2G or not 2G

Bachelorthesis mathematics (WISB 399)

7,5 ECTS

Universiteit Utrecht

Isabel Floor,
6802923,

under guidance of Prof. dr. R.H. Bisseling

January 27, 2022

Contents

1 Introduction

Just over a year ago, vaccines seemed to be *the* answer to *the* question: how to get rid of COVID-19. Hence when the vaccines became available, most people were very willing to get the vaccine. So willing in fact, that different professional groups claimed that they should be one of the first to get a shot. But the Dutch government stood firm and promulgated that the vaccines would be given to people in descending order based on their age. A lot of speculation started about this strategy and whether it would not be better to start vaccinating people in ascending order based on their age, or not based on age at all. Nonetheless the government continued their campaign and now, a year later, 85% of the people that are eligible for the vaccine have gotten at least one shot.
Unfortunately, it turned out vaccines were not the answer to the question. New, more infectious variants of the virus have shaken things up again. We know COVID-19 will not give itself over any time soon, but since we do not want to go back to the lockdown we lived in begin 2020, we must sacrifice a little of our freedom to keep the new infections low and to try to decrease the speed of the virus with which it spreads around. Giving away freedom is hard for some people, so doing research on how to find a livable situation for the population without having to change the measures as rapidly as in the previous year [1] is needed.

In this thesis, I use Dutch demographics to simulate the effect of different measures. To do this, I have used the Agerank simulation that has been used to simulate different vaccination strategies. Before we discuss what I have done, I will give a brief history of this program in section 1.1, and I will discuss some of the mathematics behind it in section 1.2. Finally, in section 1.3, I will introduce my research questions.

1.1 History of Agerank

Early 2021 prof. dr R.H. Bisseling created a program to simulate vaccination strategies during a pandemic. Simultaneously, during his course called scientific computing, students were given the task to create a similar program in MATLAB. The choices some students made in those simulations sparked Bisseling to improve his own program. Though the results were not yet worthy of public announcement. The conclusion was that the government was not making a huge mistake by vaccinating the oldest and weakest people first, but there was still more research to do in this area.

In the second half of 2021 Casper Bakker, a student mathematics, has improved the Agerank program of Bisseling to compare more vaccination strategies in his thesis. To do this, and to attach more information to people in the population, the program has been made object oriented.

All the previous work to the Agerank simulation has been done in Python. From the beginning, Bisselings aim has been to create a program that simulates COVID-19 and vaccination strategies and publishing it to GitHub so that anyone can use and adapt it. The simplified syntax of Python makes Python very readable and useful for such a program [2]. Up to this point, the Agerank simulation has only been used to simulate vaccination strategies and

compare them, but since I will only simulate situations after a large part of the population already has been vaccinated, I will call it the COVID-19 simulation from now on.

1.2 Contact-network-based simulation

The COVID-19 simulation, is a contact-network-based simulation. This means that the behavior of the simulation is based on the contacts between different people. We can represent this as a graph with vertices and edges. There are N vertices in the graph with N the size of the population, where every vertice represents an individual. When two people have contact on a somewhat regular basis, there is an edge between them. The number of edges in the graph is thus based on the number of contacts a person has which depends on their age. Let us look at an example. Below we have a graph with 4 vertices and a number of edges between them. We see that V_1 has contacts V_2, V_3 and V_4.

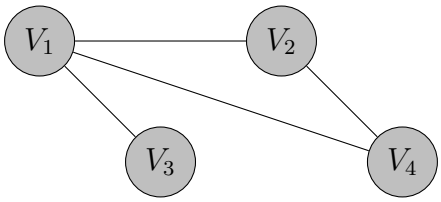


Figure 1: Contact graph of a small population

We can view this graph as a matrix consisting of zeros and ones. For example, the connection between V_1 and V_2 is represented as a one. Between V_2 and V_3 there is no contact, and thus correlates with a zero in the matrix. The graph in figure 1 corresponds to the following matrix.

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

Figure 2: Corresponding matrix to figure 1

We see that every element on the diagonal is a one. Note that you cannot infect yourself so, in theory it is not relevant if this element is a zero or a one. Now imagine that V_2 has been infected by COVID-19 and does not go into quarantine. To calculate who can get infected, we will multiply the matrix in figure 2 with a diagonal matrix where the element on position (x, x) is a one if x is infected. Hence, the element $(2, 2)$ is a one. Then we get the following multiplication.

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Figure 3: Matrix multiplication determining who can get infected

In this case, we can only see ones in the second column, but when more people are infectious, there could be more ones spread over the matrix. To collect this scattered data to see who can get infected, we will multiply the yielded matrix in figure 3 by a vector consisting of ones.

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Figure 4: Multiplication of matrix and vector to gather scattered data

We can now see that V_1 and V_4 could be possibly infected by V_2. In our simulation, whether or not V_1 and V_4 will actually get infected is also based on a number of factors, like whether a person is vaccinated or not, or if there is a measure that prevents them from meeting.

In the COVID-19 simulation, this graph is saved as a sparse matrix. This means that we only save a list of tuples with the ids of the nodes that have an edge between them. In figure 5 we can see a visualisation of how the network looks for a population of 100,000 people. Notice that the matrix is symmetric. When person A can infect person B, it must also be possible for person B to infect A. Another feature that stands out is that we can see different blocks with a certain density. We can see this very clearly for children between the ages 4 and 12, who have a lot of contacts with each other, but less with people out of their age group.

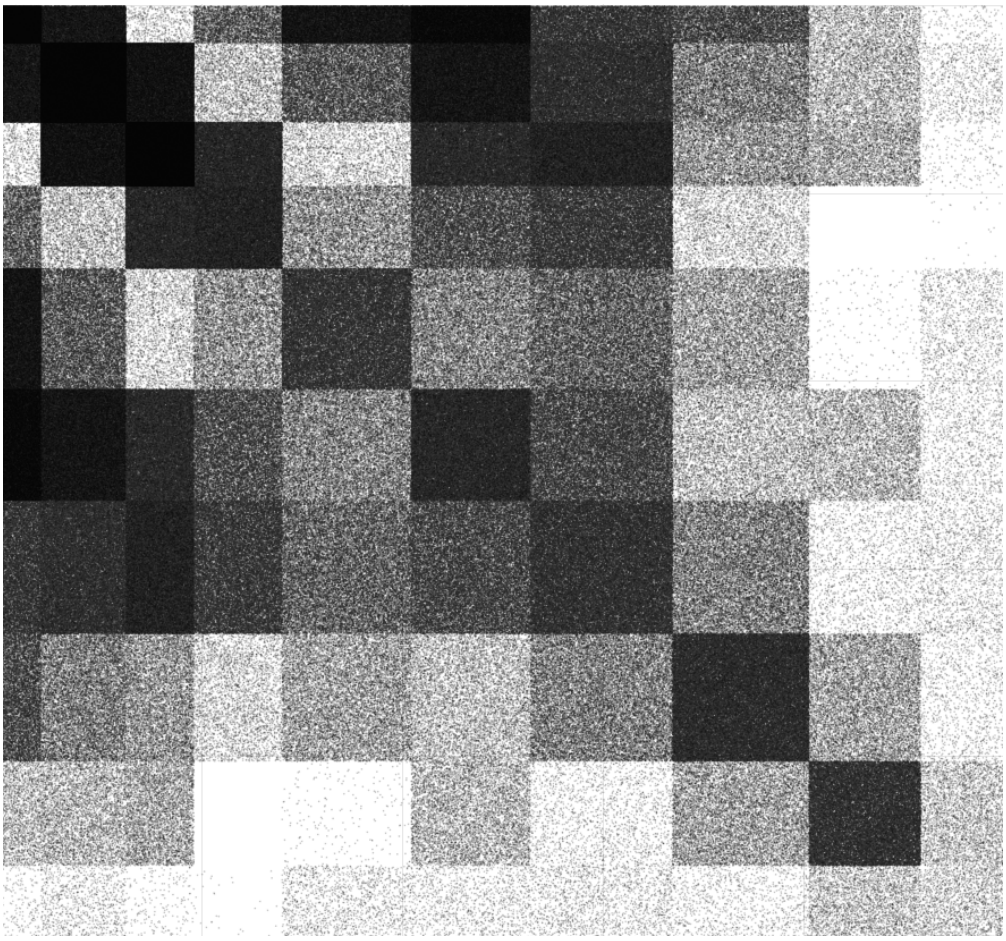


Figure 5: Visualisation of contact matrix for a population of 100,000 people

Other models that can be used to simulate the spread of a disease are a SIR model [3] and an agent based model [4]. The first is based on differential equations and the latter uses intelligent software agents. What makes modeling with a contact network interesting and different from other types of models that can be used to model the spread of a disease, is that there will be some people that are so called hubs. Hubs are people with relatively many contacts. Those people are thus more likely to get infected and simultaneously, they will also be more likely to infect more people than average. We will come back to the subject of contact-based-network simulation in section 3.2.

1.3 Research Questions

In this thesis, I will discuss the changes I have made to the COVID-19 simulation regarding the functionality and the performance and use the results to be able to answer the following questions:

- What is the effect on new infections and hospital admissions when we simulate the 2G measure compared to the effect on new infections and hospital admissions when we simulate the 3G measure?
- What is the effect on new infections and hospital admissions when children between the ages 4 and 12 can get a vaccination voluntarily compared to the effect on new infections and hospital admissions when children are not vaccinated?
- How can we make the COVID-19 simulation fast enough to run a population of 17 million people in a maximum of four hours while keeping Python as the primary programming language?

To answer those questions we will focus on two aspects of the program, the performance and the functionality. I will discuss what changes I have made to the COVID-19 simulation in section 2. Then, in section 3 I will debate what the values of the variables used in the COVID-19 simulation are and present the results that the program yields by using different values. Finally, in section 4, I will discuss what we can conclude from the results and I will talk about future work that can be done on the COVID-19 simulation.

2 Method

Python is a very readable language. Unfortunately, for every advantage there may be a disadvantage and for Python we can see this in its under-performance. To run a population as big as the Netherlands, it is required to improve its performance. In section 2.1, I will discuss how we can increase the performance of the COVID-19 simulation.

Since we have come to a point in reality where most of the Dutch people who have been given the chance have been vaccinated, the relevance of simulating vaccination strategies has lowered. In section 2.2, I will discuss the extended functionality that makes the program more relevant.

In section 2.3, I will discuss the different reasons why refactoring was required and give examples of the changes I have made to the code to solve the problems that were occurring.

2.1 Performance

There are plenty of small changes you can make to your code to improve Pythons performance depending on the functionality of your program. Therefore we will first have a look at the

time complexity in our program to see if we can increase the performance by lowering the time complexity.

In the COVID-19 simulation, the limiting factor on the runtime is function *run_model*. This function consists of repeatedly calling *infectChance* and executing *update* once. The function *infectChance* calculates the chance to get infected with COVID-19 for a single person. This function has a time complexity of $\mathcal{O}(1)$, but we run this function N times in a population of N people. The function *update* has a time complexity of $\mathcal{O}(N)$. Due to the structure of the program and its time complexity, which is already linear in n in approximation, we cannot lower the time complexity further. Because we are aiming for a big performance increase, we are left with two main possible routes to increase Pythons performance.

The first one is by adding parallelism, but since spreading a disease is a cohesive task this will include using concurrency too. Because two programs can run independently I have chosen to implement parallelism by running two or more programs at once with different variables. A second approach is by running different sections of our code in C.

Python is a dynamic language. A simplified definition of a dynamic language, is a language that defers some tasks like type checking to run time. A language like C, which is statically typed, would do this at compile time [2]. Postponing type checking to run time yields a lower development cost and a more flexible program during run time, but it causes dynamic languages like Python to be inherently slower than static languages like C.

By running sections of the code in C, we enforce the types being checked during compile-time and thus will the process of changing the variables during run time be quicker. This can be done in multiple ways, but there is a fairly new software tool that has a great reputation for speeding up numerical programs. This software tool is called Cython. Cython extends the Python language and is also called a superset of Python [5]. The added syntax makes it possible to compile statically and thereby speed-up the program significantly.

Cython has all the benefits of Python and a dash of the speed of C. There are a couple of terms to meet when using Cython and running functions in C. The biggest challenge is that you can not define your own types in Python and then run them in C. You have to define them in C and import them into your Python code.

Cython has a good reputation when it comes to scientific computing [6], but to see if Cython would make a noticeable difference to the COVID-19 simulation, I wrote a short piece of code mimicking what happens during the simulation. It loops a number of times through a list of objects of the type person, changing some of their variables. To have access to the type person in Cython, we have to implement it as a struct in C. For this test I have chosen to give the type person three variables. They contain the ID of a person, the number of days that have passed since they got infected and the number of weeks that have passed since they have been vaccinated. This resulted in the following piece of code.

```
cdef struct person:
    int personID, daysInfected, weeksVaccinated
```

To compare, I have implementated a similair class in Python. This looks like the following piece of code.

```
class person(object):
    def __init__(self, personID, daysInfected, weeksVaccinated)
        self.personID = personID
        self.daysInfected = daysInfected
        self.weeksVaccinated = weeksVaccinated
```

For both Cython and Python we create a list of 100,000 objects of the type person and use that as input for the function. I have implemented a Cython and a Python version of the following pseudo-code.

```
FOR each person in the list of people
    IF daysSinceInfection > 0 THEN
        INCREMENT daysSinceInfection
    INCREMENT weeksSinceVaccination
ENDFOR
```

To test the effect of using Cython, I first execute this function five hundred times using Cython and use the time module of Python to measure how long it takes to finish. Then I execute this function again five hundred times, but just with Python and again measure the time how long this takes. The input in both cases, is the same list of 100,000 people, but the first time the objects in the list are implemented in C and the second time the objects in the list are implemented in Python. When running this test I got the following result.

```
—— using Cython it takes 5.58 seconds ——
—— using Python it takes 10.3 seconds ——
```

Only in this short piece of code Cython increases the speed almost two times. Because we keep changing a lot of variables during run time in the COVID-19 simulation, I expect using Cython will increase the performance of the code significantly and thus I have been convinced to implement this into the COVID-19 simulation.

2.2 Functionality

To make the COVID-19 simulation more relevant for the current situation in the Netherlands, I had to change some of the functionalities and add functionality. This includes adding preprocessing to give the population a head start, extending the households and adding booster vaccines. I will discuss these and more in this section.

Preprocessing
Since this simulation focuses on the measures instead of the vaccinations, I use preprocessing to reconstruct the weeks prior to the point where I start my simulation. I have chosen to take week 45 of 2021 as imaginary starting point since the infections in the Netherlands were stable in the two weeks prior to week 45.

I reconstructed the vaccinations that have been given in the first 44 weeks of 2021. To do this, I have used information on how many people have been vaccinated per week per age [7]. This results in 70% of the population being vaccinated at the start of the simulation.

To give the infections a head start, I have reconstructed the number of infections that took place in week 43 and 44 in the Netherlands. For this I have used information about the number of infections per day per age [8].

Since the information about vaccination readiness of the prior months is irrelevant, I have used information about the vaccination readiness in week 42 [9].

Extension of households
In the previous version of the COVID-19 simulation, the following kinds of households were implemented:

- Couples without children
- Couples with 1, 2, 3 or 4 children
- Single parents with 1, 2, 3 or 4 children

I have added the following types of households.

- Student houses
- Care institutions

To decide the number of people in the population that should be living in a student house and in a care institution, I have used the following variables. Of the population of the Netherlands, 2.18% are people living in a student house with an average of seven people per student house [10] and 0.67% are people living in a care institution [11] with an average of 48 people based on the 2363 care institutions in the Netherlands [12]. Another 17.7% of the population is living alone [13]. This resulted in still having a number of people in my population who were not assigned a household. Therefore I have divided the number of people that should not be living at home into households containing two to four people. To speed up the process of checking if someone in the household is infectious, I have added the variable *infected* to the object household. This variable holds the number of people in the household that are infectious. In addition, this number only holds the number of people that are not in quarantine and not in the hospital.

Addition of close contact groups
During some of the measures, children and students do not have to stay distant from each other in school classes. When they are not walking around, it is not mandatory to wear a face mask. Therefore, a lot of infections take place at schools. To be able to simulate this in the simulation, I have divided all children up to a certain age into groups to mimic those school classes.

To create the school classes, I have divided all children between the ages of 4 and 18 in groups with an average of 23 children. This number is based on the number of children in a class in primary school in 2018 in the Netherlands [14]. For high school, the number of

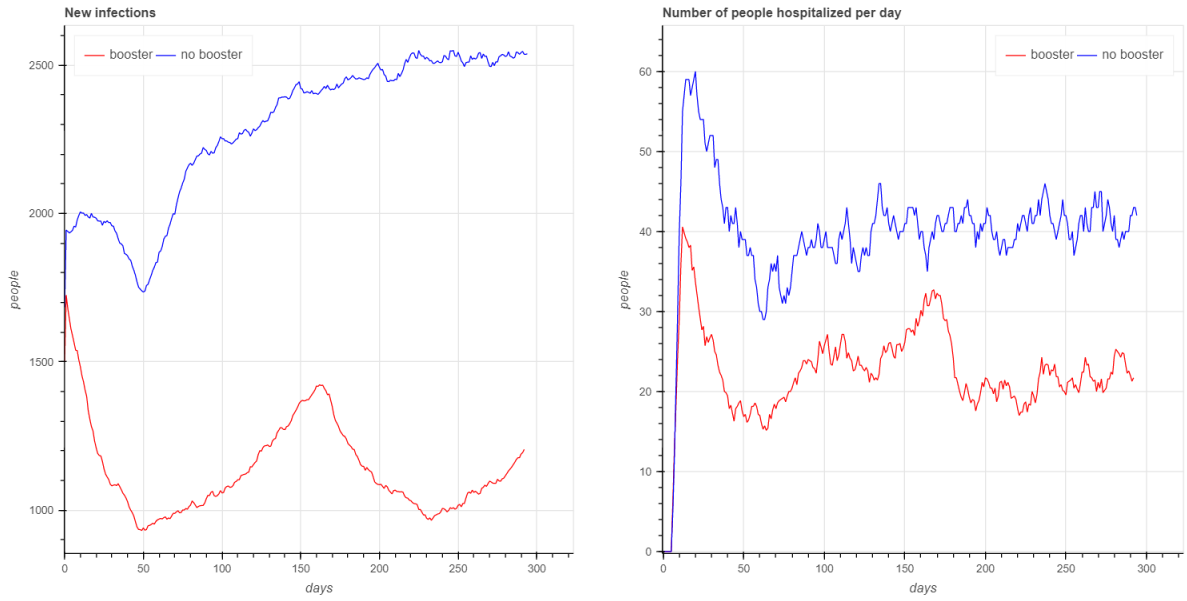
children in a class differs per educational institution. The averages are at least 13 and at most 26, but I have chosen to still use the average of 23 for these groups too.

Addition of booster vaccines

In addition to vaccinating people that are unvaccinated, I have added a function that gives a number of people a booster vaccine. The booster vaccines are distributed based on age, descending. People who have not been vaccinated or who have been vaccinated within six months prior, cannot get a booster vaccine.

Currently, roughly 250,000 people get a booster vaccine in the Netherlands each day [15]. This means that for a population of 1 million, almost 15,000 people can be vaccinated each day. To show the effect of distributing booster vaccines, I have compared the following two situations. (1) A situation in which we do not give boosters. People only get two vaccines, and (2) a situation where we keep giving people a booster every six months. To compare these situations, there are a couple of requirements. First of all, the vaccines do not become more effective when given more shots. I have made the assumption that the boosters will not be more effective than the vaccines. Little information on this subject can be found. Below you can see the results that were yielded.

The simulations were done on a population with a size of one million people. In figure 6 we can see the total number of new infections per day. We see that the number of new infections keeps increasing over time. The number of hospital admissions is more stable, but a lot higher when we do not booster, than when we do booster.



(a) (b)

Figure 6: (a) Number of new infections in two different situations, (b) number of new hospital admissions in two different situations

Child vaccination
To be able to vaccinate the children with ages between 4 and 12, I have added a function tha vaccinates a number of children randomly based on the number of vaccines that can be given on a day.

People who do not want to be vaccinated
Assuming that people who do not want a vaccine tend to live together, I have implemented that people living with someone who does not want a vaccine have a higher chance of not wanting a vaccine.

I expect that those extensions on the program will increase the accuracy of the program and are sufficient to be able to simulate the different situations needed to be able to answer the research questions.

2.3 Refactoring

Being the third person working on the COVID-19 simulation, a lot of implementation choices have been made in the past. Those were good enough for the purpose of the project at that point, but now that the purpose has shifted, some of the choices made in the past are not sufficient for my thesis. I have made the following changes to the code to solve the problems that occurred.

The first problem was that some methods were long and did lots of different things. For example, there were only three methods in charge of making all households. I was not able to add student houses without duplicating code and making the methods even longer. Therefore I split these methods up into parts giving every method one task. I ended up with eight methods closely working together to make all the different types of households. Secondly, the simulation originally could not handle a population of more than one million. This was caused by long lists being passed around between functions. By adding all those methods to a class and declaring the variables used in them at the top of the class, all methods were able to access them.

3 Results

In this section, I will present the results that were obtained by running the COVID-19 simulation with different variables. I will enlighten what choices I made as to the variables in section 3.1. In section 3.2 and 3.3, I will go into further detail about how we can compare different measures and present the results that were obtained by running the COVID-19 simulation. Lastly, in section 3.4, I will explain how I adapted the code to work with Cython.

3.1 Choice of variables

Over time, some of the values of the variables used in the original COVID-19 simulation have deprecated. Therefore, I have found new values which I will discuss in this section. Choosing

good, realistic variables for the COVID-19 simulation is harder than it looks. When the virus is too contageous in our parameters, too many people will be infected in the beginning before it has a chance to spread around. Making the virus not contageous enough results in barely any infections.

Another problem is that in comparison, the infections in the Netherlands are pretty low. When we have 6,000 new infections in the Netherlands, we would only have 35 new infections in a population of 100,000 people. This makes it hard to see what kind of difference it makes to use a certain measure, because how do we know that the difference in infections is actually caused by the measure instead of by the randomness of the simulation? One solution to this problem would be to run multiple simulations and take their average, but this can result in unrealistic numbers when the peaks in infections alter each other. Another solution to this is to run the simulation with a bigger population. Therefore I have chosen to use a population size of one million to answer the first and second research question.

Protection after infection
Research has shown that people who have recently gone through an infection are less likely to get infected again [16]. The longer it has been since someone has been infected, the less protected a person is. I have chosen to not implement this decrease in infection since it is difficult to implement. Instead, I have implemented that a person can not get infected again within 4 weeks after they have been infected.

Protection of vaccine
Moderna and Pfizer-BioNTech are the most commonly vaccines used in the Netherlands [17]. Since they are very similar in effectiveness [18], I have decided to not differentiate between these two vaccines. Every person in my population is given the Pfizer-BioNTech vaccine.
An American study monitored 2.7% of the U.S. population for the period February 1, 2021 to October 1, 2021 to see the vaccination effectiveness changes over time [18]. It found that when comparing people vaccinated with Pfizer-BioNtech versus unvaccinated people, the adjusted hazard ratio was 0.13 in March. This means that when an unvaccinated person has a 100% chance to get infected, that same person would have had a 13% chance to get infected if they would have been vaccinated with Pfizer-BioNTech. This study also shows that the decline of the vaccination effectiveness is visible from the first month of having been vaccinated when the hazard ratio increases to 0.17. The hazard of getting infected by COVID-19 has increased from 0.13 to 0.27 in three months and another three months later it has increased to 0.57. I have therefore chosen to handle a doubling time of 13 weeks and calculated the growth factor using the vaccination effectiveness at $t = 0$ and at $t = 13$ where t is the number of weeks that has passed since a person has gotten the vaccination. This resulted in the following equation,

$$M = \min(0.13 \cdot 1.06^t, 1).$$

Note that the chance to be infected with COVID-19 cannot be larger than 1. We will only calculate this function for $t \in \{1 \dots 45 + s\}$ where s is the number of weeks we simulate and 45 the number of weeks we reconstruct during the preprocessing. Since we need to calculate it for every person for 300 days, it is quite a time consuming function. Therefore I have chosen to calculate M for $t \in \{1 \dots 43 + 45\}$ during preprocessing and store it in a list so we can access it in $\mathcal{O}(1)$ time during the simulation.

Chance to go into quarantine
In surveys that have been taken during this pandemic people have been asked if they follow the measures given by the government. Only 65% indicate to stay at home when they have symptoms that could be caused by COVID-19 [19]. We have to take into account that not all those people have COVID-19, but the chance of someone having COVID-19 is still present. For example, in the week of 10 January 2022, an average of 35.1% tests confirmed to be positive [20], which means about 12% of the people who do not go into quarantine have COVID-19. Another reason why people would not go into quarantine is because they do not experience any symptoms, but are still contagious. Therefore I have compared different values and decided that the best value for this variable is 0.9.

Chance to get infected by a random person
When we are considering a lockdown situation, you will only see your close relatives, but when there is no lockdown, people travel by public transport, go out to eat in a restaurant or go to the theatre among other things. In all those places they can get infected by someone they do not know or by someone they do not regularly see. Since it is possible to get infected by someone who is not in one of your contacts in the contact matrix, a random element must be added in infecting people. To do so, I have created a method that randomly infects a number of people proportional to the number of people that have been infected that day. When there are s people infected on day x, we will randomly infect $0.01 \cdot s$ extra people on day x.

Vaccination readiness
Another variable that I have changed, is the vaccination readiness. In the previous versions of this project, the vaccination readiness was 85% regardless of age. I have changed this to an updated value based on age [21]. The percentages used in the simulation are presented in table 1.

Age	16 - 24	25 - 39	40 - 54	55 - 69	70+
Vaccination readiness	91.6%	90.6%	92.9%	96.9%	98.7%

Table 1: Vaccination readiness based on age

3.2 Comparing 2G, 3G and a lockdown

To answer the first research question, we have to sort out what the difference in encounters is for people when using a 2G or a 3G strategy.

During the summer of 2021, the Dutch government introduced a QR code for different places like restaurants and theaters. For a person to be allowed access, they needed a QR code. To get a QR code, someone had to be vaccinated for at least 2 weeks, someone should have recovered from COVID-19 within six months prior, or someone should test negative to gain a QR for 24 hours. They called this a 3G measure.

This change caused people to have more contact moments with different people than before. With this change from lockdown, a person could get infected by someone without knowing

them. This change is rather hard to implement because it is random, but somewhat predictable. In the COVID-19 simulation we infect a number of people randomly every day based on the number of infections of that day with the function *infectRandom* as mentioned in section 3.1.

Besides the 3G measure, a 2G measure was discussed which means that a person can only retrieve a QR code by being vaccinated or having recovered from an infection within six months. In essence, the difference between 2G and 3G is that the contact moments a completely unvaccinated person has is drastically lowered. When sticking to the rule, such a person would only see their cohabitants and classmates. Note that the 2G and 3G measures are not applicable to children under the age of 12.

To compare the 2G and 3G measure, I have simulated two different situations. In the first situation people who have not been vaccinated and are older than 12, have been removed from the contact matrix. Since they and their edges are removed, they cannot be infected anymore and they cannot infect others in the contact matrix. The only people they still can be infected by or infect, are their cohabitants and possibly their classmates. For the second situation we do not have any extra measures. Both simulations are done on a population of one million people 300 days and are plotted in figure 7.

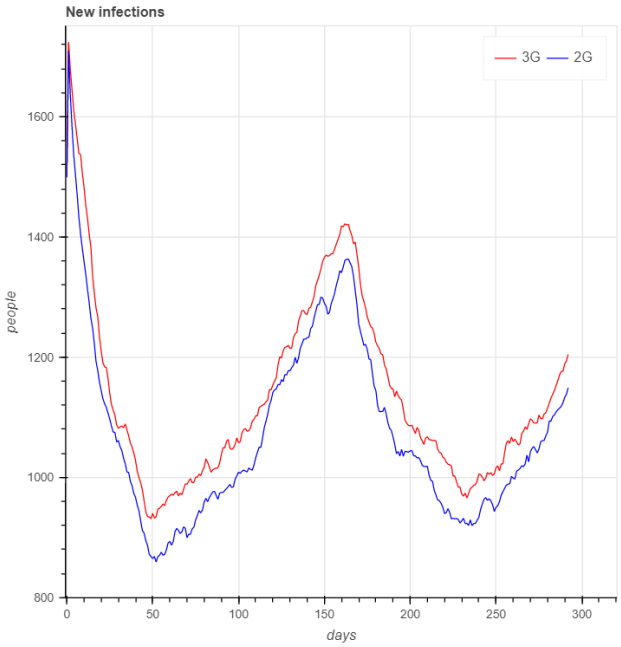
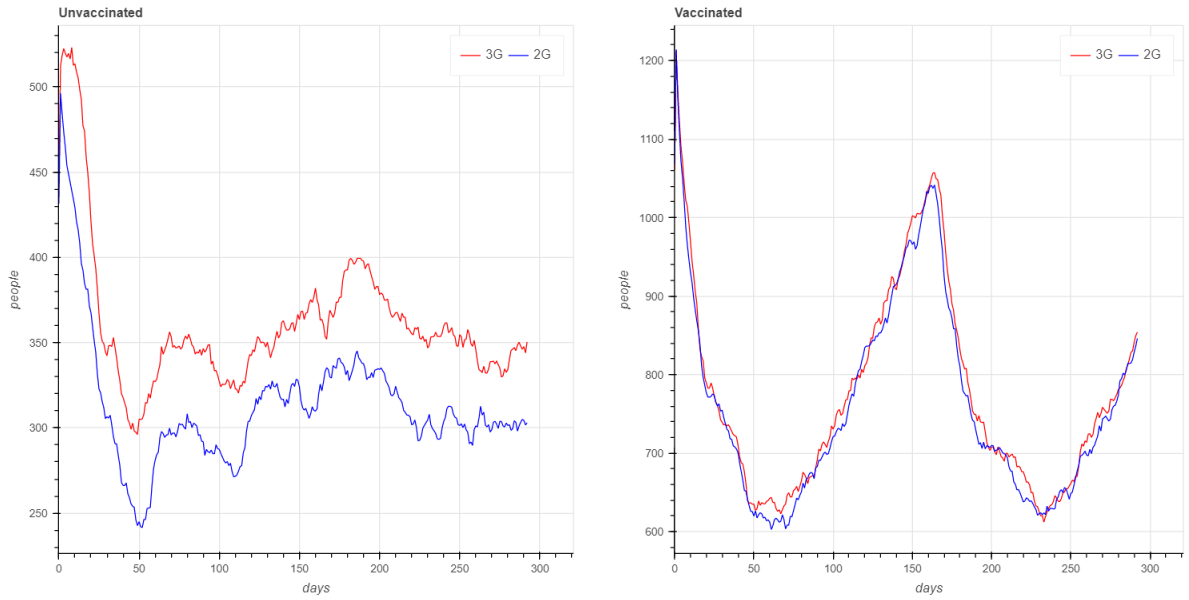


Figure 7: Total number of new infections per day

First, note that the y-axis starts around 800. Secondly, notice that there is a wave in new infections around day 170. This is caused by the vaccination effectiveness. Around the six month mark, the hazard of still getting infected has increased to almost 60% in combination with the choice to let people receive a booster only if they have been vaccinated more than 24 weeks in advance, this results in this wave. We can see that there are less new infections every day, but we see that the shape of the graphs are the same. Now when we look at the

new infections divided into two groups in figure 8, we can see even more clearly what the effect is of a 2G measure.



(a) (b)

Figure 8: (a) Number of new infections when only considering vaccinated people, (b) number of new infections when only considering unvaccinated people

We can see that the 2G measure does not make much of a difference in number of infections of vaccinated people, but we do see a difference in infections of unvaccinated people. Note that all children under 12 are included in the unvaccinated group which are not restricted from going into public areas. Hence a lot of unvaccinated people will still get infected.

We see that there is a difference between the number of infections when comparing the 2G and the 3G measure. To put the difference in perspective, below I have included a graph in which the number of new infections per day is given when there would be a lockdown. In this case, the lockdown measure has caused all people in the simulation to have close contact with 10 times less people. Note that schools do not close, so children will still have contact moments.

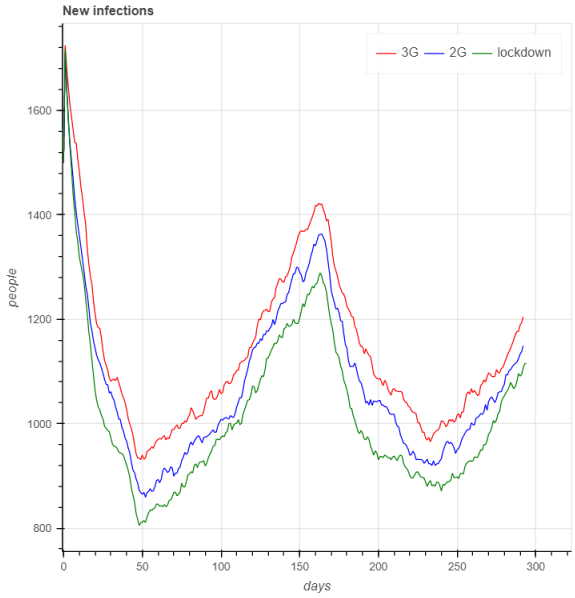


Figure 9: Total number of new infections per day in three different situations

In this figure we can see that the number of infections per day in lockdown is slightly lower than in a situation where there is a 2G measure. At the same time, all graphs have the same shape. We can see that all the measures simulated and shown above do not result in a long term solution. Eventually the effectiveness of the vaccination decreases and more people will get infected again.

As the research also examines hospital admissions, I have plotted those in the graph in figure 10.

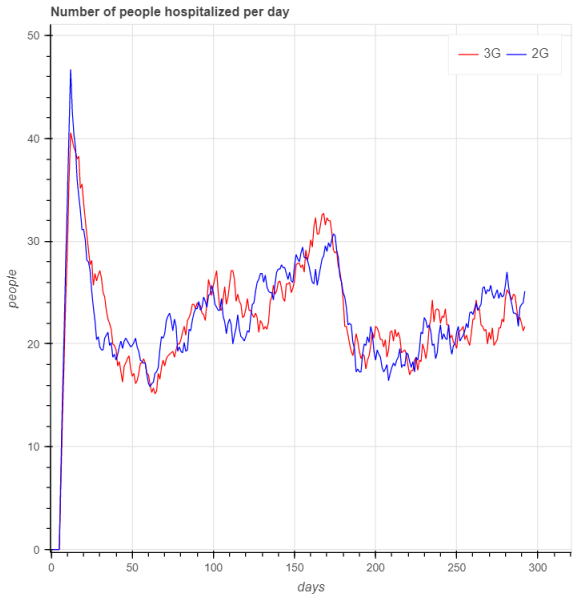


Figure 10: Number of hospital admissions per day in two different situations

We can see that there is no difference in the number of hospital admissions visible. The reason for this, is that the oldest people have the highest chance of being admitted to the hospital. At the same time, 90% of people with an age over 80 have been fully vaccinated. In comparison, 67% children with ages between 12 and 17 have been fully vaccinated, but they have a much lower chance to be admitted to a hospital [15].

Back to the hubs mentioned in section 1.2. Recall that hubs are people who have relatively many contacts. When we remove hubs from a network, this can result in a population with multiple disjoint portions. This makes it impossible for the disease to jump from one portion of the population to another, making it easier to contain the virus. Let us have a look at the graph of a small population in figure 11.

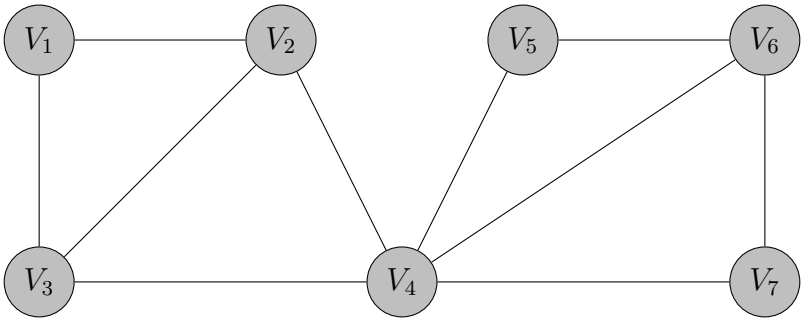


Figure 11: Contact graph of a small population

Consider the situation that V_7 has been infected with COVID-19 without them knowing and is now walking around infectious. Unknowingly they infected V_4 and now, since V_4 has many connections, the disease easily spreads all over the population. In the graph above, we can see that there is a clear hub, V_4. Imagine that V_4 would go in quarantine and its edges would be removed from the graph. We yield the graph shown in figure 12.

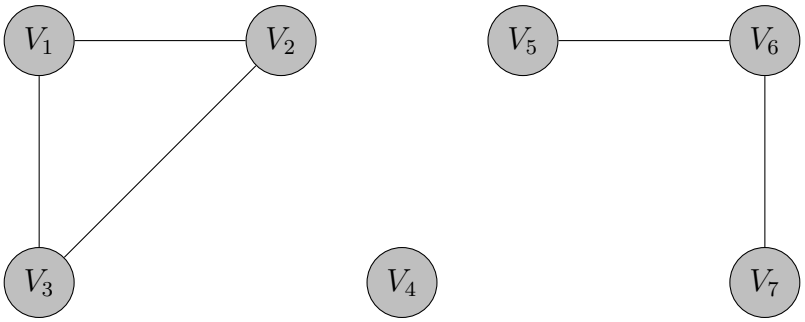


Figure 12: Contact graph of a small population

We see that the population is split into three disjoint sections. Now when V_7 gets infected, the disease can only spread around in one section of the population. Here V_1, V_2, V_3 and V_4 have no risk in getting infected. With only one person in quarantine, we have prevented multiple infections in this small population.

With the 2G measure, we are doing something similar: we have some people removed from the contact matrix hoping this will prevent infections. The problem is however that we are removing unvaccinated people, but since only about 15% of the population is unvaccinated, they are probably not the hubs. Maybe they already barely have any contacts. This causes the 2G measure to not result in the decrease in infections that is desired to contain a virus.

3.3 Vaccinating children

To answer the research question about vaccinating children, we will simulate what happens when we vaccinate children between the ages of 4 and 12 randomly. In the Netherlands, it is possible to vaccinate roughly 50,000 people per day [15]. So for a population of one million, almost 3,000 people can be vaccinated each day.
To make a good comparison we keep all the other variables the same. In both situations there is a 3G measure as described in section 3.2. The yielded result when looking at the number of new infections can be found in figure 13.

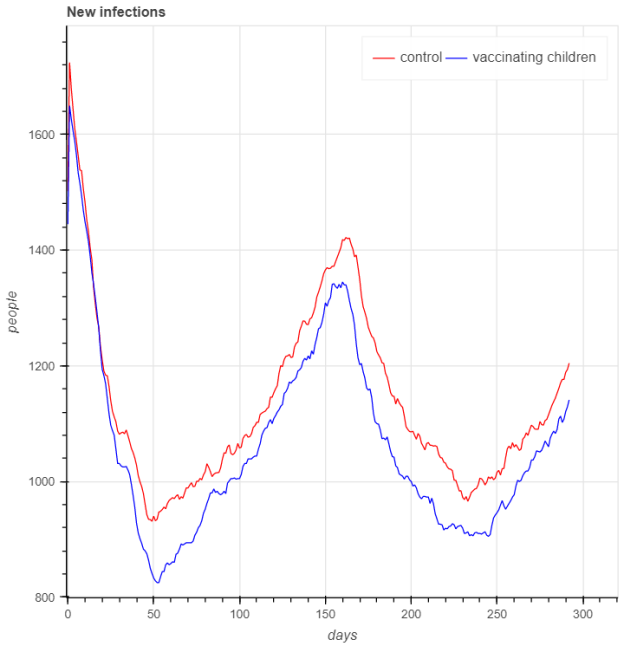


Figure 13: Total number of new infections per day

Similar to the 2G measure, vaccinating children with ages between 4 and 12 results in less new infections, but the graph still has the same shape.

Recently, IO Research has published an article about the vaccination readiness for children [22]. Only 35% of the parents will definitely or will probably let their child get a vaccination. Therefore, I have also simulated a situation in which we stop vaccinating children when we have vaccinated roughly 35%. The results that were obtained can be found in the following figure.

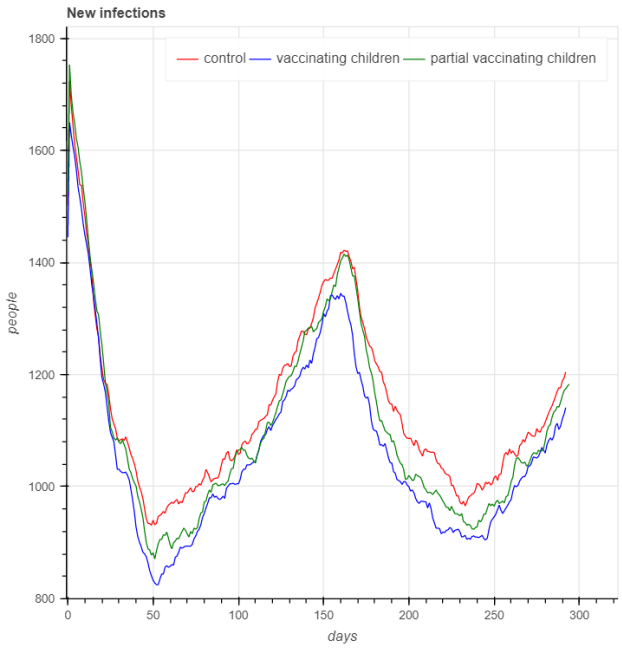


Figure 14: Total number of new infections per day

In figure 14 we see that there would be a lot less infections if all children are vaccinated, but vaccinating about 35% of all children under the age of 12 shows less infections too.

As the research question also examines hospital admissions, I have plotted those in the graph below. Note that I have compared a control situation and a situation in which we vaccinate all children. In both situations there is 3G measure.

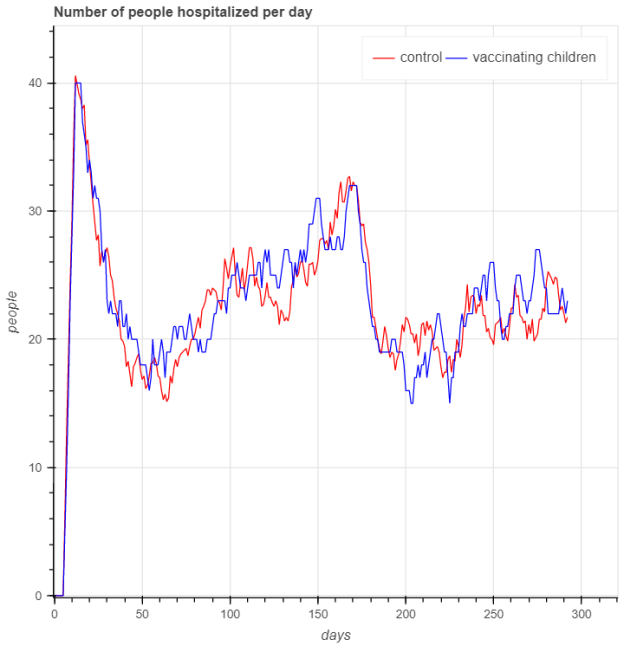
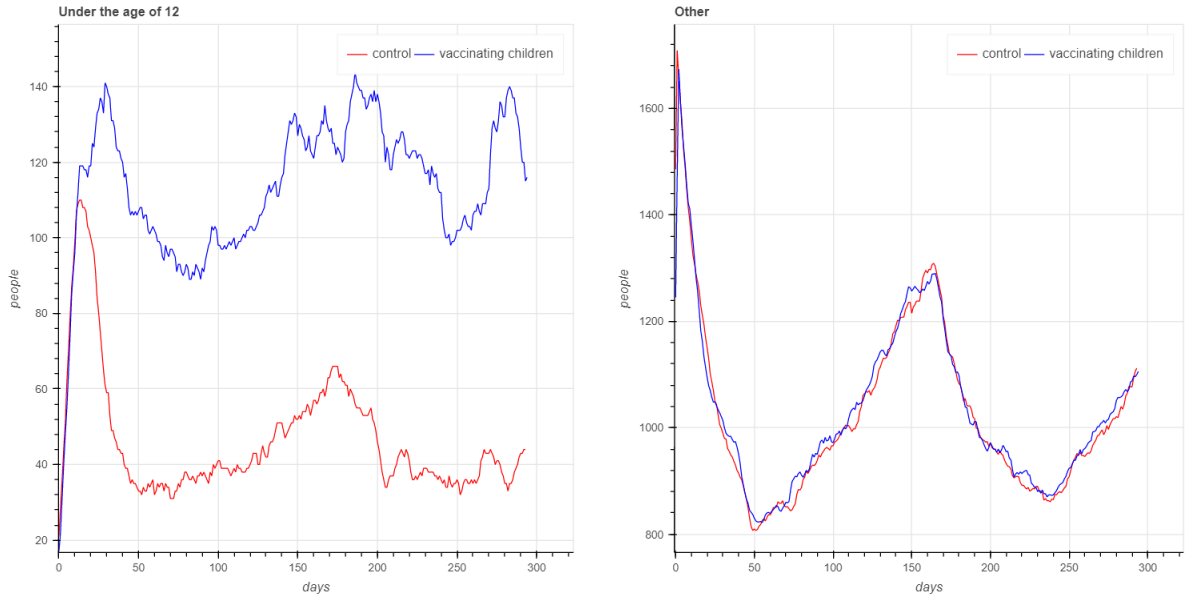


Figure 15: Total number of new infections per day

Looking at the number of hospital admissions in figure 15, we see that it makes no difference if we vaccinate children between the ages of 4 and 12. We can explain this by dividing the number of new infections into two groups, a group with children under 12, and the other people.



(a) (b)

Figure 16: (a) Number of new infections when only considering children under the age of 12, (b) number of new infections when only considering people over the age of 11

We see that when we vaccinate children, a lot less children are being infected by COVID-19, but it does not make a difference for people over the age of 11. The number of infections under those people does not decrease. This explains why the number of hospital admissions does not decrease when we vaccinate children between the age of 4 and 12. Remember that we have chosen to keep the schools open. Therefore the results could be different from a situation where schools are closed.

3.4 Cython

To optimize the run time of the COVID-19 simulation with Cython, I have created two custom types in C, *person* and *changes*. The type *person* consists of all the information we save for a single person in the population. The type *changes* consists of the variables total infected, currently infected, vaccinated and recovered. To optimize the run time, I have added typing information to the functions that are called when running the model. One of those functions is the function *update*. In this function we loop through all the people in the population and update their status in case they have been infected by the disease, we change one or more of their variables. We also change the variables of the tracker based on if the person goes into a new phase of their illness. Updating variables is something that takes

a lot of time in Python since there is no typing information saved during compilation time. Therefore, the run time has already drastically decreased after adding typing information to the variables in this method.

Other methods that are frequently used are methods related to calculating the chance to get infected. Adding typing information to the variables in these methods has also resulted in visible performance improvement. An example of what difference typing information can make by using Cython is shown below. This method calculates the effectiveness of the vaccine a person has gotten. This is what it looks like in Python.

```
def prevVaccinated(self, time, person):
    if person.vaccinated == 1 :
    number = time//7 + 46 - person["weekOfVaccination"]
        return vaccEffectiveness[number]
    else :
        return 1
```

We see that we first lookup in what week the person has been vaccinated. Then we calculate how many weeks have passed and what the adjusted hazard rate is. After giving typing information to the variables using Cython, the function looks as follows.

```
cdef float prevVaccinated(self, int time, person person):
    cdef int number
    if person.vaccinated == 1 :
    number = time//7 + 46 - person.weekOfVaccination
        return vaccEffectiveness[number]
    else :
        return 1
```

There is only a small difference in the syntax making the code still easily readable, but when using the syntax from Cython in all the methods in the class, we yield a noticeable increase in performance.

To answer the third research question on how to increase the performance of the program, I have added typing information to all functions relevant for increasing the performance. Some of the functions to which I have added typing information can be found in the appendix.

Afterwards, I have compared the time that it took for the program to finish a couple of different runs. When simulating a population of 100,000 for 300 days, it takes approximately 220 seconds to finish when using Python. Using Cython it only takes approximately 120 seconds. When we simulate a larger population, we yield an ever bigger performance increase. When simulating a population of one million for 300 days, it takes approximately 60 minutes to finish when using Python while it only takes 30 minutes to finish when using Cython.

We see that with little syntax changes, we are able to increase the performance quite a bit using Cython. Nonetheless, we are still not able to simulate a population of 17 million for 300 days within 4 hours. We will discuss what changes could be made to the code to reach this goal in section 4.1.

4 Conclusion

Now that we have seen the results from the different simulations, we can answer the research questions. The first research question was:

> *What is the effect on new infections and hospital admissions when we simulate the 2G measure compared to the effect on new infections and hospital admissions when we simulate the 3G measure?*

We have seen that the 2G measure does result in a lower infection rate, but it is not a long term solution to the repeated increase in infections when the vaccination effectiveness decreases. We also saw that a lockdown where everybody has 90% less contacts yields a very similar result. When looking at the hospital admissions, we have seen that the 2G measure does not make a noticeable difference in the number of hospital admissions compared to the 3G measure. Our second research question was:

> *What is the effect on new infections and hospital admissions when children between the age of 4 and 12 can get a vaccination voluntarily compared to the effect on new infections and hospital admissions when children are not vaccinated?*

Just like the 2G measure, we saw that vaccinating children results in less new infections, but it still does not allow us to get the virus under control. What we can see is that vaccinating children, which are 10% of the population, is just as effective as putting 15% of the population in lockdown. When we only vaccinate 35% of the children, we see that the number of infections is lower than when we would not vaccinate children, but higher than when we would vaccinate all children. This measure does not result in a difference in hospital admissions since vaccinating children does not have an effect on the number of infections under adults. Our last research question was:

> *How can we make the COVID-19 simulation fast enough to run a population of 17 million people in a maximum of four hours while keeping Python as the primary programming language?*

We have discussed Cython and its advantages extensively in section 3.3. We have seen that we can simulate a population of 100,000 for 300 days in 2 minutes and that we can simulate a population of one million in 30 minutes. We should thus be able to simulate a population of 17 million for about 130 days within 4 hours. To be able to run the simulation for 300 days within the 4 hours, we require the addition of other optimisation techniques than the ones used in this thesis.

4.1 Future work

Plenty of future work can be done. One of the main things that would improve the credibility of this project is software testing. Right now there is no automatic software testing, by e.g.

unit tests, but the project is an open source project published on GitHub, therefore anyone can check the results.

Other subjects that need future work is the option of protection for infection due to an earlier infection. Right now people are protected for four weeks and then there is a hard cut off. In reality, we have learned that earlier infections give some protection, but like the vaccines this decreases over time.

In this project we also do not consider new variants. It would be interesting to see what the upswing of new variants would mean for the number of infections. What makes it difficult, is that new variants and protection from earlier infection are two related things. New variants are not predictable making it difficult to estimate how infectious they will be or how sick they will make a person.

Another aspect that could change the results yielded by this project, is the addition of more freedom for the people in the population. Right now there is not much randomness in which contacts people will have while in reality people go to theatres, restaurants and travel with public transport. Further work can be done trying to explore how this can be integrated into the program.

Lastly, the population is now closed from the rest of the 'world'. We do not consider the case where someone temporarily leaves the population and comes back again possibly being infectious.

References

[1] Number of confirmed cases over time. https://coronadashboard.rijksoverheid.nl/landelijk/positief-geteste-mensen. Accessed: 12-12-2021.

[2] Guido van Rossum. *An Introduction to Python*. Network Theory Limited, 2003.

[3] Ronald Ross. An application of the theory of probabilities to the study of a priori pathometry.—part i. *Proceedings of the Royal Society of London. Series A, Containing papers of a mathematical and physical character*, 92(638):204–230, 1916.

[4] Charles M Macal and Michael J North. Agent-based modeling and simulation. In *Proceedings of the 2009 Winter Simulation Conference (WSC)*, pages 86–98. IEEE, 2009.

[5] Cython c-extensions for python. cython.org. Accessed: 12-12-2021.

[6] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2):31–39, 2010.

[7] Cumulatieve vaccinatiegraad voor volledige covid-19-vaccinatie naar geboortejaar en week. https://www.rivm.nl/covid-19-vaccinatie/cijfers-vaccinatieprogramma. Accessed: 15-12-2021.

[8] Explanation of data. https://coronadashboard.rijksoverheid.nl/verantwoordingpercentage-positief-geteste-personen-via-ggd-teststraten. Accessed: 15-12-2021.

[9] Verandering vaccinatiebereidheid naar leeftijd. https://www.rivm.nl/gedragsonderzoek/maatregelen-welbevinden/vaccinatie. Accessed: 15-12-2021.

[10] Landelijke monitor studentenhuisvesting 2021. 2021.

[11] aantal bewonders van verzorgings en verpleeghuizen 2019. https://www.cbs.nl/nl-nl/maatwerk/2020/13/aantal-bewoners-van-verzorgings-en-verpleeghuizen-2019. Accessed: 10 - 12 - 2021.

[12] Number of care institutions. https://www.zorgkaartnederland.nl/verpleeghuis-en-verzorgingshuis. Accessed: 08-01-2022.

[13] Huishoudens, samenstelling, grootte. https://opendata.cbs.nl/statline//CBS/nl/dataset/71486NED/t Accessed: 08-01-2022.

[14] Hoe is de groep van mijn kind op de basisschool samengesteld? https://www.rijksoverheid.nl/onderwerpen/basisonderwijs/vraag-en-antwoord/hoe-zijn-de-groepen-in-het-basisonderwijs-bo-samengesteld. Accessed: 18-01-2022.

[15] Vaccinations over time. https://coronadashboard.rijksoverheid.nl/landelijk/vaccinaties. Accessed: 04-01-2022.

[16] Malik Peiris and Gabriel M Leung. What can we expect from first-generation covid-19 vaccines? *Lancet (London, England)*, 396(10261):1467, 2020.

[17] Welk vaccin wordt voor het boosteren gebruikt. https://www.rivm.nl/covid-19-vaccinatie/vragen-achtergronden/boostervaccinatie. Accessed: 08-01-2022.

[18] Caitlin c. Murphy Nickilou Y. Krigbaum Arthur W. Wallace Barbara A. Cohn, Piera M. Cirillo. Sars-cov-2 vaccine protection and deaths among united states veterans during 2021. 2021. Accessed: 15-12-2021.

[19] Compliance and support over time. https://coronadashboard.rijksoverheid.nl/landelijk/gedrag. Accessed: 13-01-2022.

[20] Average percentage of confirmed cases over the last 7 days. https://coronadashboard.rijksoverheid.nl/landelijk/positief-geteste-mensen. Accessed: 17-01-2022.

[21] Verandering vaccinatiebereidheid naar leeftijd. https://www.rivm.nl/gedragsonderzoek/maatregelen-welbevinden/vaccinatie. Accessed: 08-01-2022.

[22] Vaccination readiness parents children 5 - 11 stays low. https://www.ioresearch.nl/actueel/vaccinatiebereidheid-ouders-5-11-jarigen-blijft-laag/. Accessed: 18-01-2022.

5 Appendix

Below I have included the class simulation. This class is responsible for running the simulation.

```
import random as rd
from initialize import *
from preprocessing import *
from vaccination import *
from customTypes cimport person, changes
import numpy as np



cdef class simulation:
    def __cinit__(self, vaccEffectiveness,
                  meeting,
                  network,
                  housedict,
                  othergroups,
                  people):
        self.vaccEffectiveness = vaccEffectiveness
        self.meeting = meeting
        self.network = network
        self.housedict = housedict
        self.othergroups = othergroups
        self.people = people
        self.startage = N - 1
        self.sumt = 0
    cdef list vaccEffectiveness, meeting, network, othergroups
    cdef public list people, housedict
    cdef person person
    cdef int id, total, startage
    cdef float p, r, chance
    cdef int x[1000000], y[1000000]
    cdef int sumt

    cdef infect_standard(self, int time):
        """This function performs one time step (day) of the
        infections a is the n by n adjacency matrix of the network
        status represents the health status of the persons.
        In this step, infectious persons infect their susceptible
        contacts with a certain probability."""
        cdef int total = 0, id
        cdef person person
```

```
        # determine list of infectious persons from status
        for per in self.people:
            person = per
            id = person.person_id
            if person.infectious == 1:
                self.x[id] = 1
            else :
                self.x[id] = 0
            if twoG and person.vaccinated == 0 and person.age > 12:
                self.x[id] = 0
            self.y[id] = 0

        # propagate the infections
        for edge in self.network:
            (i, j) = edge
            self.y[i] += self.x[j]

        for per in self.people:
            person = per
            id = person.person_id
            # incorporate the daily probability of meeting a contact
            # taking into account the possibility of being infected twice

            if twoG :
                if person.vaccinated == 0 and person.age > 12:
                    continue
            #print(self.y[self.person.person_id])
            if self.y[id] > 0:
               if person.susceptible == 1 and \
                  rd.uniform(0,1) < 1 - (1 - P_MEETING) ** self.y[id] and \
                  rd.uniform(0,1) < self.prevVaccinated(time, person) :
                      person.daysSinceInfection = 1
                      self.people[id] = person

    cdef int infectChance(self, int time, person person) :
        """ This function calculates the chance that someone will
        get infect based on the infections in their innercircle.
        This chance is base on:
        - how many people in their household are infected
        - how many people in their classroom are infected
        and an overall difference if someone has been vaccinated or not
        and how long ago"""
        cdef int chance = 0

        # if cohab or classmate is sick, chance increases
```

```
        chance += self.cohabIsSick(person)
        chance += self.classmateIsSick(person)
        return min(chance, 100)



    cdef int cohabIsSick(self, person person):
        """ This function checks if there is someone sick in
        the same household as the person in question"""
        cdef int id = person.household
        cdef int r = self.housedict[id].infected
        if r > 0 :
            return 2
        return 0


    cdef int classmateIsSick(self, person person):
        """ this function checks if there is someone sick in
        the same household as the person in question"""
            cdef int id = person.schoolClass
            try:
                return othergroups[id].infected * 1
            except :
                return 0

    cdef float prevVaccinated(self, int time, person person):
        """ this function calculates how long ago someone has been
        vaccinated according to research the protection of a vaccine
        decreases over time"""
        cdef int number
        if person.vaccinated == 1 :
            number = time//7 + 47 - person.weekOfVaccination
            return self.vaccEffectiveness[number]
        else :
            return 1

    cdef infectRandom(self) :
        """This function randomly infects 10 people with ages
        between 12 and 60 these infections represent the infections
        that would result from getting infected from
        someone a person does not personally know. """
        cdef person person
        cdef int index, infectnumber = infectPeopleRandom
        for i in range(infectnumber) :
            index = random.randint(0,N-1)
```

```
        person = self.people[index]
        if (person.age < 12 or person.age > 60) and \
            person.susceptible == 1:
            index = random.randint(0, N-1)
            person = self.people[index]
        person.daysSinceInfection = 1
        self.people[index] = person

cdef vaccinate(self, int week ):
    """This function vaccinates a certain amount of people.
    The number of people that get vaccinated will decrease
    over time since the number of people getting a booster
    will increase over time """
    cdef int totalWeek = vaccinationsGiven * N, index
    cdef person person
    for i in range(totalWeek):
        index = random.randint(0,N-1)
        person = self.people[index]
        while person.vaccinated == 1 or \
              person.age < 12 or \
              person.susceptible == 0 or \
              person.dontVaccme == 1:
            index = random.randint(0,N-1)
            person = self.people[index]
        person.weekOfVaccination = week
        person.vaccinated = 1
        self.people[index] = person
        index -= 1
    return

cdef boosters(self,weekp):
    """In this function we give a number of people a boostervaccine.
    The vaccines are distrubted from oldest to youngest and you can
    only get a boostervaccine if you have already been vaccinated
    before. We update the week of vaccination to the current week
    and use a parameter startage which keeps track of where
    we left vaccinating the previous day."""

    # initialize variables
    cdef int numberOfBoosters = boostersGiven*N,
             week = weekp  + 47
    cdef person person
    cdef int age = self.startage

    # for every booster that we are going to give, we decide who will
```

```
        get it. we distribute them from oldest to youngest
        for i in range(numberOfBoosters):
            # if the person has not been vaccinated, is sick or has gotten
            # their vaccination within 6 months prior we skip them
            person = self.people[age]
            while person.vaccinated == 0 or \
                  person.susceptible == 0 or \
                  week - person.weekOfVaccination < 24:
                age -= 1
                if age < 1 :
                    self.startage = N -1
                    return
                person = self.people[age]
            person.weekOfVaccination = week
            self.people[age] = person
            age -= 1
        self.startage = age
        return

    cdef vaccinateChildren(self, int week):
        """In this function we vaccinate children under 12. We chose a
        random index and check if this kid has already been vaccinated,
        if so we chose another one."""

        # initialize variables used
        cdef int index, number
        cdef person person

        # we loop through the number of vaccines we are going to give
        number = int(vaccinationsGiven*N)
        for i in range(number):
            # less than 1/7th of the Dutch population is under 12
            index = random.randint(0,int(N/7))
            person = self.people[index]

            # if their age is not right, they are already vaccinated,
            # if they are sick, or they wish not to be vaccinated,
            # we skip them
            while person.age < 4 or \
                  person.age > 11 or \
                  person.susceptible == 0 or \
                  person.vaccinated == 1 or \
                  person.dontVaccme == 1   :
                index = random.randint(0,int(N/7))
                person = self.people[index]
```

```
            person.vaccinated = 1
            person.weekOfVaccination = week
            self.people[index] = person

    cdef changes update(self, fatality, changes newchanges):
        """This function updates the status and increments the number
        of days that a person has been infected.
        For a new infection, days[i]=1. For uninfected persons,
        days[i]=0.
        Input: infection fatality rate and age of persons i
        """
        # typing the variables
        cdef person person
        cdef int newinfections = 0, recovered = 0, id, houseID, schooolID

        for p in self.people:
            person = p
            id = person.person_id
            if person.daysSinceInfection > 0 :
                person.daysSinceInfection += 1
                if person.daysSinceInfection == 2 :
                    newchanges.newInfections += 1
                    newchanges.vaccinated += person.vaccinated
                    newchanges.nonvaccinated += 1 - person.vaccinated
                    newchanges.totalInfected += 1
                    person.susceptible = 0
                    person.infectionProtected = 1

            # on the day symptoms should be noticeable we check if
            # someone goes into quarantine. If someone does not go
            # in quarantine because they don't want to, they don't
            # know that they have COVID-19 or for other reasons, they
            # stay in their household. For this it doesn't matter
            # if they have or do not have symptoms
                if person.daysSinceInfection == DAY_SYMPTOMS :
                    if rd.random() < P_QUARANTINE:
                            # person gets symptoms and quarantines
                            person.quarantined = 1
                    else:
                            # person gets symptoms but does not quarantine
                            houseID = person.household
                            self.housedict[houseID].infected += 1
                            person.infectious = 1
                            schoolID = person.schoolClass
```

```
                    if schoolID > -1 :
                        self.othergroups[schoolID].infected += 1

            # on the recovery day of a person we check if they recover
            # or if they get admitted to the hospital
            # when they are hospitalised we change their status based
            # on if they had been quarantined.
        elif person.daysSinceInfection == DAY_RECOVERY :
                if rd.random() < RATIO_HF * fatality[person.age]:
                    if person.vaccinated == 1 and rd.uniform(0,1) < 0.95
                        continue
                    person.hospitalised = 1
                    if person.quarantined == 0:
                        person.infectious -= 1
                        houseId = person.household
                        self.housedict[houseId].infected -= 1
                        schoolId = person.schoolClass
                        if schoolId > -1 :
                            self.othergroups[schoolId].infected -= 1
                    else :
                        person.quarantined = 0
                    person.hospitalised = 1
                    newchanges.hospitalized += 1
                else:
                    if person.quarantined == 0 :
                        houseID = person.household
                        self.housedict[houseID].infected -= 1
                        schoolId = person.schoolClass
                        if schoolId > -1:
                            self.othergroups[schoolId].infected -= 1
                    person.infectious = 0
                    person.quarantined = 0
                    person.daysSinceInfection = 0
                    person.recovered = 1
                    person.daysSinceRecovery = 1

            # on the day of release of someone who has been
            # hospitalised, they have a chance of dying
        elif person.daysSinceInfection == DAY_RELEASE:
            #nchanges["hospitalized"] -= 1
            person.hospitalised = 0
            if rd.random() < 1 / RATIO_HF:
                person.infectious = 0
                person.quarantined = 0
                person.daysSinceInfection = 0
```

```
                    person.deceased = 1
                    newchanges.deceased += 1
                else:
                    person.infectious = 0
                    person.quarantined = 0
                    person.daysSinceInfection = 0
                    person.recovered = 1
                    person.daysSinceRecovery = 1

        if person.daysSinceRecovery > 0 :
            if person.daysSinceRecovery == 14 :
                person.susceptible = 1
            elif person.daysSinceRecovery == protectionInfection :
                person.infectionProtected = 0
                person.daysSinceRecovery = 0
            person.daysSinceRecovery += 1
        self.people[id] = person
    return newchanges
    def run_model(self,data, tracker, timesteps):
        """ This function simulates infections for a given number of
        days given by the input timesteps. The input for this function
        is the population containing information about all the people
        and their households, the contact matrix containing occasional
        meetings between people. A tracker that keeps track of the
        statistical changes and the number of timesteps."""

        # identify types of variables
        cdef person person
        cdef changes newchanges
        cdef int id, startage = N - 1, c,
        cdef float number, total = 0, r, p

        # initializing values for the tracker
        newchanges.totalInfected = 0
        newchanges.deceased = 0


        # initialize infections
        self.infectRandom()
        newchanges = self.update(data['IFR'], newchanges)


        # loop through time
        for time in range(timesteps):
            # printing the day we are currently simulating
```

```
sys.stdout.write('\r' + "Time step: " + str(time))
sys.stdout.flush()
# loop through all the people in the population,
# if they are not sick we calculate their change
# of getting infected. If they get infected it is
# their first day since infection.
for per in self.people:
    person = per
    if person.susceptible == 1 and  \
       person.infectionProtected == 0:
        r = rd.randint(0,100)
        p = self.infectChance(time,person)
        if r < p :
            r = rd.uniform(0,1)
            p = self.prevVaccinated(time,person)
            if r < p :
                person.daysSinceInfection = 1
                id = person.person_id
                self.people[id] = person

# we infect a number of people based on occasional meetings
self.infect_standard(time)

# we infect a number of people randomly.
# If there is a lockdown, we skip this
if not lockdown :
    self.infectRandom()

newchanges.newInfections = 0
newchanges.vaccinated = 0
newchanges.nonvaccinated = 0
newchanges.hospitalized = 0

# update the people in the population
newchanges = self.update(data['IFR'], newchanges)

# vaccinate a number of people who have not had a vaccine
# yet in the first 25 afterwards we have the same number
# of fully vaccinated people as in the Netherlands
# as of January 2022
if time < 25 :
   self.vaccinate(time//7)

# booster people if boolean is true
if booster :
```

```
                self.boosters(time//7)

            # vaccinate children if boolean is true
            if vaccCHildren and time < 40:
                self.vaccinateChildren(time//7)

            # update the tracker
            tracker.update_statistics(newchanges)
        return tracker


def model(filenames, type, timesteps):
    """ This function initializes and runs the model for a certain
    number of timesteps. It returns a pandas dataframe containing
    all data at time t for t in timesteps. """
    tracker = track_statistics()

    tracker_changes = tracker.init_empty_changes()
    data, network, tracker_changes, currentPopulation =
        initialise_model(filenames, type, tracker_changes)

    # update the dataframe
    tracker.update_statistics(tracker_changes)

    # preprocessing

    # creating lists
    vaccEffectiveness = VaccinationEffectiveness()
    meeting = pmeetinglist()

    # Run the model
    cdef simulation sim= simulation(vaccEffectiveness,
                                    meeting, network,
                                    currentPopulation.houseDict,
                                    currentPopulation.otherGroups,
                                    currentPopulation.people)
    sim = preprocessing(filenames, data, sim)

    # run model
    tracker = sim.run_model(data, tracker, timesteps)

    return tracker
```