EN2106 Computing 1, Autumn Semester 2020

Updated 25/11/20

Coursework Instructions

Submission deadline: 12:00 midday, Friday December 18th (Week 10)

This coursework represents 30% of your overall mark for the module

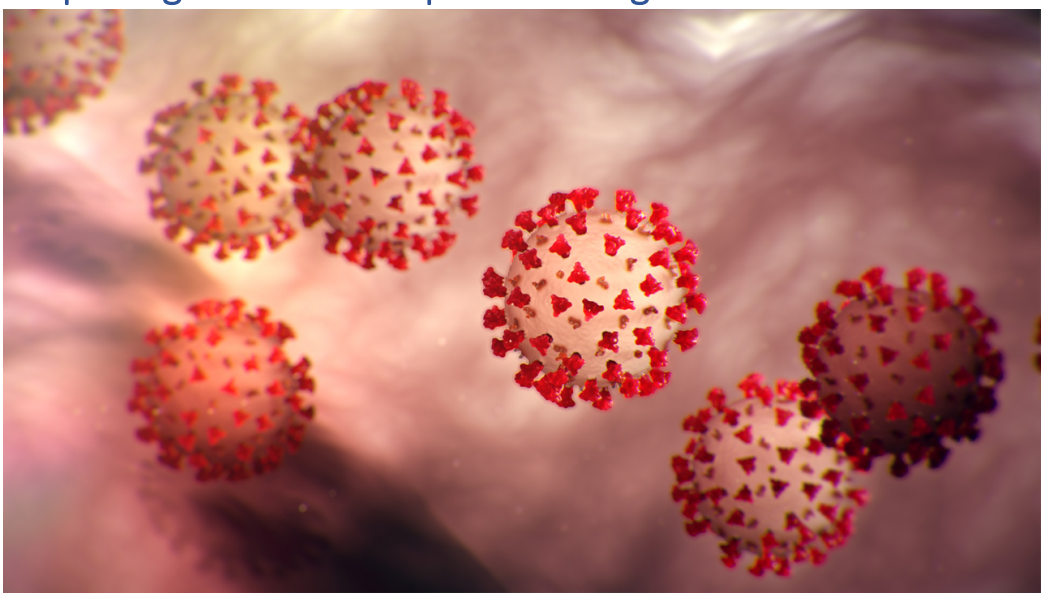
Dr. Daniel Mitchard (mitcharddr@cardiff.ac.uk), Dr. Daniel Gallichan (gallichand@cardiff.ac.uk)

Dr. Matthew Allmark (allmarkmj1@cardiff.ac.uk)

You will need to upload 2 files to Learning Central (2 separate submission pages)

1. Your MATLAB files contained within a single zip file named 'c<StudentID>.zip' (e.g. 'c1000001.zip' for a student ID of 1000001)
2. Your report in Microsoft Word or PDF format, including a signed plagiarism declaration (use the same filename as above, but with '.docx' or '.pdf' extension).

Exploring Coronavirus spread through data and simulation



Important Information

As detailed on the front page you are expected to upload two files to the appropriate submission pages on Learning Central. One is a zip file containing all of the MATLAB m-file scripts and functions that you have created for the coursework – and the other should be your written report.

The coursework will be marked as follows out of a **total of 75 marks**:

- MATLAB code submitted in the zip file. One m-file can be used for all parts of Task 1, separate files should be used for each part of Task 2 and any functions written should be included as well. This part is worth **37 marks** (distributed as indicated throughout this document).
- Report (single document submitted in PDF or MS Word format) containing code output, figures and brief descriptions of how you completed each task. This part is worth **30 marks** (distributed as indicated throughout this document).
- Additionally, **4 marks** are awarded for appropriate commenting within your code and **4 marks** are awarded for following all of the upload instructions.

The report is your opportunity to provide an explanation of the approach you used and how your code works – alongside the figures that your code outputs. If you are submitting incomplete code or code that does not work, you can use the report to explain what you tried and the problems you experienced. You do not need to detail how functions work that were covered already in EN2106 teaching but you should explain your understanding of any other functions you have used (e.g. from MATLAB help).

You must include the plagiarism declaration at the start of your report followed by a section for each task in the coursework. Unlike a lab report, essay or thesis, you do not need to include an introduction, theory, results or a conclusion. You do not need to use all of your word limit but should not exceed it.

For inclusion of figures: if you are using the desktop version of MATLAB you can simply use 'copy' and 'paste' to include them into your report. If you are using MATLAB Online it may be easiest to take a screenshot of the plots and crop them before putting into your report.

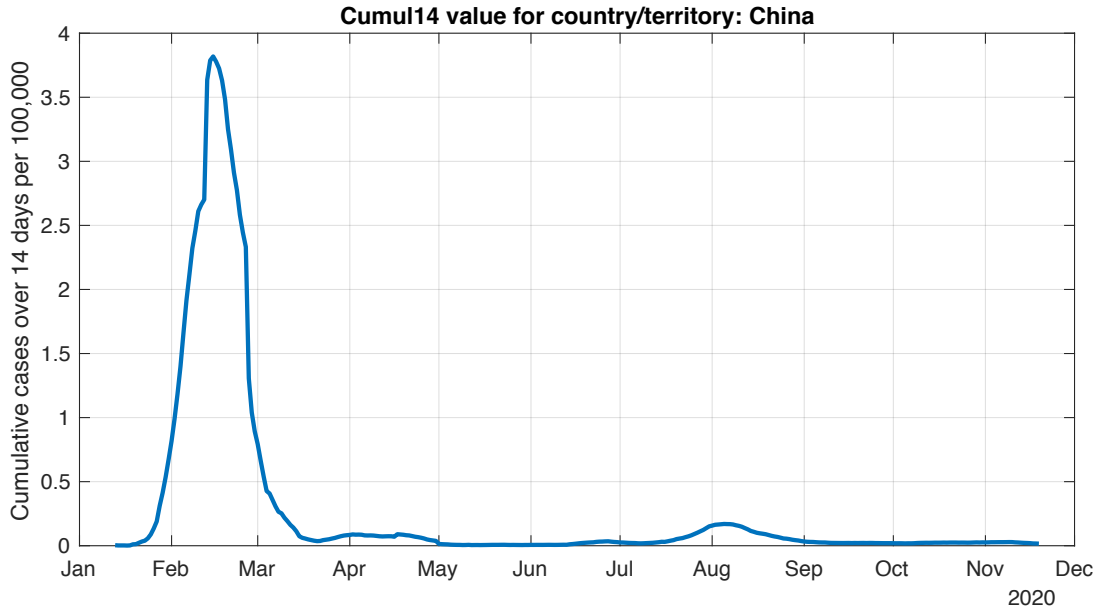
Marks are given for correct use of English and grammar, following the guidelines and staying within the word limits, thorough and concise explanations which correspond to your code and clear and correctly displayed figures. The word limit is 200 words per subtask (Task 1(a), 1(b), etc), so a total of 1,400. You do not have to use all your word limit but must not exceed it. You must include the code output and graphs as outlined for each task.

Task 1 – Analysing the data

Start by downloading the 'EN2106_coursework.zip' file from Learning Central – and unzip this to somewhere you can work with in MATLAB.

Data is publicly available from the European Centre for Disease Prevention and Control website which monitors the reported cases of coronavirus across the world. This data has been downloaded as the file 'ecdc_covid_data.csv' which you can use for this Task – and is included in the zip file.

Load the file 'process_covid_data.m' into MATLAB and make sure you have the data file in the same folder. Run the template script and make sure that you can see a curve like the one shown below:



Important!

If you are having problems getting this to work, make sure you post details of your problem to the EN2106 Discussion Board so that we can make sure you can get started with this exercise!

If you check the comments in the template script you will see that the column of data which corresponds to cumulative COVID-19 cases over 14 days per 100,000 people has been copied into a new table column called 'cumul14' – as this is an easier variable name to work with for the rest of the analysis.

You may find it useful to browse the data (either in Excel or in the variable viewer in MATLAB, accessed by double-clicking the 'coviddata' variable in the Workspace) to get an idea of how the spreadsheet of data is organised and what the different columns of data correspond to.

Read through the comments in the template script and make sure you understand what all the variables correspond to and that you understand how the code works which plots the curve for a single country. Look through the 'countries' variable and change the value for 'iC' manually to a few different countries to get an idea of what the different curves look like.

For each part of Task 1 you can add additional code to the same 'process_covid_data.m' script – and when you are finished include this file in your zip file for upload.

Task 1(a) – Find the 10 most populated countries

You will notice that one of the columns of data contains the population of the country (or territory) from 2019. Your task is to use this data to identify the 10 most populous countries (or territories) in this dataset.

You will need to:

- Define a new variable (initialise it with zeros) as a column vector which matches the size of the 'countries' variable which will hold the population for each country
- Create a for-loop which runs through all the countries
- For each country in the loop, identify its population and fill the population variable you have created
- It turns out that a couple of the countries do not have valid population data, resulting in 'NaN' (not a number). Replace these values with zero for your population variable.
- Now use the 'sort' function to sort your population variable. Check the MATLAB help for the 'sort' function – and use two outputs so that you can store not only the sorted population values, but also the indices which correspond to the countries in their sorted order. *Note that 'sort' defaults to sort in ascending order, and you will want to sort by descending order if you want to identify the highest populations.*
- Use a for-loop and the 'disp' function to display the 10 most populous countries in order, along with the population of each of those countries.
- Copy & paste (or use a screenshot) of the output list of ten most populous countries in the Command Window and include this in your report.

[Code: 5 marks, Report: 4 marks]

Task 1(b) – Create curves for the 5 most populous countries

Use a for-loop to create a new figure which overlays the curves of the 5 most populous countries (making use of the output of your sorting from Task 1a).

Add a legend to your plot which identifies the 5 countries. To generate this, **do not** type the country names manually, but instead use the fact that you can pass a cell array of labels to the 'legend' function. Also, check the help for the 'legend' function so that you can position the legend in the top-left corner instead of the top-right so that it does not overlap with the curves that have been plotted.

Copy & paste / screenshot the graphs you produced into your report.

[Code: 4 marks, Report:4 marks]

Task 1(c) – Find the countries with the highest maximum of cumulative cases per 14-days and create curves for the 5 highest

- Create a new variable which will store the maximum value for the 'cumul14' data for each country. Use a for-loop to fill this variable with the appropriate value.
- As in Task 1a you will need to replace a few values of NaN with zeros.
- Given that the 'cumul14' value is not so reliable for countries with smaller populations, set the value of your maximum to zero for countries with a population less than 1 million (to effectively remove them from the comparison).
- As before, create a list of the 10 highest ranking countries with this metric, and display their names alongside the value for the max in 'cumul14'.

- Use a for-loop to create a plot which overlays the 5 highest ranking countries according to this metric.
- Add code below the for-loop to add the curve for the United Kingdom to the plot for comparison.
- Add a legend located in the top-left of the plot.
- Copy & paste / screenshot the graphs you produced into your report.

[Code: 5 marks, Report: 4 marks]

Task 1(d) – Find the first new countries that might have been added to a quarantine list

We can assume that in Europe the first wave of the virus was mostly over by June 2020. Data by country worldwide is then also perhaps a bit more consistent as there was more widespread use of mass testing at that time (which was typically not available during the first wave). In the UK, it was on 8th June 2020 that the first quarantine rules were introduced, so we will use this as the cut-off point for our hypothetical analysis.

Your task is to identify the first 10 countries for which the 'cumul14' value goes past a threshold value of 100 cases per 100,000 **after** 8th June 2020.

To do this you will need to:

- Create a new column vector to store the date that the threshold is exceeded – which should be of 'datetime' datatype. Because 'zero' doesn't make sense as a time, we can initialise this vector to 'NaT' ("Not a Time") which is the equivalent of NaN for the datetime datatype.
- Loop through all countries to fill this column vector with the appropriate values
- Inside the loop, identify the rows of the spreadsheet that correspond to each country as before – and add additional conditions which narrow the selection to only dates after June 8th and only 'cumul14' values above the threshold. You can add multiple conditions here by using a single ampersand (i.e. '&') between them (you previously met the double ampersand, i.e. '&&', which requires scalar results, whereas here we are dealing with vectors).
- For each country check what the 'cumul14' value is on 8th June itself, and if this is already above 100 then do not count this country towards the list (as it would already be quarantined). The easiest way to not include the country is to keep the NaT value for the identified date that the threshold is exceeded.
- Display the first 10 countries which meet these conditions, alongside the date which they pass the threshold for 'cumul14'. Note that you will need to use the 'char' function on the datetime variable in order for it to be converted properly to a string for display.
- Plot the 'cumul14' curves for the first 5 countries which meet these conditions
- Copy & paste / screenshot the graphs you produced into your report.

[Code: 5 marks, Report 4 marks]

Task 2 – Modelling progression of an epidemic

There are various mathematical models that have been proposed which try to capture elements of the dynamics of how a virus might spread through a population. One such model is known as the SIR model, where the population is considered in three compartments, S = Susceptible, I = Infectious and R = Recovered. The 3 governing equations which model the rate of change of the size of each compartment are as follows:

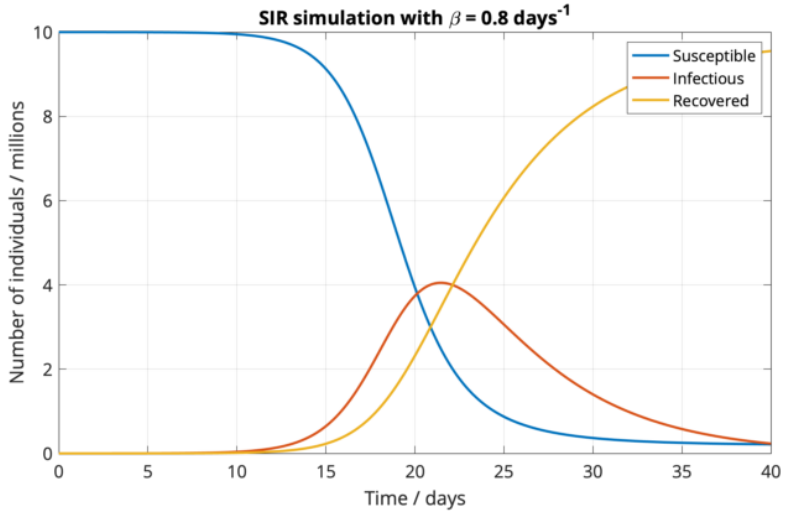
$$\frac{dS(t)}{dt} = -\frac{\beta S(t) I(t)}{N}$$
$$\frac{dI(t)}{dt} = \frac{\beta S(t) I(t)}{N} - \gamma I(t)$$
$$\frac{dR(t)}{dt} = \gamma I(t)$$

Where β is given by $\beta = ab$, and a is the rate of an individual contracting the disease when exposed to an infectious person, and b is the number of people that an individual comes into contact with per unit time. The parameter γ represents the inverse of the average amount of time that an individual remains infectious for before moving into the 'recovered' category. The parameter N represents the total number of individuals in the population.

The model is obviously very simplistic and makes several assumptions, and one of the key assumptions for interpreting the shapes of the curves is that recovered individuals will have gained immunity and cannot be reinfected.

Task 2(a) – Plot for a simple case

Create a script (call it 'task2a.m') which uses the Euler Method (which you learnt about during this module) to find the solution to these equations given the following parameters: $\beta = 0.8$ days^{-1}, $\gamma = 0.2$ days^{-1} and $N = 10{,}000{,}000$. You will also need the following boundary conditions: $I_0 = 100$, $S_0 = N - I_0$ and $R_0 = 0$. Create your simulation using 1000 timesteps over 40 days. You should create a graph similar to this one (which you should include in your report):



You can create the symbol β in a MATLAB title by typing `\beta` *and you can add superscript text by using, e.g.* `days^{-1}`

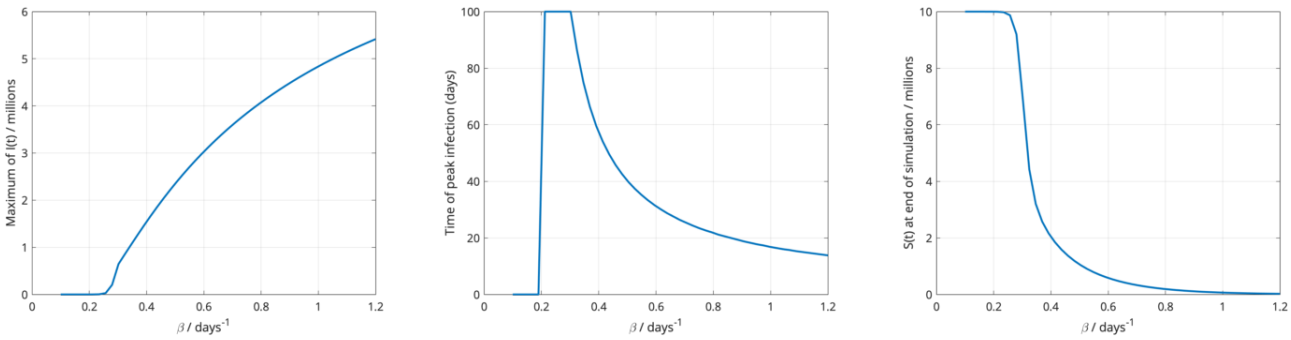
[Code: 6 marks, Report: 4 marks]

Task 2(b) – Flattening the curve

You will notice if you change the value for β in the simulation you created in Task 2(a) that the shapes of the curves all change – and critically, the maximum number of infectious individuals can vary a lot. Lower values of β correspond to lower infection rates and a lower maximum number of infectious individuals – this corresponds to the phrase that is mentioned often in the media reporting about coronavirus of 'flattening the curve'. This can be very important as we might assume that a certain proportion (here we will assume 1%) of infectious individuals require hospital treatment in an intensive care unit (ICU) – and there are typically a very limited total number of spaces in the ICU.

For this task you will investigate how the maximum number of infectious individuals varies with β and find the value for β which would keep the number of individuals requiring ICU admission below the capacity.

- Start by turning your code from Task 2(a) into a function which calculates $S(t), I(t)$ and $R(t)$ for a given time vector t, based on the parameters β, γ and N.
- Create a new script (call it 'task2b.m') which you will use to call your function
- Use a time-vector running from 0 to 100 days in 1000 timesteps
- Generate a vector of β values to test – a linear range from 0.1 to 1.2 in 50 steps
- Use a for-loop to test each of these values in your SIR-model function
- Inside the for-loop, also calculate values for the maximum value of $I(t)$, the time at which this maximum occurs and the value for $S(t)$ at the end of each simulation
- Generate plots for each of these values over the range of β being tested – you should end up with plots like the following:



If we assume that there are 2000 ICU beds available in total and that at any time-point 1% of the number of infectious individuals require ICU treatment, find the critical value for β where the threshold capacity is exceeded. To do this it will be necessary to narrow the range of β values to focus on the region of interest, then use MATLAB code (rather than simply judging by eye) to find the critical β value. You should also create a plot of your chosen range of β values with the critical value for $I(t)$ shown as a dashed line to confirm that these lines intersect at the critical β value.

Include all of your plots in your report.

[Code: 6 marks, Report: 4 marks]

Task 2(c) – The effect of intervention

Create a copy of your script from Task 2(a) and call it 'task2c.m'. We saw that with $\beta = 0.8$ days^{-1} the peak in infectious individuals is around 4 million (around 40% of our population). Looking at the whole curve it also appears as though the number of infectious individuals only rises significantly after 10 days – but of course this is because of the scale of the y-axis. Let's look closer at what is happening in those first 10 days.

- Change the timescale of your simulation to 0 to 10 days (keeping with 1000 timesteps)
- Remove $S(t)$ and $R(t)$ from your plots to focus only on $I(t)$
- You should see that $I(t)$ is growing exponentially during the first 10 days

Now modify your code so that at $t = 8$ days there is a reduction in the value for β, from $\beta_0 = 0.8$ days^{-1} to $\beta_{new} = 0.1$ days^{-1}, as we assume that government interventions have been introduced. Change the timescale of your simulation now to 0 to 150 days (keeping with 1000 timesteps). Experiment with several different values of β_{new} between 0.1 and 0.3 days^{-1}.

You will notice that there is a significant transition in the shape of the curve that occurs at $\beta = 0.2$. This point corresponds to where the ratio $\beta/\gamma = 1$. This ratio also corresponds to the 'R number' that you will have heard about in the media – the average number of further new infections resulting from each infected individual. To try to avoid confusion with $R(t)$ in the SIR model, we will refer here to the R number as R_n. **Briefly discuss in your report** the significance of keeping R_n below one – including consideration of which contributions to R_n can be altered by government intervention (such as social distancing and the wearing of masks) and which are fixed by properties of the infection. Support your discussion with two plots, one which overlays curves which correspond to R_n after 8 days of 0.9, 1.0 and 1.1 – and a second which overlays curves which correspond to R_n after 8 days of 1.1, 1.2 and 1.5 (two plots are used to better appreciate the changes in scale of the y-axis).

[Code: 6 marks, Report: 6 marks]