COSC2002/2902 Computational Modelling

Assignment — due Sunday May 31st, 11:59pm

The assignment consists of two questions. It is worth 10% of your total assessment for this Unit of Study. You will be marked on the correctness and quality of the code, as well as your written responses to the questions.

You should submit your assignment as a jupyter notebook through the Canvas submission portal. Use your SID in the filename.

Written responses to assignment questions should also be included in the jupyter notebook, using the ‘markdown’ option for a cell (so that your whole notebook can still be run without error). Check that your notebook runs before submission.

You are welcome to work in pairs. If submitting as a pair please both submit the notebook (with both your SIDs in the file name), so that Canvas records a submission for both members of the pair. If you submit this assignment with another student, you certify that you have both made a fair contribution to it, and are happy to both receive the same mark.

Assignments that have simply been copied will not be accepted. Copying the work of another person without acknowledgement is plagiarism and contrary to University policies. By uploading your submission to Canvas, you are certifying that you have read and understood the University Academic Dishonesty and Plagiarism in Coursework Policy.

Question 1 (10 marks)

Given the impact it has had on all our lives it seems we should have a go at modelling the spread of a virus through the community.

Let's approach this problem in this way. Begin with a 2-dimensional grid of squares (the "world"). A number of healthy people are randomly distributed throughout the world, as well as one infected person. Each day, each person either stays still or moves one square up, down, right, or left, or one square diagonally. If an infected person spends a day on the same square as a healthy person, the healthy person becomes infected.

Your implementation should have four parts. First, specify the parameters of the simulation. Second, initialise the people in the simulation. Third, run the simulation (i.e., at each timestep, update the locations of each person and whether or not they have been infected). Finally, plot and output the results.

Parameters

The simulation should be initialized with four parameters: the size of the 2-dimensional world people inhabit (`sidelength`, default 40 squares per side), the maximum time to run the simulation for (`maxtime`, default 1000 days), the number of people (`npeople`, default 100), and the rate at which people can recover from the virus (`recovery`, default 0).

People

Each person will be associated with a number of pieces of information: (i) their x coordinate (an integer between 1 and the size of the world), (ii) their y coordinate (same), and (iii) whether or not they're healthy, infected, or recovered.

Simulation

The "simulation" consists of the calculations that occur inside the loop over time.

First, we need to figure out where people move to. For this, we'll need an inner loop over people. At each timestep there are nine options, corresponding to the eight immediate neighbours of a person, and their current position (meaning they don't move). If people move outside the world ($x, y < 0$ or $x, y >$ `sidelength-1`), they come through the other side of the world (i.e. we shall assume we have periodic boundary conditions). So, a person who moves to $x = -1$ will instead move to $x =$`sidelength-1`.

Second, we need to figure out which people are infected, and which are not.

Third comes the tricky part: we need to handle infections. To do this, we can loop over just the infected people, then loop over all the healthy people. For each infected/healthy pair, we need to check whether the healthy person is occupying the same square as the infected person (i..e., their x and y coordinates are identical), and whether or not they have already recovered from the infection. If they are healthy and haven't recovered then they become infected.

Finally, for each infected person we need to check if they spontaneously return to being healthy: i.e., with probability `recovery` at each time step the infected person becomes a 'recovered' individual, and so is resistant to reinfection.

Plotting

Construct a scatter plot, where a dot shows the position of a healthy, infected or recovered person on the xy plane. Plot healthy people in blue, infected in red, and recovered in green.

Questions

(1.1) If there is no recovery, what is the average length of time until the last person gets infected? Provide also an estimate of the uncertainty in your value.

(1.2) Averaging over many runs: Plot the number of infected people as a function of time.

(1.3) If you halve the size of the world (i.e., set `sidelength=20`), how does this change the time it takes until the last person is infected? Briefly discuss the real-world implications of this result.

(1.4) Now let's allow people to recover. Using the original size of the world (`sidelength=40`), change the recovery rate from 0 to 0.01 (i.e., every 100 days on average an infected person will spontaneously recover). Run the experiment a few times and then describe qualitatively what you observe. Do these results look realistic? For a representative case plot the number of infected people, the number of healthy people, and the number of recovered people versus time on the same set of axes.

(1.5) Let's now explore the effect of social distancing measures. We could do this by changing the size of the grid dynamically, or changing how quickly people move (reducing the chance of interaction), but an easier way is to simply change the recovery rate. If we increase the recovery rate then this is similar to making the grid larger (or have people move more slowly) as the key parameter is the R_0, the number of people each infected person infects. This number will go down on average if the infected people recover before they get a chance to infect many people. By running a number of experiments determine the average infection peak fraction as a function of the recovery parameter. Plot also the average total fraction who end up being infected as a function of the recovery parameter. Comment on your findings with respect to the idea of 'flattening the curve'.

(1.6) **[COSC2902 ONLY]** Plot the R_0 parameter as a function of time (on average) for the case of a recovery rate of 0 and a recovery rate of 0.01. Describe any implications of your observations.

Question 2 (10 marks)

Looking at a bus timetable it is mysterious that often we seem to be waiting much longer than we should be. Fortunately with the development of real-time bus tracking the need to look at bus timetables is a thing of the past. However, we still might like to know how long we expect to wait at a bus stop if we arrive randomly and just hope for the best. In this question we shall simulate waiting for buses, under various assumptions about bus arrival times, and try to determine the average wait time experienced by people at a bus stop. We shall compare our simulated bus statistics with example real-world data, and seek to conclude how accurate our assumptions and conclusions are.

We shall look at three different bus models, all with the same average time of 10 minutes between bus arrivals:

Model 1: We assume there are N buses in time T_N and that $T_N/N = 10$ minutes per bus, i.e. on average the buses are 10 minutes apart, but the buses could arrive any time between $T = 0$ and $T = T_N$; Note, we are not saying that buses arrive exactly 10 minutes apart, only that on average they are 10 minutes apart (this is an important distinction!).

Model 2: We assume the buses are scheduled to arrive every 10 minutes, but that they are uniformly distributed between being up to x minutes early, and up to x minutes late, i.e. 'lateness' is a uniform distribution from $-x$ to $+x$;

Model 3: We assume the buses are scheduled to arrive every 10 minutes, but that the nth bus is offset from the scheduled arrival time by d_n minutes, where d_n is chosen from a normal distribution with mean 0 and standard deviation σ.

Note that in Model 2 and Model 3 it is possible for a scheduled bus to arrive **before** an earlier scheduled bus. The models thus allow for the possibility of a bus overtaking another bus that left the depot earlier, which is possible for bus stops towards the end of a bus route (where there's the most variability in bus arrival times). Practically, this means we should make sure the bus arrival times are sorted to appear in increasing order, before we seek to determine how long a particular passenger has to wait.

A basic strategy to determining passenger wait times would be the following:

- Set up bus arrival times, in order of increasing time;
- Set up all passenger arrival times, in order of increasing time;
- For each passenger, identify which bus the passenger will be getting on;
- For each passenger, determine the passenger wait time.

To proceed we should also decide on the number of buses and the number of passengers. The more buses, the better will be your statistics. For the number of passengers this will be determined by the average number being picked up at the bus stop. An average of between 5 and 10 would seem to be reasonable, so total passengers will be 5 (or 10) times the number of buses. Note (again) we are not requiring that there are exactly 5 (or 10) passengers on each bus, only that on average there are 5 (or 10) passengers per bus.

(2.1) For each Model carry out the following (for Model 2, take $x = 5$, and for Model 3, take $\sigma = 4$):

(i) Confirm that the average interval between buses in each case is 10 minutes;

(ii) Determine the average waiting time for people;

(iii) Plot a histogram of the interval between bus arrivals;

(iv) Identify the nature of the distribution of bus intervals for each model.

(2.2) How does the estimate of average wait time depend on x (for Model 2) and σ (for Model 3)? A short explanation of the main observations is sufficient. How do the distributions of bus intervals change with these parameters (again a short paragraph of the main observations is sufficient).

(2.3) Real-world bus data is available on Canvas, "Route_1_minutes_late.csv" and "Route_2_minutes_late.csv". These files contain a list of times for buses compared to their scheduled arrival times. A negative number means the bus was early.

(i) Read in the data and plot histograms of the minutes late for each route;

(ii) Compare the data to predictions from your simulations. Which model seems to be the best fit for the real-world data?

(2.4) What can you conclude about the wait time for passengers? From the timetable, it is expected to be 5 minutes. From your simulations, and comparison with real-world data, is this reasonable, or can we expect a different wait time? A comment with reference (i.e. supported by) your earlier results is sufficient.

(2.5) **[COSC2902 ONLY]** In part (2.3) we looked at 'minutes late' for real bus data. The problem is, in our simulations we looked at passenger wait times, and we can't obtain this information from the data file as it is. Let's look instead at the original 'raw' bus data, contained in the file "arrival_times.csv". This file contains scheduled and actual arrival times for two bus stops. Use this data to explore the following questions:

(i) Assuming the buses run exactly on schedule, plot the average time between buses versus time of day. How reasonable is our assumption of a fixed average time (i.e. our assumption of buses arriving 10 minutes apart)?

(ii) Explore the data and propose a possible computational model we could use to explore passenger wait times (you do not need to actually provide code, just an outline of your proposed model).

(iii) Explore the data and identify one interesting thing that you conclude from your analysis.