MA124 Maths by Computer: Assignment 2

The SIR model - flattening the curve

The SIR model is given by the following system of ODEs

$$\dot{S} = -\frac{\beta S I}{N}, \quad \dot{I} = \frac{\beta S I}{N} - \gamma I, \quad \dot{R} = \gamma I,$$

where S is the number of susceptible individuals, I is the number of infectious individuals, R is the number of recovered or removed individuals. N is the total number of individuals, i.e. $N = S + I + R$. One can easily verify that $\dot{N} = 0$ by adding the three equations. Hence N is constant. β and γ are parameters: β is the average number of contacts per person per time, γ is the reciprocal of the time an individual remains infectious. We are potentially interested in cases where the parameter β is not constant in time.

The SIR model can be used to understand important aspect of infectious diseases such as Covid-19. In particular it can be used to understand interventions that flatten the curve.

Part A. Baseline case (4 marks)

Assignment: Using Python, solve the above SIR model and examine the solution for one set of parameter values and one initial condition.

Details:

- Use the following parameter values for the baseline case: $\beta = 0.3$ and $\gamma = 0.1$, corresponding to $R_0 = 3$. Use the initial conditions $S(0) = 10^6 - 5$, $I(0) = 5$, and $R(0) = 0$.
- Produce two plots: a time series showing $S(t)$, $R(t)$, $I(t)$ and a phase portrait in the (S, I) plane.
- On the phase portrait, indicate the point on the trajectory where I reaches its maximum. (You will have to work out how to find this maximum point.) This is the peak in the infection curve. Print this maximum value.

You are expected to use the material in the SIR case study as a starting point. Hence there is little new to do here except compute, plot and print the maximum value of I. Hint you should find the appropriate NumPy function that helps with this. You will need not only the maximum of I, but also the value of S where that maximum occurs. Google around if you need to. DO NOT USE A FOR LOOP!

The next two parts of the assignment will consider ways to reduce the peak value of the infection curve, i.e., flatten the curve.

Part B. Prior immunity (8 marks)

One can study the basic effect of immunity with the SIR model by starting from an initial condition in which a percentage of the population is in the R class already. Think of these as individuals who have been removed from S and put into R, either because they have prior natural immunity or because they have been vaccinated. Either way, they are immune and hence not susceptible at the start of the disease outbreak.

Assignment: Illustrate how prior immunity flattens the curve by solving the ODE for a case with prior immunity and comparing with the baseline case.

Details:

- The parameter values and total population size are as for the baseline case. The initial value $R(0)$ is your choice. $I(0) = 5$ as before. $S(0)$ is whatever it needs to be so that $N = 10^6$. You should run some cases with different $R(0)$ and decide a value you like. Values in the range of 10% to 25% of the total population N are good. You can try $R(0)$ greater than 25% of N, but for the report do not exceed 25%.

- Produce two plots: a time series and a phase portrait. In each you want to show the baseline case and the prior immunity case. Hence six curves in the time series and two curves in the phase portrait. For the time series in particular you may need to vary line types to make things clear. Use legends.

- On the phase portrait, indicate the points on each trajectory where I reaches its maximum. Print these maximum values for each to quantify how much the curve is flattened.

- Optional (but recommended): Produce a third plot showing the time series for $I(t)$ with and without prior immunity. By excluding $S(t)$ and $R(t)$, the comparison between the two infection curves is clearer. Hopefully you have flattened the curve.

Hint: you obviously will copy-paste code from Part A. You should use new variables for the prior-immunity case such that the baseline solution computed in Part A is not somehow overwritten. Then all you need to do is solve the ODE for the new initial conditions and produce nice comparison plots.

Part C. Social distancing (10 marks)

One can model social distancing, or a mask mandate, by varying the parameter β that controls the rate of infection. We will use the following simple approximation. When a mask mandate is in effect, the parameter β will be reduced to 0.24, otherwise it will be 0.3, as in the baseline case. More specifically, we will consider β to change between these two values according to:

$$\begin{gathered}
\beta = 0.3 \quad \text{if} \quad t \le t_m \\
\beta = 0.24 \quad \text{if} \quad t_m < t \le t_m + 30 \\
\beta = 0.3 \quad \text{if} \quad t_m + 30 < t
\end{gathered}$$

t_m corresponds to the time the mask mandate goes into effect. We consider a 30-day mandate period and so $t_m + 30$ corresponds to the time the mask mandate is lifted.

In principle one need not restrict to such specific circumstances, but this will simplify things. Note, that with β varying with time, the ODE is no longer autonomous.

Assignment: Write a Python function to compute the right-hand-side of the SIR model with time-dependent β given by the above expression. Solve the ODE for different values of t_m and find a value of t_m that gives the smallest value for maximum value of I, i.e., find t_m that most flattens the curve. Plot this solution and compare with the baseline case.

Details:

- As for the baseline case, use $\gamma = 0.1$ and the initial condition: $S(0) = 10^6 - 5$, $I(0) = 5$, and $R(0) = 0$.
- You do not need to write Python code to search over different values of t_m. Use trial and error. Also, you do not need to find the exact best value of t_m but find a value that is within ± 5 days of the best choice. You do not need to show how you determined t_m. One you have decided t_m, set that value in your function.
- Produce two plots: a time series and a phase portrait. In each you want to show the baseline case and the mask-mandate case. Hence six curves in the time series and two curves in the phase portrait. For the time series in particular you may need to vary line types to make things clear. Use legends.
- On the phase portrait, indicate the points on each trajectory where I reaches its maximum. Print these maximum values for each to quantify how much the curve is flattened.
- Optional (but recommended): Produce a third plot showing the time series for $I(t)$ with and without the mask mandate. The comparison between the two infection curves is clearer. Hopefully you have flattened the curve.

Note, if you have more experience with programming, you may use some more sophisticated methods to programme the time dependence of β in your right-hand-side function.

Further 3 marks

A further 3 marks will be awarded for each assignment based on overall quality and clarity of the submitted notebook.

Submission

You will submit a single Jupyter notebook.

- The last thing you should do before submitting the notebook is to Restart Kernel and Run All Cells. You should then save the notebook and submit the .ipynb file. **You will**

lose one mark if you submit a notebook that has not been run.

- A template is not provided. You are expected to begin from the SIR model case study.
- There are no challenge marks this week and hopefully everyone will be able to get high marks on the assignment.
- Clearly this assignment lends itself to producing a nice document. A good guide for what the final notebook should be like is: if the notebook is run and all code cells are collapsed, the notebook should be readable as a **short** report, primarily consisting of a short intro to each section followed by figures and descriptions of the figures. Think of the Markdown content as extended figure captions. Use the SIR model case study as a guide.
- Use should also use the comments in the SIR model case study as a guide to the level of comments expected. Each code cell should begin with a comment line or lines concisely stating what the cell is for. Functions should have comments or docstrings describing what they do. One assumes the reader understands Python. Add comments to set off blocks of code or to note anything tricky. In most cases Python code explains itself and does not need comments.

In [ ]: