Computational Physics Final Report

Joshua Humphrey

May 2025

1 Introduction

Our task for our final was to create a model of a pandemic; the exact thing a graduate student is doing for their research. Fortunately for us, the model we were asked to create is a rather simple model from which we can develop a better understanding of pandemics (despite living through one already) and of coding. I will do what was asked of me and discuss the coding decisions I made within the assignment. Following this will be a couple of tests of the model: how does the operation time increase as the number of people increase, and if the model is able to replicate "flattening the curve" through reduced infection chance.

2 Code Analysis and Results

Beginning with the first part that fell into place, I very quickly decided that I would want to utilize Python's affinity for objects and rather literally objectify people. In this case, this is pretty useful to do. Being healthy and being immune aren't the only things to keep track of; since these people are moving I also want to store their positions and velocities. Since these people might potentially die when sick, I need to store if/when they die and how far in the disease they are. Storing all of these in arrays as is simply wouldn't work, but storing the data related to the individual inside the individual solves a lot of headaches that not using classes would invoke.

The actual model has a few assumptions built into it that simplify live incredibly that should be discussed before getting into the technical details. Firstly, we are only interested in the active period of people's days where they are actively moving around. Thus, days in this model can effectively be thought of as having only 12 hours. Secondly, reinfections are not considered in this model. If an individual survives the disease, they are effectively immune to it. Speaking of the disease, it lasts 10 days, and the presumed infection rate and mortality rate are the same, at 50 percent each. In other words. for these fictional people, a coin toss determines if they live or die. Once infected, the coin is "flipped"; i.e, either a 0 or 1 is randomly generated, if it's a zero it's game over for that individual at some point in the middle of their infection. If it's a one, this algorithmic universe predetermined that the individual would survive. Potentially a more sophisticated solution could exist where the chances of dying is related to some function in order to open the door to potential minimization problems, but that is beyond the scope of this, supposed to be simple, model.

Speaking of these peoples' universe, I'd like to discuss the main loop of the program. It will go on for 500 timesteps, unless otherwise specified for the version without animation. For every interval, it will move one of many persons and check to see if it is too close to another individual. If so, then it will run the sickness interaction, if one of the individuals is sick, and the physical interaction if they are incredibly close. This interaction is arguably the most complex; my attempt at a solution was whenever two people get too close to one another, I check the sum of their velocities. If the component in either the x or y direction is close to 0, then that means the two velocities were heading towards one another. I chose to invert the velocities in that direction only for both the participants in this collision. This movement system is not without its flaws, however. Suppose there are two particles heading in roughly the same direction. Their velocity components will add up to be over the condition set. Unfortunately, this can result in some weird behaviour. However, I think for the purposes of a simple model this is relatively allowed. The recommendation was to invert both the components of both participants velocities upon any collision. Again, if the two objects are moving in the same direction, then if they happen to collide they both invert to move in the same direction again, causing a potential loop if neither of them can escape colliding with the other. The attempt at considering only the part of velocity that's actually doing the "colliding" was also an attempt to alleviate this issue. This system isn't perfect; however, we are only interested in a simple model and I believe that this third assumption that this movement is fine enough is a fair enough one. However for a more accurate model, I think that an updated movement system would be necessary. Whilst this is just a project meant to be a final, this would be an area for "future research" to improve this program.

Finally, the overall velocity of each individual is relatively high. This is both another way to alleviate the issue of people "sticking" to one another via constantly inverting their directions, and to ensure that a pandemic does spread. Part of this problem, unique to me, is that my mortality rate and my infection rate are exactly the same. If individuals weren't moving around a lot, I could potentially never have a "pandemic" since it was just one randomly determined person who never got a chance to meaningfully spread it. Therefore, having the objects have a decent amount of speed seemed to be a good option.

3 Computational Time

Besides simply defining parameters and methods that could be written outside the class, there's not much more to discuss about how the model works. Moving onto testing the model, having more people in a simulation can increase the computational power required in a significant, non-linear way.

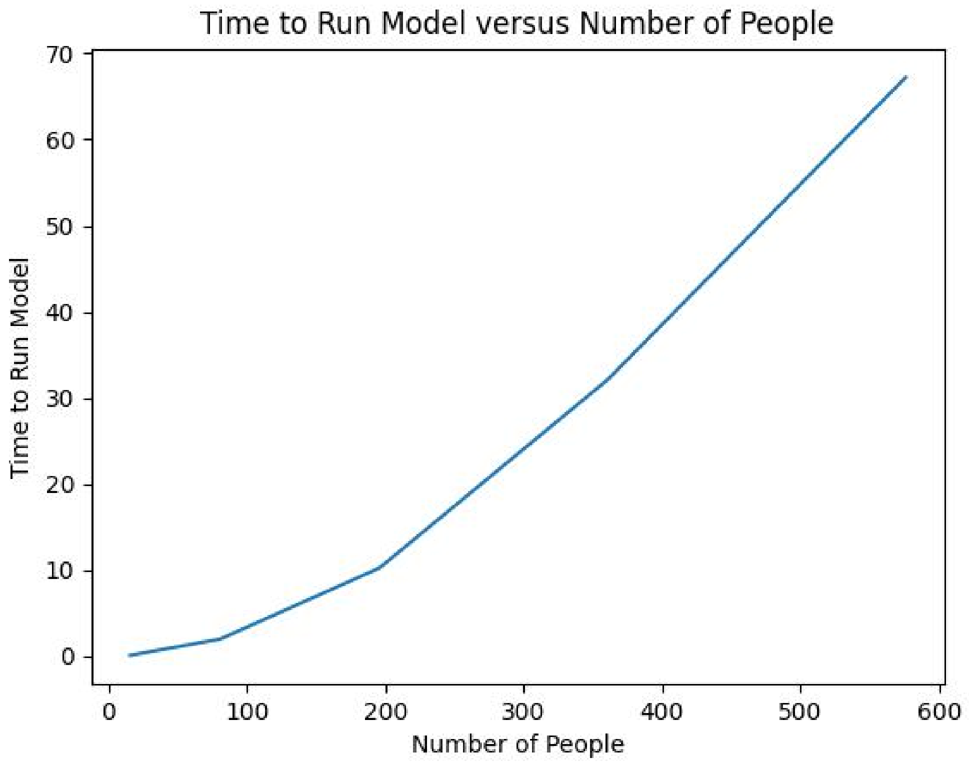

Figure 1: A plot of how long it took to run the model, using 5 different data points to generate the trend shown

The above illustration shows the obvious: the computer takes more time to think since it has much more to consider. That's why the increase isn't exactly linear, even adding one new person requires the computer to consider how that new person interacts with the rest, if it does at all. However, if there are not many people, it's not really a pandemic. The sweet spot I decided upon using for the next task was 196 people, since it took around 10 seconds to run through.

4 Flattening the Curve

The previous point was testing the model computationally; now, we want to replicate flattening the curve. To do this, the infection chance for those infected lowered by 10 percent. To replicate this effect by making a random integer from 1 to 10 inclusive. If the person running the "sickness check" was masked, they would become sick if the random number was 4 or lower. Else, they would become sick if the random number was 5 or lower. Below are my results:

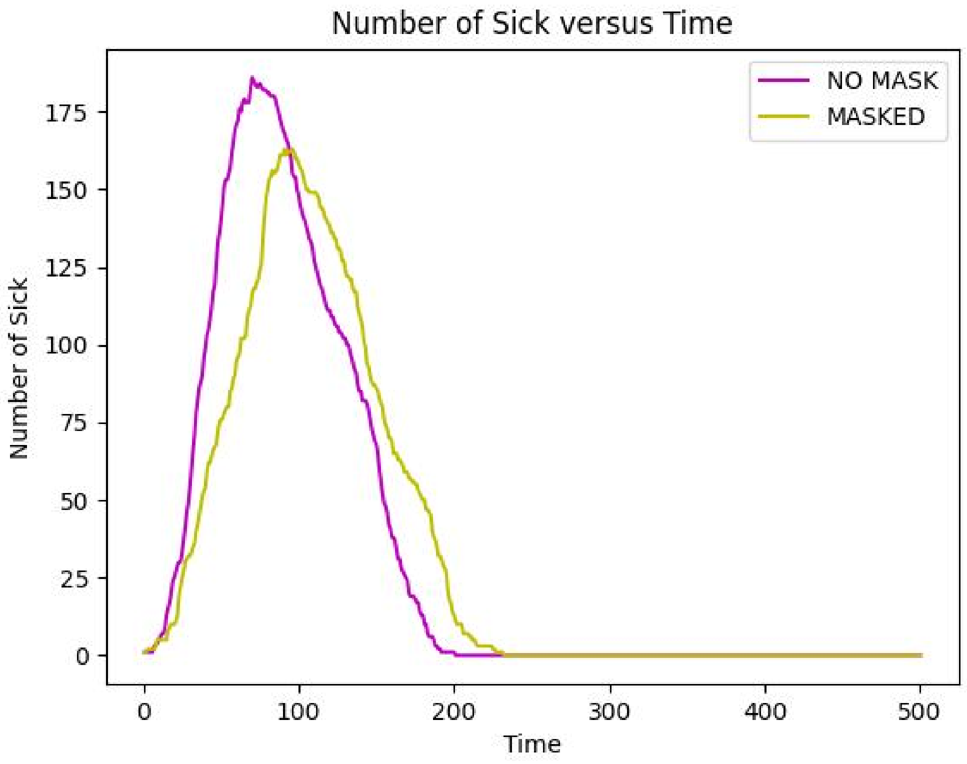

Figure 2: Recreating ”flattening the curve” as was popular to hear about in 2020. Everyone was masked for the masked simulation.

Notably, the peak for the simulation where everyone had masks is lower and shifted over compared to the one without masks. When creating this chart, the velocities of the people were significantly reduced. Whilst there were concerns not everyone would get infected if so, this was done in order to better show the effect. With the higher speed, people could just move and spread the disease fast in spite of the reduced infection chance. Otherwise this code does not differ from the code for timing the model.

5 Conclusion

Fortunately, or perhaps unfortunately, for the graduate student whose expertise is exactly this, creating even this simple model had its challenges. There are some incredible simplifications done, but in spite of those we were able to observe the non-linear relationship between computational power and number of people in the model. Moreover, we were able to briefly imitate life via flattening a (fictional) curve. Perhaps this is in line with the rest of the physics journey so far, where problems are at first simplified then get complicated. Perhaps creating models to recreate trends we've seen in order to validate them hints towards some growth in our computational skills. Or, and hear me out, this was just the assignment I had to do.