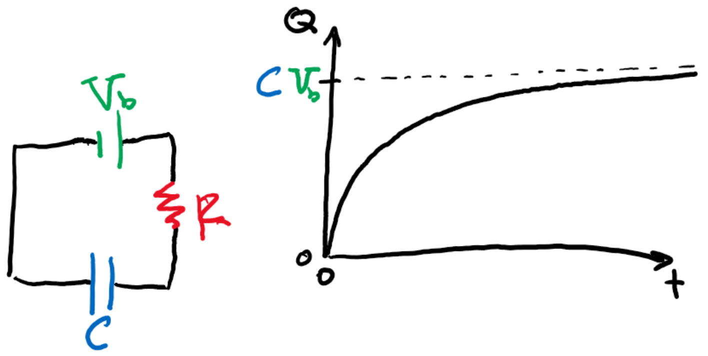
3.5 Circuit Model

3.5.1 RC Circuit

A theoretical model of the COVID-19 data can be made using a circuit. The first thing to do to create a circuit like this is to look at our goal. We want a circuit so that we can measure some quantitiy and we have the following graphs:



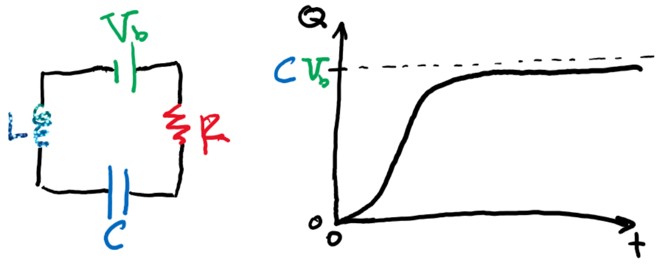
The nature of the data as $i \to \infty$ is similar to that of a charging RC circuit. So I will use that as a base point.



The asymptotic behavior of this circuit matches that of the logistic growth apparant in the coronavirus data, but the beginning of the graphs are very different. The derivative of the RC circuit graph increases too suddenly. In other words, $I = \dfrac{dQ}{dt}$ increases too suddenly, or $\dfrac{dI}{dt}$ is too large. A way we can fix this is to "penalize" changes in current, which is exactly what an inductor does.

3.5.2 LRC Circuit

So now we have an LRC circuit, which will model the data a lot better.



While this model already follows the logistic pattern, just like the COVID-19 data, it can be made better since the it does not spend enough time in the exponential portion of logistic growth.

3.5.3 LRC Circuit with Moving Target

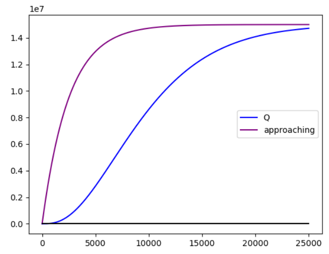
The way I chose to handle this is to modify the circuit so that it isn't approaching a fixed value. Instead it will approach a time-dependent value that approaches a fixed value from below. The effect of this is that it will slow the growth to the charge in the capacitor, causing the growth to spend a longer time in the exponential portion. But as $t \to \infty$ nothing will change since what the charge is approaching is arbitrary close to what it would have approached otherwise.

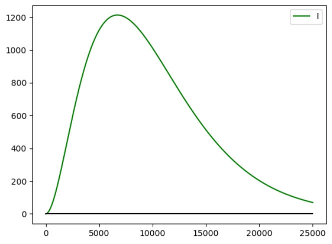
This behavior is similar to the ideas of all convergent sequences being Cauchy, expect for instead of having us sample a_m and a_n from the same convergent sequence, we are sampling them from different sequences that both approach the same target.

Overall, the effect it that is stretchs out the exponential portion without stretching out the later portion of the growth. We can change what the charge is approaching by changing the voltage source according to the equation

$$V(t) = \mu(1 - \exp(-t/\tau)) \tag{17}$$

Using this, I was able to generate a curve that models logistic growth very well:





To solve this circuit, I used a numeric differential equation solving method called the Runge-Kutta method. To do this, I created a state vector consisting of the charge in the capacitor and the current through the circuit, and then created a first-order vector differential equation:

$$f(\begin{bmatrix} Q \\ I \end{bmatrix}, t) = \frac{d \begin{bmatrix} Q \\ I \end{bmatrix}}{dt} = \begin{bmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} Q \\ I \end{bmatrix} + \begin{bmatrix} 0 \\ V(t) \end{bmatrix} \tag{18}$$

This equation was derived from Kirchhoff's Voltage Law. After that I just apply the Runge-Kutta method as follows

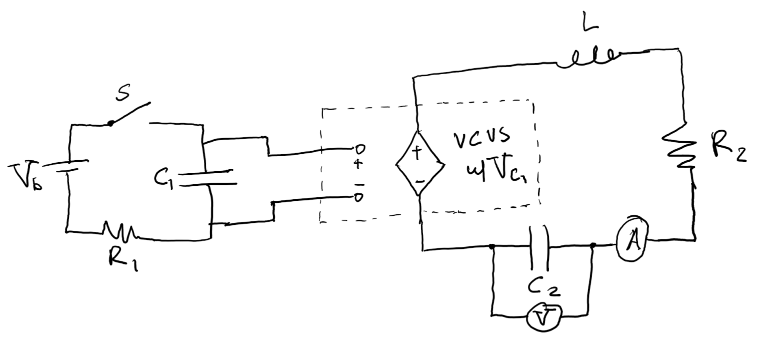
```
ts = np.arange(start, end, h)
Qs = []
Is = []

x = np.array([Q0, I0], float)

for t in ts:
    Qs.append(x[0])
    Is.append(x[1])

    k1 = h * f(x, t)
    k2 = h * f(x + 0.5 * k1, t + 0.5 * h)
    k3 = h * f(x + 0.5 * k2, t + 0.5 * h)
    k4 = h * f(x + k3, t + h)
    x += (k1 + 2 * k2 + 2 * k3 + k4) / 6
```

3.5.4 Building the Circuit

Building the basic LRC circuit is quite simple, but implementing the growth in the voltage source becomes a little more challenging. The way to create this effect to use a Voltage Controlled Voltage Source (VCVS) controlled by an RC circuit.

To break it down, the first thing to notice is that the growth of the voltage function $V(t)$ is exactly the same as that of the charge in or voltage across a charging capacitor in an RC circuit (with $\tau = RC$). So now we just have to use the same voltage generated by the capacitor in our battery without interfering with the RC circuit. A VSCS can read the voltage accross something and then reproduce a voltage proportion to it as a seperate voltage source.

Combining everything together, our circuit becomes



which will produce the graphs above by measuring the voltmeter (which is proportional the the charge) and ammeter and closing the swithc S at $t = t_0$. There are a lot of parameters to a setup like this:

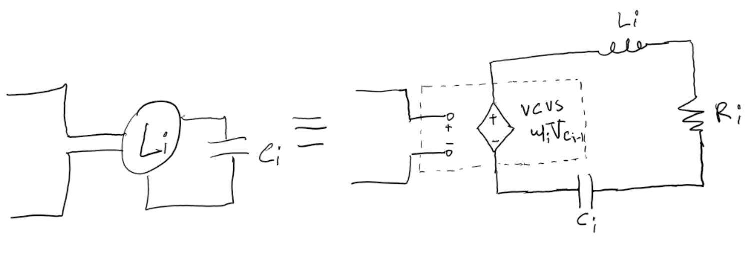
$$[V_b, t_0, R_1, C_1, Q_0^1, I_0^1, \mu, L, R_2, C_2, I_0^2, Q_0^2]$$

All these parameters will have the be tweaked to match the data you are trying to model.

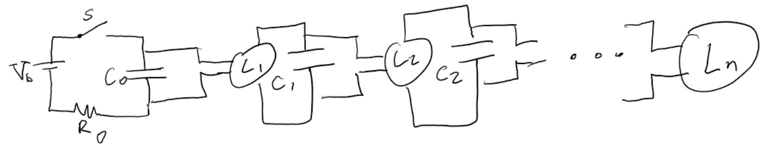
3.5.5 Possible Improvements

Another idea for a possible time-dependent voltage function $V(t)$ is a function the is similar to a logistic growth function as well. Such a voltage would slow down the exponential part even more, while still leaving the tail end of the function nearly untouched. The longer the exponential side of the voltage function is the longer our new logistic function will be.

The way to create a circuit like this is to chain together a similar setup we had for the LRC circuit multiple time. First let's define a new circuit element L_i as



and then we chain it together n times, as follows



This would create a lot of parameters, so a technique that can be used in for this (and even the simplier circuit models), is simulating the circuit and performing gradient descent to minimize a loss function. Then you are essentially training the circuit to model the data.

3.6 Neural Network

TODO: Code the neural network stuff

4 Conclusion

TODO: Get back to once I finish writing the rest

4.1 Country Ranking

According to trends using (R-squared statistic):
China Italy Spain Global France United States India
But this does not only have to do with the action taken since any time shifts in the data it not accounted for. Manually looking at the data to see which countries action was most effective, here is a new ranking:
Spain Italy China France United States Global India

4.2 Predictions

Not be over before End of May/ Start of June
Max Cases from auto analysis:

5 The Code

5.1 Libraries

5.2 Project Structure

5.3 RB Math Package

5.3.1 Gradient Descent and Models

5.3.2 Transforms

5.3.3 Plot Function

5.4 Reading Data

5.5 Plots