Delft University of Technology
Faculty Electical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics
together with Hasselt University

A Spatial Markov Chain Cellular Automata Model for the Spread of the COVID-19 virus

Including parameter estimation

Bachelor thesis at
Delft Institute of Applied Mathematics
in coorporation with Hasselt University
as partial fulfillment of the requirements

for the degree of

BACHELOR OF SCIENCE
in
Applied Mathematics
by

Jenny Lu

Delft, the Netherlands, September 26, 2020

Bachelor thesis APPLIED MATHEMATICS

A Spatial Markov Chain Cellular Automata Model for the Spread of the COVID-19 virus

Including parameter estimation

JENNY LU

Delft University of Technology
Hasselt University

Supervisors

Dr.ir. F.H. van der Meulen
Prof. Dr.ir. F.J. Vermolen

Other commissioner

Dr. J.G. Spandaw

September, 2020 Delft

Abstract

In this bachelor thesis we propose a spatial Markov Chain Cellular Automata model for the spread of the COVID-19 virus as well as two methods for parameter estimation. Network topologies are used to model the progression of the epidemic by considering each individual on a grid and using stochastic principles to determine the transition between different states. The model is able to predict the time-evolution of outbreaks under different lockdown policies. Additionally, the impact of variation in infection probability and recovery rates on the amount of active cases, deaths as well as the length of the epidemic is investigated. These results can provide us with insights and predictions of the spread of the virus under different scenarios. Parameter estimation is done by using both Maximum likelihood estimation and Bayesian estimation based on simulated data. The produced estimates were relatively accurate, suggesting that these methods can be applied in order to estimate the parameters of the proposed model based on actual data.

Keywords

Keywords in this paper are: Mathematical model, COVID-19, Coronavirus, SARS-CoV-2, Pandemic, Numerical simulation, Parameter estimation, Markov chain cellular automata, epidemic model, maximum likelihood estimation, bayesian estimation.

Table of symbols

In the table 1 the main used symbols in this thesis are stated with their respective definition.

Symbol	**definition**
1	susceptible state of a person
2	infected state of a person
3	recovered state of a person
4	dead state of a person
$a_{ij}(t)$	intensity of the contact between individual i and individual j
$N_i(t)$	set of the neighbours of individual i at time t
$N_i^I(t)$	set of infected neighbours of individual i at time t
λ_g	overall infection probability rate
τ	short time interval
μ	recovery rate
T_r	time to recovery
T_d	time to death
α	probability of death
$\beta(t)$	lockdown parameter
T_{ld}	time interval of the lockdown

Table 1: Table of the symbols used in this thesis with their respective definitions.

Contents

1 Introduction

In December 2019, a severe outbreak of a novel disease caused by a virus in Wuhan, a city with over 11 million people in central China, started. This virus is currently known as coronavirus or COVID-19. Despite the immediate and drastic measures by the Chinese government, the infectious disease spread rapidly across China and many other countries [1]. On January 30^{th} 2020, the World Health Organization (WHO) declared the epidemic disease to be a Public Healthy Emergency of International Concern [2]. Even though the virus started in China, many other countries have suffered more deaths compared to China as a result of for instance inappropriate lockdown measures. Such countries include the United States of America, Brazil, Mexico, United Kingdom, India and Italy [3]. Other countries suffered less extreme than these countries, but the virus still caused a lot of chaos and lockdown rules.

When writing this report, the world is still threatened by this global pandemic caused by the COVID-19 virus. The infectious disease caused by the virus is renamed as strain severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) by the World Health Organisation [4]. The severity of the illness caused by the virus depends per individual. In general the elderly have a higher chance of dying compared to younger individuals. Certain health conditions are also of importance, such as asthma or heart problems. These people tend to have a bigger chance of dying from the disease. The disease caused by COVID-19, is characterised by flu-like symptoms such as fever, dry cough and tiredness . Serious symptoms that can arise are for instance chest pain, difficulty breathing or even the loss of speech or movement [5]. At the moment of writing, almost a million people have died globally [6].

The virus spreads from person to person who are in close contact through respiratory droplets, by contact with contaminated objects/surfaces or by direct contact with an infected person. Governments all over the world have implemented various lockdown policies in order to slow the spread of the disease. The severity of the lockdown rules varies from country to country. Some countries even implemented a curfew while others only encourage people to keep 1.5 metres distance. The infectious pandemics have substantial effects on finance as well. Millions of people have lost their jobs. Many companies are struggling to exist, and have to take heavy measures to keep the business running. Especially the passenger transport sector has been hit greatly. Airplane companies like KLM have drastically reduced the amount of flights and have fired many employees in order to exist. Several smaller airlines have already collapsed due to the COVID-19 outbreak. Another sector that is hit hard is the tourism sector [7]. Due to the lockdown rules, many countries have closed their borders for tourists and many people have also canceled their holiday plans. Restaurants and bars had to close or only limit

themselves to a certain amount of customers each time. It is therefore of crucial importance to be able to model the spread of the virus.

To predict the dynamics of the spread of the virus, many different mathematical models have been developed. Many of these models are based on the S(E)IR-model or another general purpose epidemic model. The models are then adapted specifically to model the spread of the COVID-19 virus. Some models are more general while others are made specific for a certain country, like China or Italy. These models are for instance presented in [8], [9] and [10]. Data from The approach that is presented in [11] has some similarities with the model that is presented in the current paper.

The main goal of this paper is to develop a mathematical model that is based on a cellular automata to model the spread of the COVID-19 virus. Besides the mathematical model, parameter estimation for this particular model is done in both a frequentist and a bayesian way. The model described in this report is looks at each person individually and also incorporates the influence of the topology of the network on the evolution of the epidemic. Another difference with the common SIR model is that this model is also of stochastic nature. The proposed model is still a simple model with few parameters.

We consider a fixed population of n people, where we are going to look at each person individually. Every person has a set of neighbours, which represent the people he/she is in contact with. Contact between individuals does not always lead to infection and therefore a stochastic process is considered. Furthermore, the intensity of the contact between two people varies from couple to couple. This is logical as in reality a person might have more contact with his roommate than with a friend who lives overseas. The probability that each person gets infected is determined by the intensity of the contacts between them.

Once a person is infected, he/she can either recover or die in this model. Recovery is also implemented as a stochastic event with a certain probability of happening. Death is modeled by tracking the amount of time a person is infected. So, if a person is still infected after a certain period of time, this models assumes that this person will then die. The reason behind this is because the longer an individual is infected by the disease, the worse the condition of vital organs such as the heart and lungs becomes. Of course, this is not exactly what happens in reality, but this assumption makes some sense. In this case we are only going to consider deaths as a result of the coronavirus, other causes of deaths are not taken into account. Lockdown policies have been implemented in the model by adjusting a prespecified parameter.

The model has input parameters that can be estimated by a given data set. In order to do this, both a frequentist and a Bayesian way of parameter estimation are discussed. The frequentist way is by means of maximum likelihood estimation,

while the Bayesian way is done with two Markov Chain Monte Carlo methods called the Metropolis-Hastings Algorithm and the Metropolis-Adjusted Langevin Algorithm. Data validation has not been done yet, but future studies are recommended to extent the current model and use data validation to adjust the model to specific countries or regions.

The structure of this thesis is as follows. First, in section 2 the mathematical model will be developed and explained. Subsequently, the numerical model will be derived in section 3. Next, various lockdown scenarios are going to be considered in section 4. Section 5 analyses the effect of variation in the infection probability and recovery rate on the evolution of the epidemic. Subsequently, parameter estimation using Maximum Likelihood and Bayesian estimation is explained in sections 6, 7 and 8. Sections 9 and 10 discusses the findings and conclusions. Finally, in the Appendix 11 figures from the simulations are presented, in the Appendix 12 some concepts from probability used in this report are stated and in the Appendix 13 explanations of implementations in Python can be found.

2 The Mathematical Model

In this section the mathematical model will be derived and explained. In order to understand the model, first the two concepts of cellular automata and Markov are explained. After that, the model will be derived.

2.1 Cellular automata

The model that will be presented in this paper is based on a cellular automaton. A cellular automaton consists of a grid of cells, where each cell can be in a finite number of discrete states. According to a set of rules that incorporates neighbouring cells, the cells enter different states over discrete time steps. The rules are applied to each cell iteratively at every time step. The state that a cell enters is based on its current state as well as the state of its neighbours.

2.2 Markov chain

A Markov chain is a sequence of random variables X_1, X_2,... such that the conditional distribution of X_{n+1} given the previous variables X_1,...X_n depends only on the information given by X_n. An equivalent formulation is that given the 'present' value of the random variable X_n, the 'future' value of the random variable X_{n+1} is independent of the 'past' (which is represented by X_1,...,X_{n-1}). Only the current information is relevant, the history is irrelevant. In this paper we will only consider Markov chains that are 'time-homogeneous'. This means that the conditional distribution of X_{n+1} given X_n does not depend on n. Therefore, the transition from one state to another state always follows the same 'mechanism' or 'rule'. (Note that the definition for the random variables is also valid for random vectors.)

2.3 The set-up of the model

The mathematical model that is proposed here is used to evaluate the spread of the disease within a grid, which represents a certain area or territory, during a fixed time interval. We do not consider movement of people between territories. Consider a graph consisting of nodes and edges. Every node represents an individual and every edge represents the interpersonal relations. So if node i is connected to another node j by means of a edge, then person i and person j are in direct physical contact with each other. If there is no edge, then it means that these two people do not have direct physical contact with each other. They might still be in contact via Skype or mail, but we are only considering direct physical contact here, as the virus can only spread via this way.

Every individual can be in one of four different states: susceptible, infected, recovered (or resistant) or dead. It is assumed that if a person is susceptible, then he/she can only become infected. Once the person is infected, he/she can either stay infected or go to the recovered/resistant state or the dead state. No other transitions are possible in the current model. Another assumption that we use is that only infected people spread the disease. So if an individual is in a susceptible, recovered/resistant or dead state, he/she is not able to pass on the virus. This assumption is, of course, not in line with reality. The disease can also be transmitted through surfaces or objects and studies have shown that just recovered people can still transfer the disease up to two weeks after recovery [12]. However, this effect is neglected in the current modelling. From now on, we will refer to the recovered or resistant state as 'recovered'. Aside from that, in this model we assume that people can only die because of the coronavirus. Other ways of dying are neglected. Since people periodically catch up with family members, relatives, and friends, we incorporate the time-evolution of contacts between individuals as well.
Mathematically we use the following notation: we consider a constant population of n individuals. We introduce the vector $\mathbf{X}$ as a vector of length n and this vector contains the states of all the individuals. Note that the dead people are also counted as being part of the population (even though in reality they are essentially not). Every person's state will be denoted by x_i where the value of x_i is an integer in the set $\{1, 2, 3, 4\}$ and i is the index of the i^{th} person in the model.

The numbers in the set $\{1, 2, 3, 4\}$ represent the following:

1. corresponds to the susceptible state
2. corresponds to the infected state
3. corresponds to the recovered state
4. corresponds to the dead state

The connection between person i and person j at time t is denoted by $a_{ij}(t)$, where $a_{ij}(t) = 0$ represents the case when there is no physical contact between individuals i and j. This connection is time-dependent as the intensity between contacts varies over time. Large values of $a_{ij}(t)$ represent a lot of contact between the two people and small values of $a_{ij}(t)$ represent less contact. All the values of $a_{ij}(t)$ are stored in a matrix $A(t)$, which will be referred to as the *adjacency matrix*.

2.4 The transfer of the virus between individuals

In this section we describe the transfer of the virus between individuals. For simplicity, we are only going to focus on the transfer of the virus between two different individuals: from individual i to individual j, where $i, j \in \{1, 2, ..., n\}$. Assume that two people i and j are in physical contact, where the value of $a_{ij}(t) \in (0, 1]$. All the other cases can then be derived from this 'two individual' case.
Suppose that individual i is infected and that individual j is susceptible. The time between going from the susceptible state to the infected state is assumed to follow an exponential distribution with infection parameter rate $\lambda_{ij}(t)$ in a given time interval denoted by τ. The time interval τ is assumed to be small. Hence, the probability that person j becomes infected in the small time interval τ, given that person i is infected, is given by:

$$P(x_j(t+\tau) = 2 | x_j(t) = 1, x_i(t) = 2) = \int_t^{t+\tau} \lambda_{ij}(s) e^{-\lambda_{ij}(s)(s-t)} ds. \tag{1}$$

Since it is only possible to go from the susceptible state to the infected state, the probability that person j stays susceptible in a small time interval τ, given that he/she was susceptible at time t and person i is infected at time t, is given by:

$$P(x_j(t+\tau) = 1 | x_j(t) = 1, x_i(t) = 2) = 1 - \int_t^{t+\tau} \lambda_{ij}(s) e^{-\lambda_{ij}(s)(s-t)} ds. \tag{2}$$

Therefore, the probability that person j dies or recovers from the disease is zero. Hence:

$$P(x_j(t+\tau) \in \{3, 4\} | x_j(t) = 1, x_i(t) = 2) = 0. \tag{3}$$

The infection rate parameter $\lambda_{ij}(t)$ is assumed to be of the following form:

$$\lambda_{ij}(t) = \sum_a {}_{ij}(t) \lambda_g \tag{4}$$

where λ_g is a general infection rate parameter that is assumed to be the same for every individual.

Now that we have defined the probability of becoming infected if a person comes in contact with an infected individual, we consider the set of people where the person is in contact with. Define the set N_j of people where person j is in contact with by:

$$N_j(t) = \{k \in \{1, ..., n\} : a_{kj}(t) > 0\}. \tag{5}$$

where the N represents the 'neighbours' of person j, so the people that person j is in contact with. This set can be reduced to a set where we only consider all the neighbours of individual j that are in the infected state, which will be denoted by N_j^I:

$$N_j^I(t) = \{k \in N_j(t) : x_k(t) = 2\}. \tag{6}$$

where the superscript I denotes the infected individuals.
Next, we assume that all the states of the individuals in the 'neighbour' set N_j or N_j^I are all independent of each other. This makes sure that we can take the product of all the probabilities. Therefore we define the probability that node j stays susceptible as follows:

$$P(x_j(t+\tau) = 1 | x_j(t) = 1) = \prod_{k \in N_j^I(t)} (1 - \int_t^{t+\tau} \lambda_{kj}(s) e^{-\lambda_{kj}(s)(s-t)} ds). \tag{7}$$

As a direct consequence, we have that the probability that node j becomes infected is given by:

$$P(x_j(t+\tau) = 2 | x_j(t) = 1) = 1 - \prod_{k \in N_j^I(t)} (1 - \int_t^{t+\tau} \lambda_{kj}(s) e^{-\lambda_{kj}(s)(s-t)} ds). \tag{8}$$

Assume that during the time interval $[t, t+\tau]$ where $s \in [t, t+\tau]$ we have that $\lambda_{kj}(s) = \lambda_{kj}(t)$. Then we can rewrite equation (8) above as:

$$P(x_j(t+\tau) = 2 | x_j(t) = 1) = 1 - \prod_{k \in N_j^I(t)} e^{-\lambda_{kj}(t)\tau} = 1 - e^{-\tau \sum_{k \in N_j^I(t)} \lambda_{kj}(t)}. \tag{9}$$

If we substitute the definition of $\lambda_{kj}(t)$:

$$P(x_j(t+\tau) = 2 | x_j(t) = 1) = 1 - \prod_{k \in N_j^I(t)} e^{-\lambda_{kj}(t)\tau} = 1 - e^{-\tau \sum_{k \in N_j^I(t)} a_{kj}(t) \lambda_g}. \tag{10}$$

From equation (10) above it can be seen that there is some kind of effective transfer probability rate that we can define for each node x_j, namely:

$$\lambda_j^{eff} = \lambda_g \sum_{k \in N_j^I(t)} a_{kj}(t). \tag{11}$$

2.4.1 Different interpretation

Recall from probability theory that the minimum of independent exponentially distributed random variables is still exponentially distributed. The parameter that corresponds to this random variable is then the sum of all the rate parameters. Thus if $Y = min\{Y_1, ..., Y_n\}$ where $Y_i \sim Exp(\lambda_i)$ are all independent exponentially distributed random variables, then $Y \sim Exp(\sum_{i=1}^{n} \lambda_i)$.
We can also interpret the expression in equation (12) differently.

$$P(x_j(t+\tau) = 2 | x_j(t) = 1) = 1 - e^{-\tau \sum_{i \in N_i^I} \lambda_{kj}(t)} \tag{12}$$

Define the random variable $T = e^{\sum_{k \in N_j^I} \lambda_{kj}(t)}$, then this random variable is exponentially distributed with rate parameter $\sum_{k \in N_j^I} \lambda_{kj}(t)$, as it is assumed that all the neighbours k in the set N_j^I are independent of each other.
The time of infection for every neighbour $k \in N_j^I$ is exponentially distributed with parameter λ_{kj} and independent of each other, therefore the distribution of T is the same as the distribution of the minimum T_k over all $k \in N_j^I$. So:

$$T \overset{\text{distributed}}{=} min_{k \in N_j^I}(T_k)$$

If we would use this interpretation in equation (12) above, we see that the probability that person j becomes infected, given that it was not infected before in a small time interval $[t, t+\tau)$ is the same as the probability that $T \leq \tau$, so:

$$P(x_j(t+\tau) = 2 | x_j(t) = 1) = P(T \leq \tau) \tag{13}$$

Essentially we are sampling from the minimum of all exponentially distributed infection times.

2.5 The transition from the infected state to the recovered or dead state

Once a person has become infected by the virus, he/she will either recover or die after some time. Some people recover very quickly after only experiencing some (mild) symptoms while others need a very long time (possibly even on the intensive care) to recover or might even die.
In the current model, the time to recovery is assumed to be exponentially distributed with probability rate parameter $\mu > 0$. The parameter μ can be seen as the rate for recovery from the virus. It is also assumed that this rate μ is constant over time. Hence all individuals are assumed to have the same recovery probability rate. In reality this assumption is violated at this rate is dependent on the medical treatment the person might get, the severity of the symptoms, the age of the person and many more other factors.
The probability that a person i transitions to the recovered state is as follows:

$$P(x_i(t+\tau) = 3 | x_i(t) = 2) = \int_t^{t+\tau} \mu e^{-\mu(s-t)} ds. \tag{14}$$

The probability that a person stays infected is then given by:

$$P(x_i(t+\tau) = 2 | x_i(t) = 2) = 1 - \int_t^{t+\tau} \mu e^{-\mu(s-t)} ds. \tag{15}$$

By the properties of exponentially distributed variables, the expected recovery time from the moment that a patient has become infected is determined by:

$$T_r = \frac{1}{\mu}$$

where the subscript $_r$ is short for 'recovery'. The probability of dying is a bit more complicated. In this model it is assumed that if a person has been infected during a time-interval that exceeds a threshold, say $T_{death} = M \times \tau$, where $M > 0$ is some positive integer value, then the person dies with probability one. This is of course not always the case in reality, but it is a reasonable assumption. The rationale behind this assumption is that a long lasting exposure to the disease potentially damages the patient's vital organs so much that he/she dies.
To develop our intuition behind the relation between the recovery rate and time interval of death T_d, we assume that the probability that someone dies from the disease is given by α. Hence all patients that have been ill over a period that exceeds T_d are assumed to die. Then if person i was infected at time t, then the time interval of death and the probability to die are related by:

$$1 - \int_t^{t+T_d} \mu e^{-\mu(s-t)} ds = \alpha \tag{16}$$

We can rewrite it as:

$$\int_0^{T_d} \mu e^{-\mu(s-t)} ds = 1 - \alpha$$

$$1 - e^{-\mu T_d} = 1 - \alpha$$

$$T_d = -\frac{1}{\mu}\log(\alpha)) = -\log(\alpha) T_r \tag{17}$$

where log is the natural logarithm and α is a very small probability that is at most 2-3 %.
If it is assumed that person i got infected at time $t_{inf} = \min_{t>0}\{x_i(t) = 2\}$ we can write the following mathematically:

$$x_i(t_{inf} + \theta) = \begin{cases} 3, & \text{if } \theta < T_d \\ 4, & \text{if } \theta \geq T_d \end{cases} \tag{18}$$

where θ is amount of time that person i has been infected.

2.6 Lockdown policies

Many (national) governments have issued lockdown policies, in which these policies aim at reducing the number of interpersonal contacts. The reduction of interpersonal contacts should lead to lower virus reproduction numbers and to a lower hospitalization load. An absolute lockdown could even lead to a total eradication of the virus. Governments and policy-makers bear in mind that severity of lockdown policies compromises the national economy.
The lockdown rules could hold for the entire country, but many governments have also imposed rules that only hold for certain states, regions or cities. The severity of the lockdown policies also vary from country to country, dependent on the amount of infected people and the evolving spread of the virus. Some common lockdown policies include [13]:

- Keep 1.5 metres distance when outside.
- Wear a face mask in social places.
- Restaurants/bars/shops are only allowed to serve a limited amount of customers.
- Work from home.
- Wash your hands often.
- Avoid busy places.
- If you have any symptoms, refrain from social contact, and make sure to get tested as soon as possible.
- No public gatherings.
- Only inland flights.

Severe lockdown policies include for instance:

- No travel to and from a certain region.
- No travel to and from a certain city.
- You are only allowed to go outside for 1 hour a day.
- Always wear a face mask whenever you go outside.
- All social/public places are closed.

- At most two people are allowed to be together outside.
- All schools are closed.
- Everybody needs to stay at home during certain hours of the day.

There are obviously many more rules and every country has implemented different ones. The list is to give an impression of examples of lockdown rules.
In the model we will model the lockdown policy by the parameter $\beta(t)$. It is time-dependent as the lockdown rules vary from time to time due to the relaxation and sharpening of the lockdown policies.

3 The Numerical Model

In this section the implementation of the numerical method will be explained. The implementation has been done in Python 3.7.
The model is used to evaluate the spread of the disease caused by the coronavirus within some bounded territory during a fixed time interval. At the beginning of each simulation, the model parameters are set by the user (such as the size of the grid, the number of time steps, the value of λ_g and μ). We start by simplifying the model to a square grid in which every node has at most four connections. That means that every person will only have at most four neighbours or people that he/she is in contact with. This has been illustrated in Figure 1.

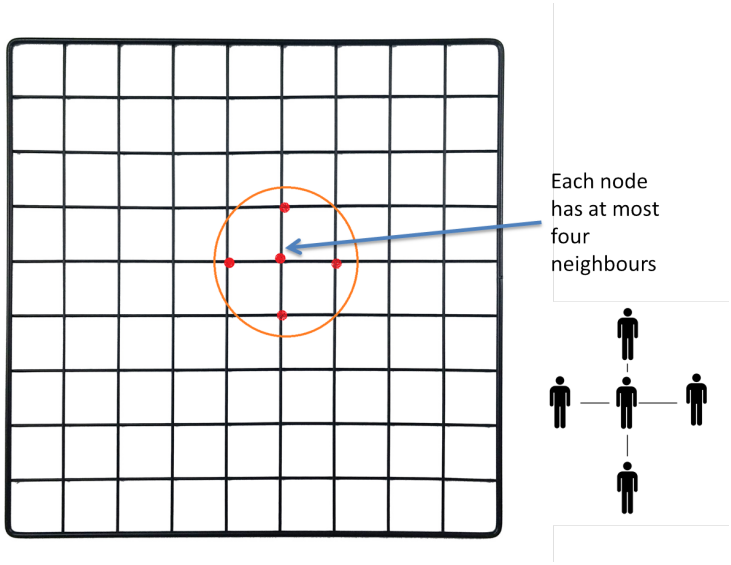

Figure 1: Square grid used, note that the dimensions of the grid are taken to be 9x9 for clarity, but of course during the simulations we will take a larger grid.

Every person on the grid will have an x and y coordinate to locate his/hers position. The grid is set up using *meshgrid* from the numpy package in Python.
All the input parameters used in the simulation are then specified. The simulation starts by entering a while loop which loops until the time reaches the end time of the simulation (specified prior) or when there are no more infected people left.
In order to model the transmission of the virus between individuals, we use the overall infection probability rate λ_g. Every time step τ the probability of getting infected is calculated and subsequently compared to a random number χ which is drawn from a standard uniform distribution between 0 and 1, that is $\chi \sim U(0, 1)$. If the number is smaller than the probability of getting infected, then the state of the person is changed from susceptible to infected. Otherwise he/she remains

susceptible. So in mathematical notation:

$$x_i(t+\tau) = \begin{cases} 1, & \text{if } \chi < P(x_i(t+\tau)=2|x_i(t)=1) \\ 2, & \text{if } \chi > P(x_i(t+\tau)=2|x_i(t)=1) \end{cases} \tag{19}$$

where $P(x_i(t+\tau)=2|x_i(t)=1)$ is the probability of getting infected. This probability is explained and given in section 2.
The transition from the infected state to the recovered or dead state is done in a similar way. However, since the transition from the infected state to the death state requires the amount of time that a person is infected, we keep track of the time-interval that a person (so a node) is infected. Every time a person stays infected we add the time-interval τ to the infection time. If the total length of the time interval that a nodal point remains in the infected state exceeds the time interval T_d, which was earlier specified as the time to death, then the state of the person is changed to the dead state. Otherwise, the state of the person remains in the infected state or changes to recovered state.
So, once a person is infected we keep track of the amount of time he/she is infected in the *TimeInfec* matrix. Then every time step τ we calculate the probability of recovery $P(x_i(t+\tau)=3|x_i(t)=2)$ according to equation 14 and compare it to a random number $\xi \sim U(0,1)$.

$$x_i(t+\tau) = \begin{cases} 2, & \text{if } \xi < P(x_i(t+\tau)=3|x_i(t)=2) \\ 3, & \text{if } \xi > P(x_i(t+\tau)=3|x_i(t)=2) \end{cases} \tag{20}$$

If the person remains infected, we add τ to the time of infection of that person.

$$x_i(t+\text{total time infected}) = \begin{cases} 2, & \text{if total time infected} < T_d \\ 4, & \text{if total time infected} > T_d \end{cases} \tag{21}$$

Since the probability of getting infected is dependent on the interpersonal contact with neighbours (so in this case the nodes that a node is linked with), it is necessary to also set up the adjacency matrix $A(t)$. Interpersonal contacts are often fluctuating (due to meetings with friends, working, shopping etc.) that is why we use randomised values drawn from a standard uniform distribution on the interval zero to one. That is, when considering person i:

$$\text{For } s \in (t, t+\tau) : a_{ij}(t) \sim U(0,1), \quad \text{if } j \in N_i. \tag{22}$$

We then sum over all the neighbours to compute the probability of getting infected. Lockdowns are implemented by premultiplying the adjacency matrix $A(t)$ by a

factor $\beta(t)$, whose value ranges between zero and one. Small values of $\beta(t)$ refer to severe lockdowns, while larger values represent more mild lockdown policies. In that case the adjacency matrix is re-defined into:

$$\hat{A}(t) = \beta(t)A(t), \tag{23}$$

So in all the expressions of the probabilities defined earlier we replace the entries in the adjacency matrix $A(t)$ by $\hat{A}(t)$, so $\hat{a_{ij}} = \beta(t)a_{ij}(t)$. The lockdown policies vary over time, that is why β is depends on t.
In addition, the implementation of the model involves matrices that store the states of the person at each time step based on their position on the grid. As there are four possible states a person can be in, there are four matrices created: one for the susceptible people, one for the infected, one for the recovered and one for the dead people. This is done in a binary way. If a person is in a certain state, there will be a 1 in the matrix, if not then there is a 0 in the matrix. At the end of the simulation, the states of all the people can be seen in these four matrices. Of course it is also possible to make this into one large matrix. This is done with a matrix called *Infomatrix*. In this matrix the state of every individual is stored according to the {1,2,3,4} numbering that is defined previously.

4 Simulation Results

In the previous sections the mathematical model has been derived and the numerical implementation has been explained. In this section the results of various simulations will be shown and examined.

In all of these simulations, a rectangular arrangement of 100x100 nodes is taken. Every internal node will only have four neighbours (left, right, up, down), every boundary node will have three neighbours and every corner node will only have two neighbours. Initially all the nodes are in susceptible state, except for a node in the lower left corner, that node is infected. Without an infected person, the model will never predict the spread of the virus. The reason why it has been chosen in the lower left corner is that this grid can be seen as one of four 'quadrants', so the results can be reflected towards the other three quadrants. It is to be noted that the results of the current simulations are hypothetical as we have only used hypothetical values. Later on in this paper, parameter estimation will be done in order to find the input parameters of the model, given a simulated data set. This enables an estimate of the input parameters based on observed data and therewith the model can be made predictive.

The parameters used in the simulations are stated in Table 2. Later on, when considering different lockdown scenarios the time of the start and end of the lockdown will be changed to see the effect of the lockdown policies.

Parameter	**Value**
grid $(n_x \times n_y)$	100×100
τ	1
λ_g	0.5
μ	0.1
end time	100τ
time lockdown start	15τ
time lockdown end	60τ
Time to death T_d	8

Table 2: Table of parameters used in the simulations

The state of every person in the population is represented by a colour. All the colours and states of the people are stated in Table 3 and in Figure 2.

Colour	**State of person i in the population**
Yellow	Susceptible (or 1)
Red	Infected (or 2)
Green	Recovered (or 3)
Blue	Dead (or 4)

Table 3: Table of the colour coding used in the pictures of the states of the system

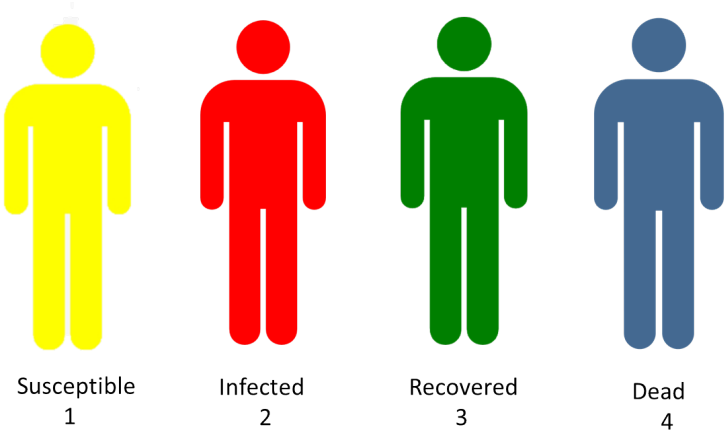

Figure 2: The states of every person with their corresponding colour.

4.1 Simulation 1: no lockdown

We start by examining the scenario where a country has not implemented a lockdown (in the model it means that $\beta(t) = 1$). We expect that the virus will be able to spread rapidly in the population, causing many active cases in a short period of time. The simulation has been run and the states of the evolution of the system are plotted in Figure 3 with the time at the top of each sub-figure.

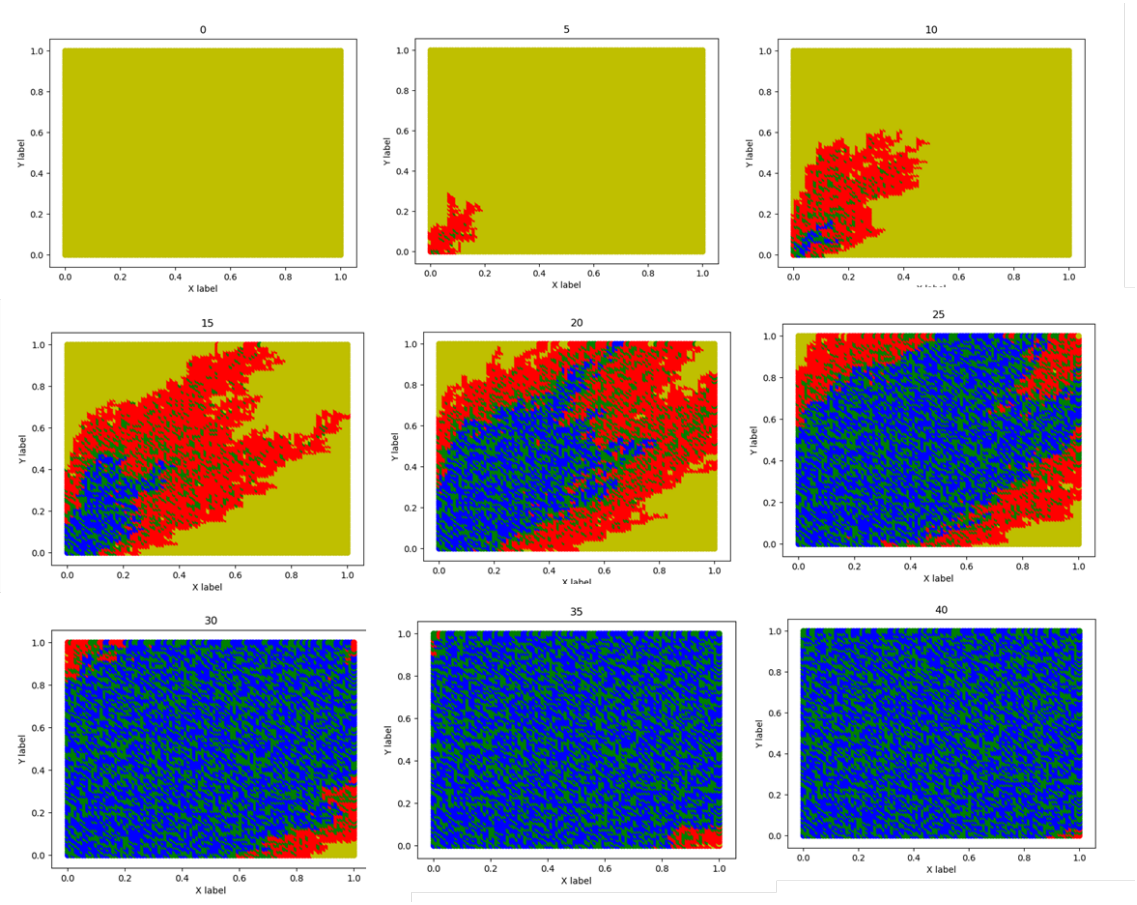

Figure 3: Graph of the state of the system at different times for a 100x100 grid and no lockdown.

From Figure 3 we see that the virus rapidly spreads among the people (almost in a 'flame-like' shape). Eventually the number of active cases decreases (as many people have already died or are recovered) until eventually everybody has either recovered, died or remained susceptible, as there is a possibility that some people never become infected. The phenomenon of people who remain susceptible at the end of the simulation occurs at each simulation scenario later in this section as well.

The end state of the system can be seen in Figure 4. We see that the simulation ended after 44 time steps.

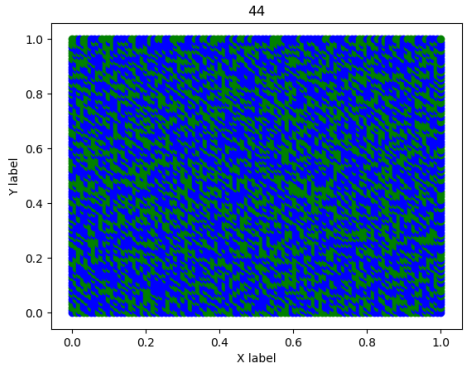

Figure 4: Graph of the end state of the system for a 100x100 grid and no lockdown.

We will refer to the different groups of people in the population as the Susceptible, Infected, Recovered and Dead sub-populations. These sub-populations consider the number susceptible, infected, recovered and dead people respectively at a certain time interval in the simulation. Capital letters have been chosen as we refer to this specific sub-population group. To get a better understanding of how the virus spreads, the sub-populations have been plotted against the time in Figure 5. In this figure we have counted the number of susceptible, infected, recovered and dead people at each time step and graphed it. It is not cumulative over time, that is the reason why after time 44τ, there are no new susceptible, infected, recovered or dead people. In Figure 6 the Recovered and Dead sub-populations have been plotted cumulative against the time. The Susceptible and Infected sub-populations have not been plotted cumulative, because the number of susceptible people only decreases over time and eventually all the infected people recover or die.
Figures 5 and 6 show that the virus spreads exponentially in the beginning, causing a rapid growth in the number of active cases and a drop in the number of susceptible people. The Infected sub-population graph looks almost like a bell-curve shape. It starts with exponential growth up until time 15τ and then it makes a round turn and has a more parabola shape afterwards, until it flattens out as there are no more infected people left.

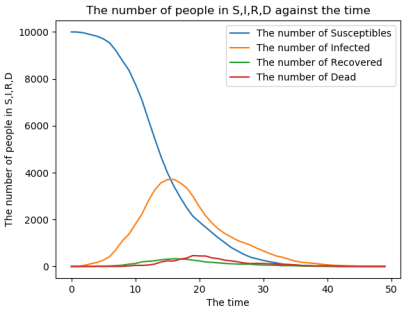

Figure 5: Graph of the number of people in Susceptible, Infected, Recovered, Dead sub-populations against the time with no lockdown.

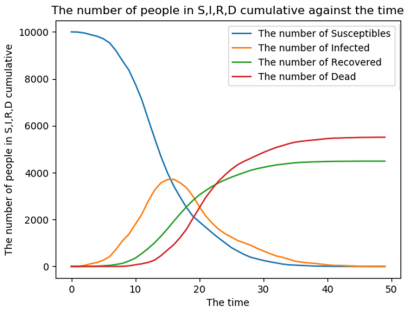

Figure 6: Graph of the number of people in Susceptible, Infected, Recovered, Dead sub-populations cumulative against the time with no lockdown.

The reason why there is a turn in the number of infected people is because at time 15τ we do not have 'enough' susceptible people left to obtain a higher peak than before, therefore the number of infections decreases afterwards. The Susceptible sub-population graph has a very steep negative slope, suggesting that the number of susceptible people rapidly decreases. The number of infected people follow a similar trend, but then reversed (so a rapid increase). After the number of infected people have peaked, the decrease in the number of susceptible people also slows down, until there are no more susceptible people remaining.
The graphs of the Recovered and Dead sub-populations in Figure 5 also show some peaks. The Recovered sub-population graph peaks a little later compared to the Infected sub-population graph at around time 18τ. This is due to the fact that it takes some time for infected people to recover. The same reasoning applies to the Dead sub-population graph as this graph peaks around time 20τ.
The cumulative graphs of the Recovered and Dead sub-populations in Figure 6 look like logistic growth. The Recovered sub-population graph starts to rise earlier compared to the Dead sub-population graph. This is in line with Figure 5 where we observed that the peak of the number of recovered people was earlier compared to the number of dead people. In this unfortunate situation we have more dead people than recovered people, but that is due to the way the parameters in the simulation have been chosen. If a different set of parameters were chosen, then it might be that there are more recovered than dead people.

4.1.1 Uncertainties in the simulations when there is no lockdown

The mathematical model that is presented is based on random numbers, therefore each simulation will have a different graph of all the four sub-populations. To see the possible 'bandwidth' that a sub-population might have, the simulation has been carried out a hundred times and the various sub-populations have been plotted against the time. This can be seen in Figure 7.

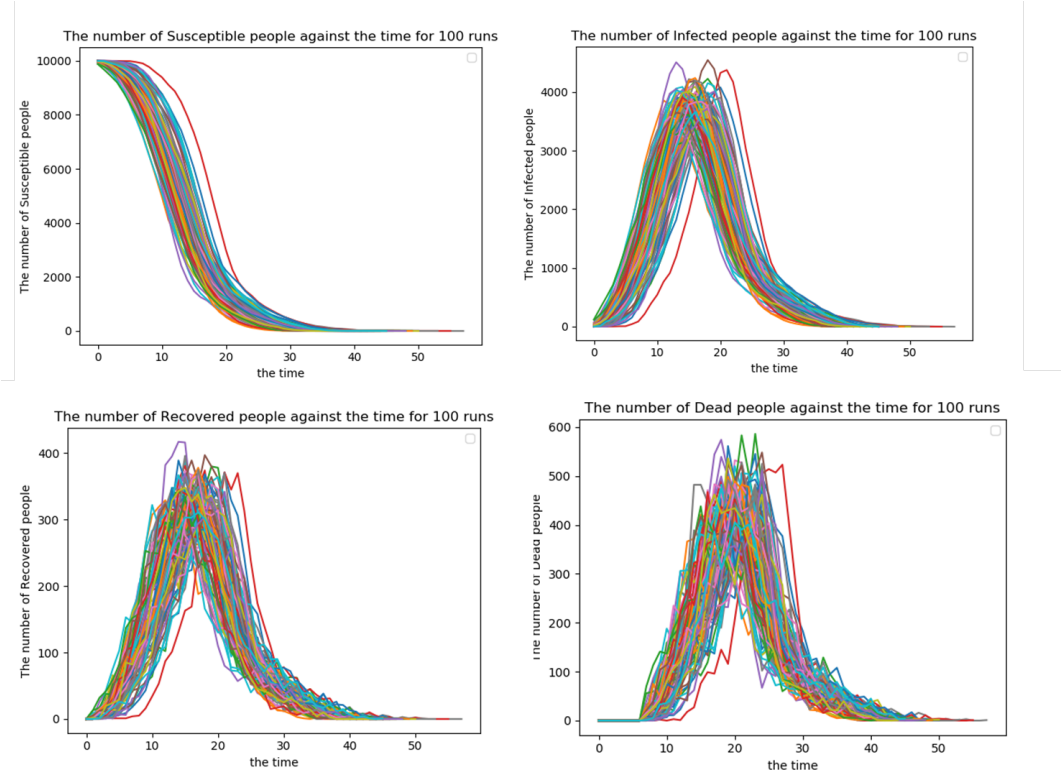

Figure 7: Non-cumulative graphs of the number of people in Susceptible, Infected, Recovered, Dead sub-populations against the time with no lockdown for 100 simulations.

From these graphs it can be seen that all the graphs of the different sub-populations all look quite similar as they show similar trends and behaviours. It is also interesting to consider the cumulative graphs of the Recovered and Dead sub-populations. These graphs are found in Figure 8.

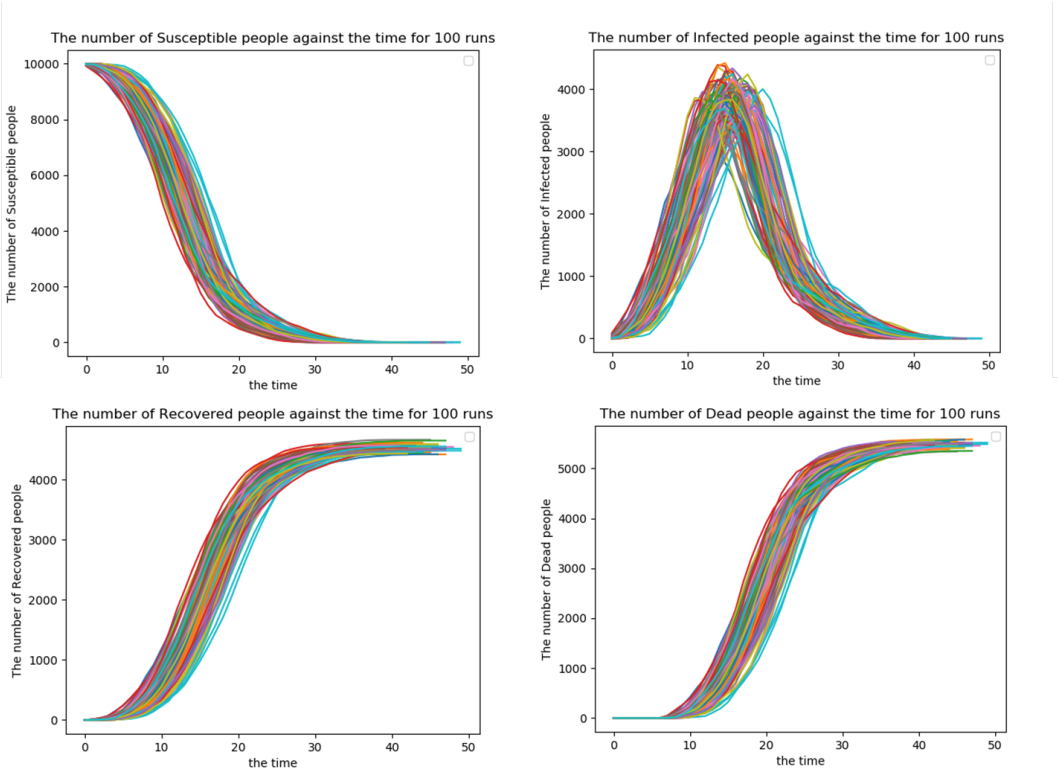

Figure 8: Cumulative graphs of the number of people in Recovered and Dead sub-populations against the time with no lockdown for 100 simulations. The graphs of the susceptible and infected sub-populations are not cumulative.

If we consider the top two figures in figures 7 and 8 more closely, then it is noticeable that the peak of the Infected sub-population graphs is around time 15τ if there is no lockdown implemented. This also corresponds to the steepest slope in the Susceptible people sub-population graphs. It might be interesting to incorporate a lockdown scenario at time 15τ. This will be done later in this section.
From Figure 8 it can be seen that the number of recovered people fluctuates around the 4000 to 4500 (when considering a total population size of 10000 people), which is roughly 40-45% of the total population. The number of dead people ranges between 5000 and 5500, which is roughly 50-55% of the total population. This is of course not the ideal outcome, but it is a result of the parameters chosen in the simulations. Figures 7 and 8 show that all the simulation results are consistent, but possess some variation.

4.2 Simulation 2: $\beta(t) = 0.5$

In this simulation a medium lockdown has been implemented. This can be seen as a lockdown scenario in between a very strict and a very soft lockdown, that is we use:

$$\beta(t) = \begin{cases} \tilde{\beta}(t) = 0.5, & \text{for } t \in T_{ld} = (15\tau, 60\tau) \\ 1, & \text{else.} \end{cases}$$

We essentially halve the amount of physical social interaction between individuals. The time evolution of the system is depicted in Figures 70 and 71 in the Appendix 11, where Figure 71 is the continuation of Figure 70. The analysis of these figures can also be found in the Appendix 11. Just like in the case when there was no lockdown, we consider the sub-populations of the susceptible, infected, recovered and dead people separately. The graphs are found in Figures 9 and 10.

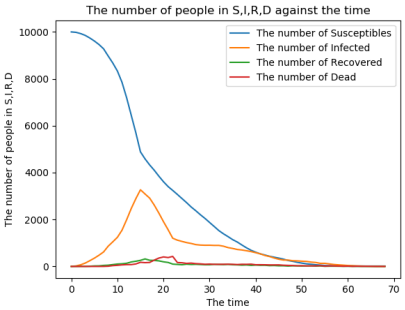

Figure 9: Graph of the sub-populations against the time for a medium lockdown of $\beta(t) = 0.5$ where the lockdown starts at time 15τ and ends at 60τ.

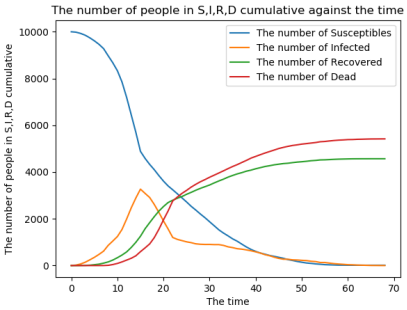

Figure 10: Cumulative graph of the sub-populations against the time for a medium lockdown of $\beta(t) = 0.5$ where the lockdown starts at time 15τ and ends at 60τ.

The impact of the lockdown at time 15τ is clearly visible in Figure 9: instead of smooth curves, some discontinuities in the derivatives in the graphs are present (especially in the curves of the susceptible and the infected people). It can be seen that the lockdown stretches the duration of the outbreak over a longer time period. This is advantageous to a decreased hospitalization load, which means that the peak burden on hospital resources and staff has been reduced. However, the number of casualties (dead) or recovered people in the end is the same with respect to the case in which there was no lockdown implemented. This can be seen in Figure 10.

If we compare Figure 10 to Figure 6, we see that besides the discontinuities in all of the four graphs and that the graphs are stretched over a longer time period, the overall trend is similar. Therefore, it can be concluded that a lockdown does help the epidemic from spreading too rapidly and that the hospitalization load can be reduced so that the peak burden on hospital equipment and staff is decreased.

4.2.1 Uncertainties in the simulations for a lockdown with $\beta(t) = 0.5$

Similar to the case when there is no lockdown, we present the results from a hundred runs of the model to show the variability in the results. The graphs are plotted in Figures 11 and 12.

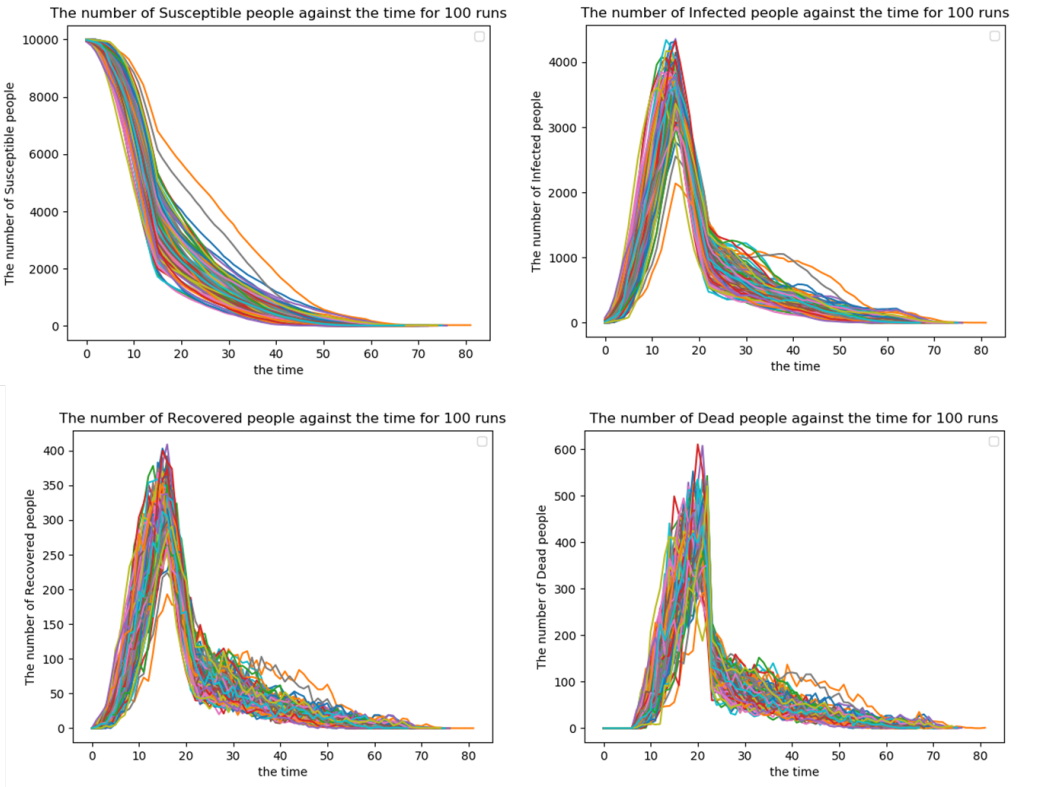

Figure 11: Graph of the number of people in the Susceptible, Infected, Recovered or Dead sub-populations against the time for a medium lockdown of $\beta(t) = 0.5$ where the lockdown starts at time 15τ for 100 simulation runs.

The non-cumulative graphs in Figure 11 show that there is some variation the the modelling results, but all the curves of the four sub-populations have similar trends. In the various graphs of the Susceptible sub-population, the discontinuity

of the derivatives at time 15τ is clearly present as a result of the lockdown. The main difference between the graphs is that the discontinuity in the derivative at time 15τ occurs at a different number of susceptible people in the population. This phenomenon is also reflected in the number of infected people: the lower the number of susceptible people at time 15τ, the higher the number of infected people, thus the less effective the lockdown is and vice versa.

The maximum number of infected people at time 15τ has quite a large range. The highest peak is at about 4300 number of infections, while the lowest is around 2000 active cases. This is a very large difference, as we are considering a population of 10000 people. So in some cases about 43% of the population might be infected at maximum, while in other cases only about 20% is infected. This can is partly be explained by the randomness in the model, but it might also illustrate why some countries have a very high number of infections (like for example Brazil and the USA) while other countries have a relatively small number of infections (like Latvia or Mongolia) [3] even when they have implemented the same lockdown rules. It is needless to say that other factors such as the culture of a country is of high influence as well as population density. Countries like Mongolia have a much lower population density compared to countries like for instance USA or China. It already differs from city to city, as the spread in big cities like Shanghai or Tokyo would go a lot faster compared to small villages like for instance Shirakawa-go. However, the variation does make the simulations better reflect reality.

The graphs of the Infected sub-populations also show a discontinuity around time 22τ, after which they decrease almost linearly towards zero. This suggests that the number of active cases has drastically decreased due due to the lockdown rules. However, the other reason of the decrease is that many people have already been infected, hence the number of infected people cannot be as high as before (as we are dealing with a constant population size in this model).
The graphs of the Recovered and Dead sub-populations look like the graph of the infected people, but with the peaks shifted more to the right. The peak of the Recovered sub-population is around time 19τ and the peak of the Dead sub-population is around time 22τ, as it takes some time for an infected person to recover or die. What is also noticeable is that there is also a large variation in the number of recovered people as well as dead people. The number of recovered people ranges between 180 to 400 in a certain time interval and the number of deaths between 200 till 600 people. In reality this might refer to places where there might be a high number of infections, but among a younger population for instance (who have a higher chance of recovery than the elderly) causing a large amount of recovered people and less deaths. In order cases due to bad public health care, there might be more deaths compared to recovered individuals.

The cumulative graphs in Figure 12 show that despite the variability in the number of recoveries or deaths in a certain time window, the total number of casualties is approximately the same for all the runs. The total number of recovered people ranges between 4000-4500 and the total number of deaths between 5000-5500. The trends we observed in Figure 11 are reflected in these graphs as well. Concluding, these results suggest that a lockdown does not influence the total number of casualties at the end, it only spreads the number of active cases as well as recoveries and deaths over a longer period of time.

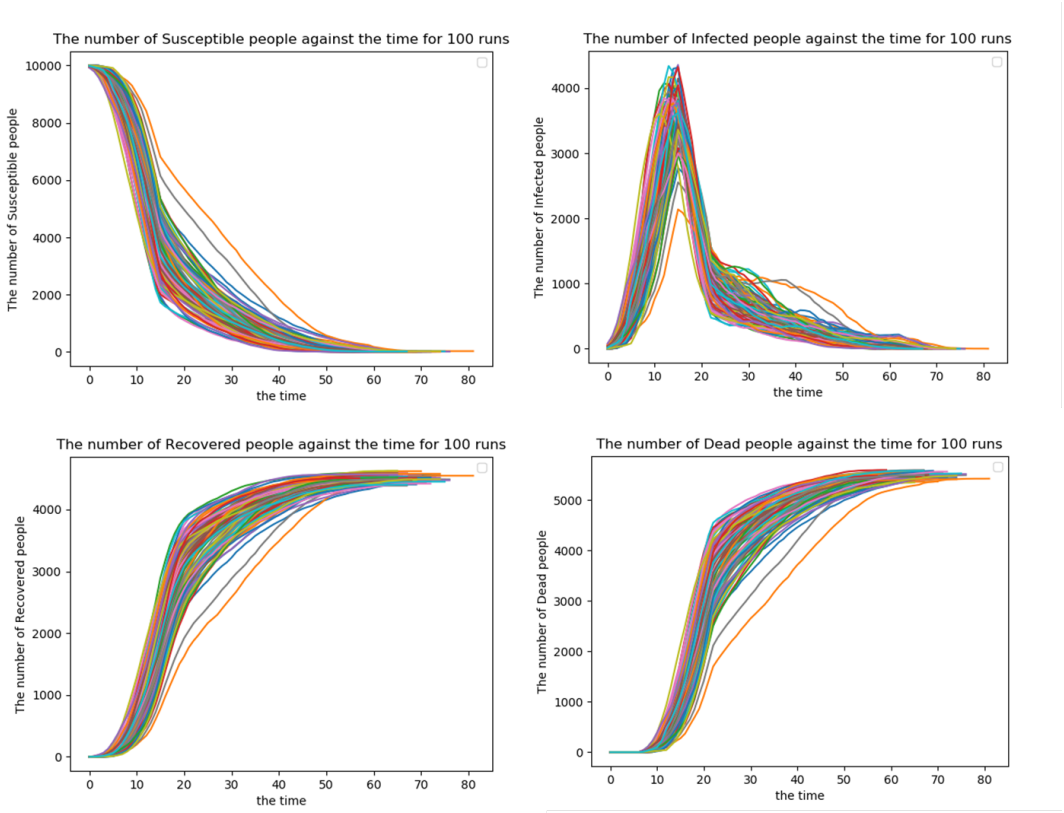

Figure 12: Graph of the number of people in the Susceptible, Infected, Recovered or Dead sub-populations against the time for a medium lockdown of $\beta(t) = 0.5$ where the lockdown starts at time 15τ for 100 simulation runs. The graphs of the Recovered and Dead sub-populations are now cumulative.

4.3 Case 3: A severe lockdown of $\beta(t) = 0.1$

Consider now a severe lockdown, in which the amount of social interaction is reduced to 10%, hence:

$$\beta(t) = \begin{cases} \tilde{\beta}(t) = 0.1, & \text{for } t \in T_{ld} = (15\tau, 60\tau) \\ 1, & \text{else.} \end{cases}$$

This lockdown might refer to the scenario where people are only allowed to be on the streets for one hour a day, all the social events are canceled and bars/restaurants/sports facilities etc. are all closed. The states of the simulation at different times can be seen in Figures 72 and 73 in Appendix 11 as well as their analysis. The graphs of the Susceptible, Infected, Recovered and Dead sub-populations can be found in Figures 13 and 14.

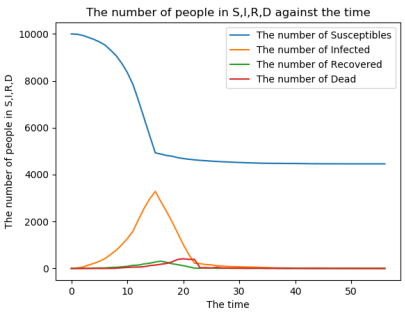

Figure 13: Graph of thesub-populations against the time for a severe lockdown of $\beta(t) = 0.1$ where the lockdown starts at time 15τ and ends at time 80τ.

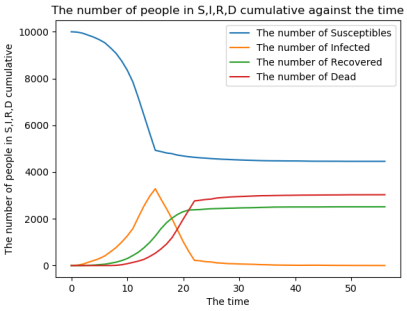

Figure 14: Cumulative graph of the sub-populations against the time for a severe lockdown of $\beta(t) = 0.1$ where the lockdown starts at time 15τ and ends at time $8\tau 0$.

The consequences of the severe lockdown is significant. Until time 15τ the graphs look similar to previous cases, but from from time 15τ on, when the lockdown is implemented there is a clear distinction seen in Figure 13. The most striking observation is that the number of susceptible people, as well infected, recovered and dead people become constant after the lockdown within a short time interval, validating the fact that the virus has been eradicated (this was seen in the simulation in Figures 72 and 73 in Appendix 11. The number of active cases drops drastically after the lockdown, making the graph of the Infected sub-population almost triangular of shape. The cumulative graph of the Recovered and Dead sub-populations in Figure 14 still look like logistic curves just like in the previous

cases and show similar trends. However, the discontinuity in the derivative of the Dead sub-population as a consequence of the severity of the lockdown is clearly present. All of this shows that such a severe lockdown will rapidly decrease the spread of the virus and eventually let the pandemic die out.

4.4 Case 4: $\beta(t) = 0.3$ a heavy lockdown

Consider the case, where the lockdown is less strict compared to the previous case of $\beta(t) = 0.1$, but stricter than the medium lockdown of $\beta(t) = 0.5$. The amount of social interaction is reduced to 30% of the usual amount as follows:

$$\beta(t) = \begin{cases} \tilde{\beta}(t) = 0.3, & \text{for } t \in T_{ld} = (15\tau, 60\tau) \\ 1, & \text{else.} \end{cases}$$

This might refer to the case where all the restaurants/pubs/barbers/sport facilities are closed, all group activities canceled and people are encouraged to stay at home, but there is no curfew nor is there any limitation in going outside. The evolution of the system at different times can be seen in Figures 74 and 75 in the Appendix 11 as well as their analysis.

The first observation that is apparent in the non-cumulative graph of the four different sub-populations in Figure 15 is that a 'second wave' is clearly seen after time 60 when the lockdown has been lifted in the number of active cases. The reason is most likely because there is a sufficient amount of susceptible people left in the population to cause another outbreak. However, the peak is not as high compared to the first spike as there are less susceptible people left to infect. The second outbreak is also observed in the cumulative graphs of the Recovered and Dead sub-populations in Figure 16 as we see a sudden increase in slope in both graphs. The triangular shape in the Infected sub-population graph as well as the discontinuities in the derivatives of all the other graphs is seen back as well due to the lockdown policy. During the lockdown period, we see that the number of active cases stays about constant, resulting in a steady increase in the number of recovered as well as dead people. The simulation finished at time 84τ.

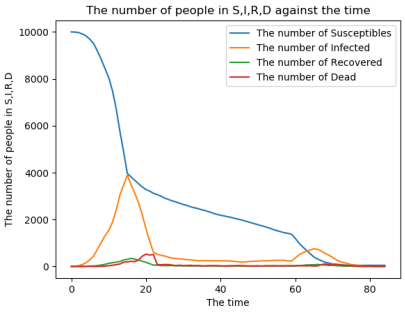

Figure 15: Graph of the sub-populations against the time for a heavy lockdown of $\beta(t) = 0.3$ where the lockdown starts at time 15τ.

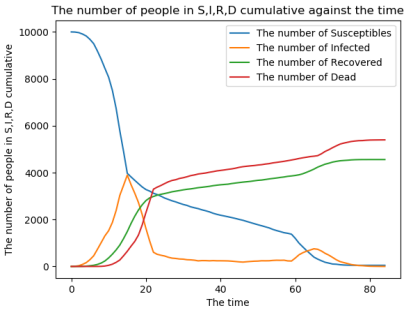

Figure 16: Cumulative graph of the sub-populations against the time for a heavy lockdown of $\beta(t) = 0.3$ where the lockdown starts at time 15τ.

Lockdown of $\beta(t) = 0.3$ until the end

We saw that if the $T_{ld} = (15\tau, 60\tau)$, there was a second peak in the number of active cases after the lockdown is lifted. What might be interesting to consider next, is the case where the lockdown is valid for the entire time period of 100 time steps. One question that may arise could be for instance: *"Will the virus eventually die out? In other words, will there be still a considerably amount of susceptible people left, but no more new infections?"* In order to answer this question, the simulation has been run for the entire time length of 100 time steps. Since the spread of the virus is similar as in figures 74 and 75, only the end state of the system is given in Appendix 11. The results are plotted against the time in Figures 17 and 18, where in the second graph the Recovered and Dead sub-populations have been plotted cumulative against the time.

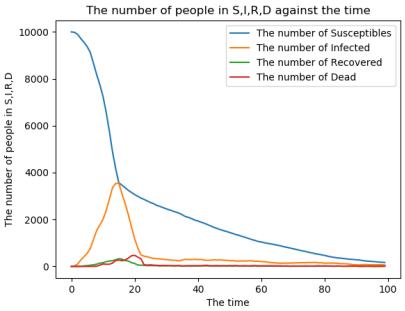

Figure 17: Graph of the sub-populations against the time for a heavy lockdown of $\beta(t) = 0.3$ where the lockdown starts at time 15τ and is valid until time 100τ.

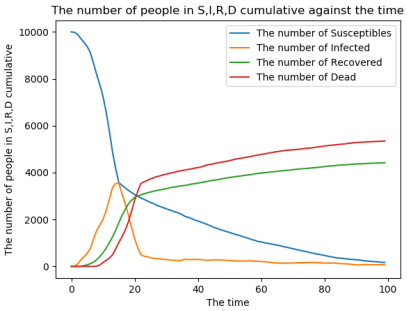

Figure 18: Cumulative graph of the sub-populations against the time for a heavy lockdown of $\beta(t) = 0.3$ where the lockdown starts at time 15τ and is valid until time 100τ.

We see that the virus has not stopped spreading in Figure 76. If we compare Figure 17 with Figure 15 we see that the graphs look very similar until time 60τ. In fact, extending the lockdown over a longer time period continues the steady number of active cases as well as steady decrease and rise in the number of susceptible and casualties respectively. This is seen by the steady slopes in the graphs. Furthermore, if we compare the Figures 16 and 18, we see the same trends. There is no second peak and due to the lockdown the number of recovered and dead people increases steadily over time. If the simulation would have lasted longer than 100τ, all the active cases would have recovered or died at the end.

4.5 Case 5: $\beta(t) = 0.7$ a mild lockdown

The last lockdown we are going to consider is a mild lockdown of $\beta(t) = 0.7$.

$$\beta(t) = \begin{cases} \tilde{\beta}(t) = 0.7, & \text{for } t \in T_{ld} = (15\tau, 60\tau) \\ 1, & \text{else.} \end{cases}$$

This might refer to a 1.5 metre society, with schools, shops and all public places open. Small social gatherings are allowed, but large events are still forbidden. Many countries are currently in this mild lockdown. The states of the system at different time steps is given in Figure 77 and Figure 78 in the Appendix 11 as well as their analysis. The results are plotted in Figures 19 and 20.

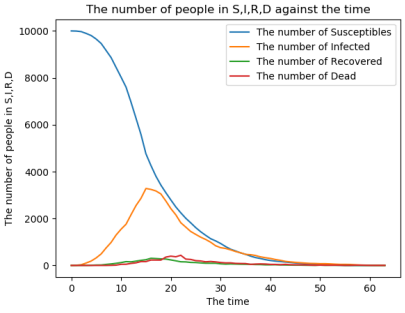

Figure 19: Graph sub-populations against the time for a mild lockdown of $\beta(t) = 0.7$ where the lockdown starts at time 15τ and ends at 60τ.

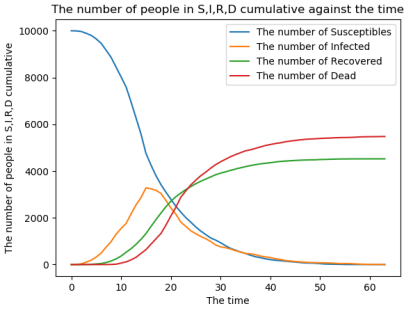

Figure 20: Cumulative graph of the sub-populations against the time for a mild lockdown of $\beta(t) = 0.7$ where the lockdown starts at time 15τ and ends at 60τ.

Surprisingly, Figures 19 and 20 look very similar to Figures 5 and 6, suggesting that the effect of such lockdown is minimal. The only difference is that in the Infected, Recovered and Dead sub-populations the graphs there are some discontinuities in the derivatives of the graphs. However, it is not as extreme as in the cases where $\beta(t) = 0.1$ and $\beta(t) = 0.3$. At 15τ we some little changes in the slope, but the effect is not significant.

The cumulative graphs of the Recovered and Dead sub-populations in Figure 20 and Figure 6 look almost identical, implying that the Recovered and Dead sub-populations are not really affected by the mild lockdown. The only difference is that the time period of the epidemic is stretched over a longer period of time. The number of casualties remains the same in the end. All in all, the graphs show that a mild lockdown has hardly any effect except for stretching the time period of the epidemic.

4.6 Lifting the lockdown rules

In this section some simulations will be shown where the lockdown is not kept constant over time or lifted immediately, but there is a step in between. We will consider a case where we first implement a severe lockdown of $\beta(t) = 0.1$, then loosen the rules to $\beta(t) = 0.5$ and then to $\beta(t) = 1$, which is no lockdown. In addition, we will also consider a case where first a heavy lockdown of $\beta(t) = 0.3$ is implemented and then lifted to a lockdown of $\beta(t) = 0.6$ or $\beta(t) = 0.7$ to what impact such lockdown has.
The parameters that are used in the simulations are found in Table 4.

Parameter	**Value**
grid $(n_x \times n_y)$	100×100
τ	1
λ_g	0.5
μ	0.1
end time	100τ
time lockdown 1 start	15τ
time lockdown 2 start	35τ
time lockdown end	60τ
Time to death T_d	8τ

Table 4: Table of parameters used in the simulations

4.6.1 Case 1: severe lockdown of $\beta(t) = 0.1$, then medium lockdown of $\beta(t) = 0.5$ followed by no lockdown of $\beta(t) = 1$

First of all, we start by examining a simulation where first a severe lockdown of $\beta(t) = 0.1$ has been implemented. This lockdown is then later lifted to a medium lockdown of $\beta(t) = 0.5$ at time 35 and at time 60 the lockdown is entirely lifted. Therefore:

$$\beta(t) = \begin{cases} \tilde{\beta}(t) = 0.1, & \text{for } t \in T_{ld_1} = (15\tau, 35\tau) \\ \tilde{\beta}(t) = 0.5, & \text{for } t \in T_{ld_2} = (35\tau, 60\tau) \\ 1, & \text{else.} \end{cases}$$

This could refer to the case where a country has first implemented a strict protocol where people are only allowed to go outside for one hour a day and all public restaurants/events/bars/shops are closed. Then the rules are lifted to a medium lockdown where shops are open again but only for limited amount of customers and everybody has to wear a face mask, and at time 60 everything is back to normal again.

The end state of the simulation can be seen in Figure 79 in the Appendix 11. From this figure it can be seen that at time 100, when the simulation ended, there is still a significant amount of susceptible people left as well as infected people. This indicates that the virus is still spreading amongst the population.
In order to understand what consequences this lockdown implementation has brought to the population, we consider the graphs of the various sub-populations in Figures 21 and 22.

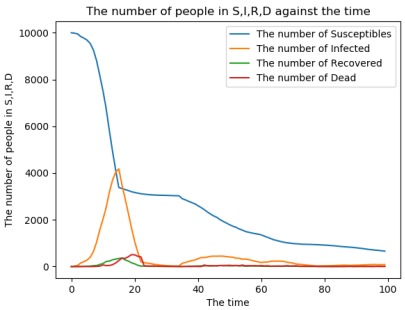

Figure 21: Graph of the sub-populations against the time. The lockdown changes from $\beta(t) = 0.1$ at time 15τ , to $\beta(t) = 0.5$ at time 35τ to $\beta(t) = 1.0$ at time 60τ.

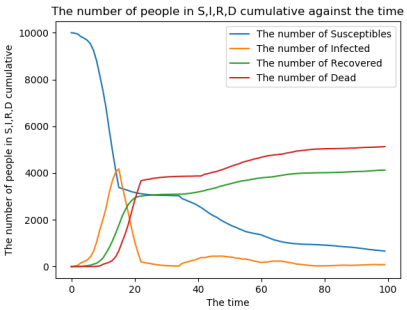

Figure 22: Cumulative graph of the sub-populations against the time. The lockdown changes from $\beta(t) = 0.1$ at time 15τ , to $\beta(t) = 0.5$ at time 35τ to $\beta(t) = 1.0$ at time 60τ.

From Figure 21 we see some interesting events as a result of the lift of the lockdown rules. Until time 15τ everything is like before as the virus is free to spread. At time 15τ, we see the consequence of the severe lockdown of $\beta(t) = 0.1$ (just like in Case 3). Due to the relaxation of the lockdown at time 35τ, a second outbreak occurs causing a rise in the number of infections. However, the number of susceptible people still decreases at a relatively constant rate, until time 60τ. At time 60τ the lockdown is lifted and we notice a third peak in the number of active cases. The second and third peak are nevertheless relatively small compared to the first peak. This is not only a result of the lockdown rules, but also a result of less susceptible people in the population.In Figure 22 the effect of the change in lockdown in the Recovered and Dead sub-population graphs is more clear. The most striking observation in these graphs is that the third peak is not really visible, suggesting that the relaxation of a medium lockdown to no lockdown doesn't affect the number of recovered or dead people significantly.

4.6.2 Case 2: heavy lockdown of $\beta(t) = 0.3$ to mild lockdown of $\beta(t) = 0.6$ and then no lockdown

The next scenario is as follows: a country has first implemented a heavy lockdown of $\beta(t) = 0.3$, then relaxed the rules to a mild lockdown of $\beta(t) = 0.6$ and after that the lockdown is entirely lifted.

$$\beta(t) = \begin{cases} \tilde{\beta}(t) = 0.3, & \text{for } t \in T_{ld_1} = (15\tau, 35\tau) \\ \tilde{\beta}(t) = 0.6, & \text{for } t \in T_{ld_2} = (35\tau, 60\tau) \\ 1, & \text{else.} \end{cases}$$

The end state of this simulation can be found in Figure 80 in the Appendix 11. As can be seen from the Figure 80 the simulation stopped running at time 79, which means that at this time there were no active cases left. To see how the virus has spread amongst the population we consider the graphs of the various sub-populations in Figures 23 and 24.

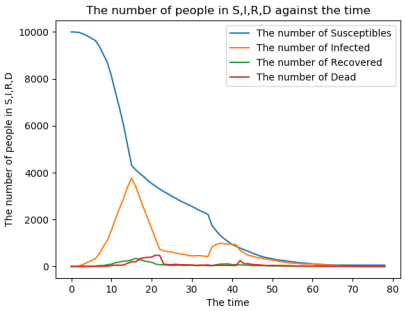

Figure 23: Graph of the sub-populations against the time. The lockdown changes from $\beta(t) = 0.3$ at time 15τ , to $\beta(t) = 0.6$ at time 35τ to $\beta(t) = 1.0$ at time 60τ.

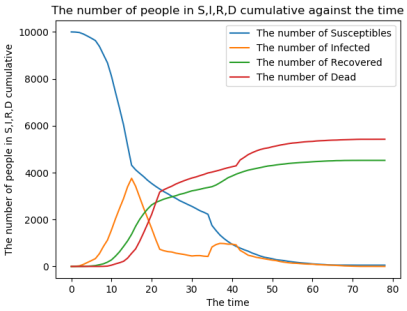

Figure 24: Cumulative graph of the sub-populations against the time. The lockdown changes from $\beta(t) = 0.3$ at time 15τ , to $\beta(t) = 0.6$ at time 35τ to $\beta(t) = 1.0$ at time 60τ.

Figure 23 shows two clear peaks in the Infected, Recovered and Dead sub-population graphs as a result of the relaxation of the lockdown rules. The effect of the first lockdown of $\beta(t) = 0.3$ is significantly present as can be seen in the number of active cases. The loosening of the rules at time 35τ causes another rise in the number of infections from time 35τ until time 42τ. It is quite interesting to see that the first peak looks almost triangular while the second peak looks like half an oval (shape wise). This is probably due to the fact that the first peak is caused by no lockdown rules, so we first have an exponential growth of infected people. Then suddenly a heavy lockdown is implemented which causes the amount of infected people to drop drastically. The second lockdown is mild, which causes a rise in the number of active cases, but since there are few susceptible people remaining in the population, it quickly turns and decreases again. The cumulative graphs in Figure 24 show the effects of the variation in the lockdown rules on the total number of recovered and dead people at different time steps. The peaks in the number of active cases correspond to the rapid rises in these graphs.

In addition, another simulation has been run, but in this case, the second lockdown was lifted till 0.7 instead of 0.6 to see if this makes any difference. The results in Figures 25 and 26 show that the difference is not significant. The same trends observed in Figures 23 and 24 are seen.

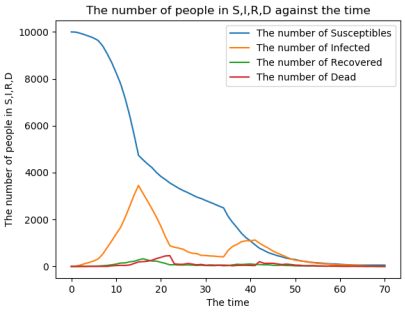

Figure 25: Graph of the sub-populations against the time. The lockdown changes from $\beta(t) = 0.3$ at time 15τ , to $\beta(t) = 0.7$ at time 35τ to $\beta(t) = 1.0$ at time 60τ.

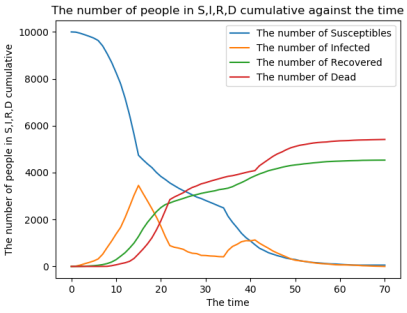

Figure 26: Cumulative graph of the sub-populations against the time. The lockdown changes from $\beta(t) = 0.3$ at time 15τ , to $\beta(t) = 0.7$ at time 35τ to $\beta(t) = 1.0$ at time 60τ.

5 Varying the parameters λ_g and μ

We are going to vary the pairs of the parameters (λ_g, μ) and see what effects the variation has on the simulation results. We consider the end time of the simulation (when there are no more infected people left), the maximum number of active cases at a certain time interval and number of people who were died, recovered or were still susceptible at the end of each simulation. In Table 5 the parameters that are used in the simulations are stated.
Examine the case when lockdown is implemented, so the government of a country has for instance not taken any actions yet against the spread of the COVID-19 virus. We vary the overall infection probability rate λ_g as well as the recovery rate parameter μ, for every pair (λ_g, μ) a hundred simulations are done and the average of the following values is calculated:

- the end time of the simulation
- the number of susceptible people at the end of the simulation
- the number of recovered people at the end of the simulation
- the number of dead people at the end of the simulation
- the maximum number of infected people at the end of the simulation

Parameter	**Value**
grid $(n_x \times n_y)$	100×100
τ	1
end time	100τ
Time to death T_d	8τ

Table 5: Table of parameters used in the simulations

To see the effects of the variation in the infection probability rate parameter λ_g and the recovery rate parameter μ, we have plotted the average of the values mentioned above in colour against the parameter values in Figures 27, 28 29 30 and 31.

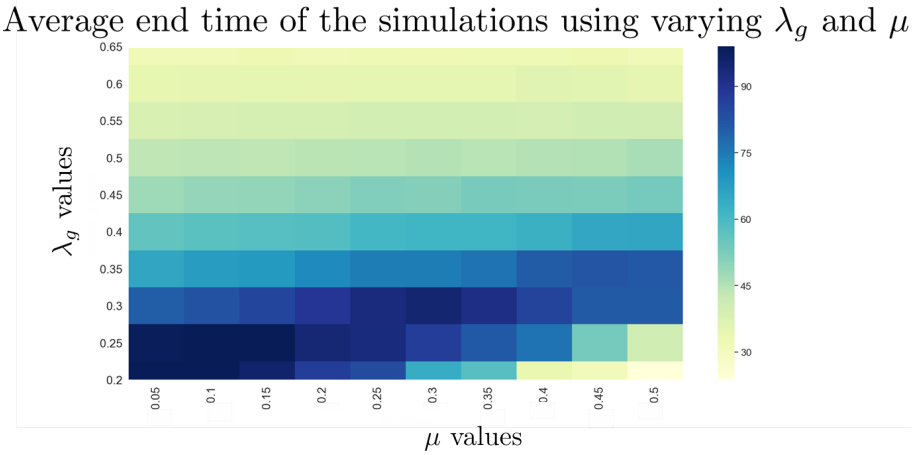

Figure 27: The average end time of the simulations using different values of λ_g and μ.

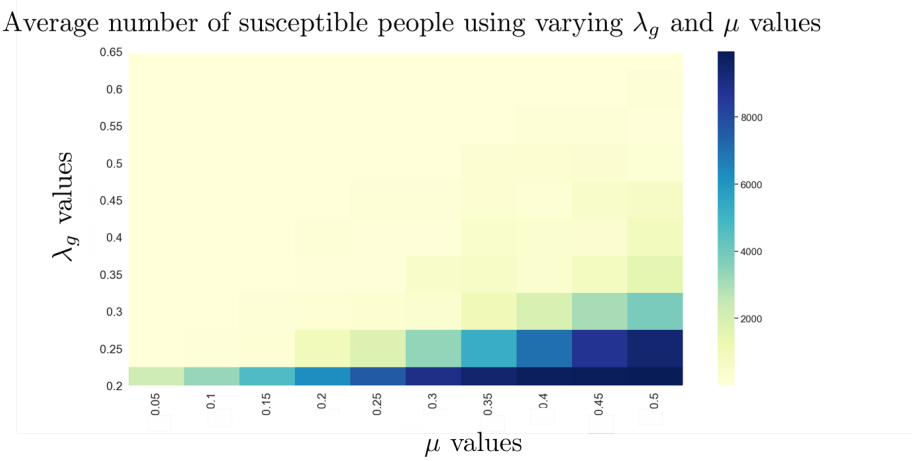

Figure 28: The average number of susceptible people at the end of the simulations using different values of λ_g and μ.

Figure 27 shows that the average end time of the simulations is largest when both the parameter values of λ_g and μ are smallest. Another observation that stands out is that the higher the μ value and the lower the λ_g value, the faster there are no more active cases, hence the lower the average end time of the simulation. This can be explained by the way the parameters have been defined: the lower the infection probability rate, the smaller the probability of getting infected by the virus and the higher the recovery rate, the higher the probability of recovery. Another observation is that for large values of λ_g (above 0.4), the average end time of the simulation is approximately the same for various values of μ. This suggests

that the recovery rate is not of big influence on the time it takes until there are no more infected people left when the infection probability rate is large. Only with lower infection probability rates, the recovery rate μ has an effect. This is logical as λ_g determines the probability of getting infected, hence the number of active cases.
If we turn to the average number of susceptible people for different values of λ_g and μ, we see that the results in Figure 27 are not entirely reflected in Figure 28. A surprising result is that the average number of susceptible people is for most of the pairs (λ_g, μ) about the same size: all under the 1000 people. Only, when the value of λ_g is less than 0.3 and simultaneously μ is more than 0.3, we see that the average number of susceptible people in the population rapidly increases. These results suggest that if the infection probability rate is low and the recovery rate is high then this results in a large number of susceptible people on average. This is in line with Figure 27 because if there are more susceptible people in the population who do not get infected, it takes less time for the number of infected cases to become zero.

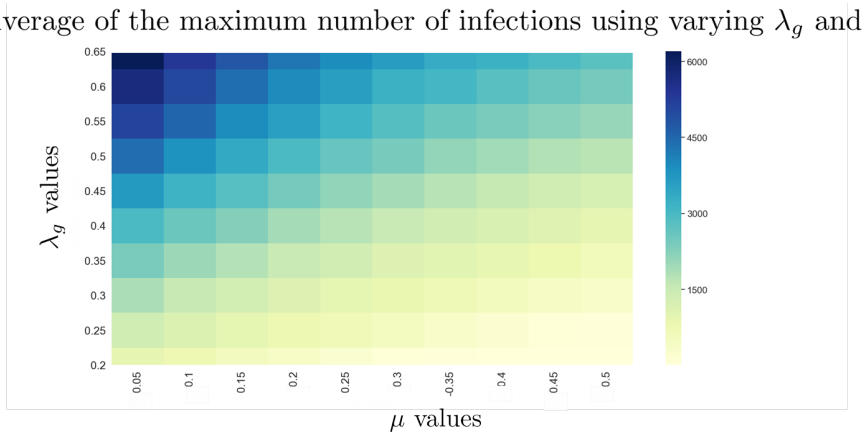

Figure 29: The average of the maximum number of infected people in the simulations using different values of λ_g and μ.

Further analysis of Figure 29 shows that the average maximum number of infections is in line with what has been observed in Figure 28. The maximum number of infections is smallest when the value of λ_g is small and μ is large. There is a increasing trend in the maximum number of infections when λ_g becomes larger and μ becomes smaller. The highest number of infections is when λ_g is largest and μ is smallest, which is what can be expected.
Figure 30 shows that average number of recovered people in the simulations. There is a clear trend of an increasing number of recovered individuals when the values of λ_g and μ both become larger. The lowest number of recovered people is surpris-

ingly seen when μ is largest (0.5) and λ_g is smallest (0.2). This phenomenon can be explained by Figure 29 and Figure 28. In Figure 28 the number of susceptible people were largest for these values of the parameters, implying that there is a smaller amount of people who got infected, which is seen in Figure 29 as well. Since there are fewer people who got infected, there is a smaller amount of recovered people, hence the result in Figure 30. Closer inspection of Figure 30 shows that for a fixed value of μ and a varying λ_g value above 0.3, the average number of recovered people appears to be approximately constant. This suggests that the value of μ determines the average number of recovered people regardless of the value of λ_g (if the value of λ_g is lower than 0.3), which is not surprising as μ is the recovery rate.

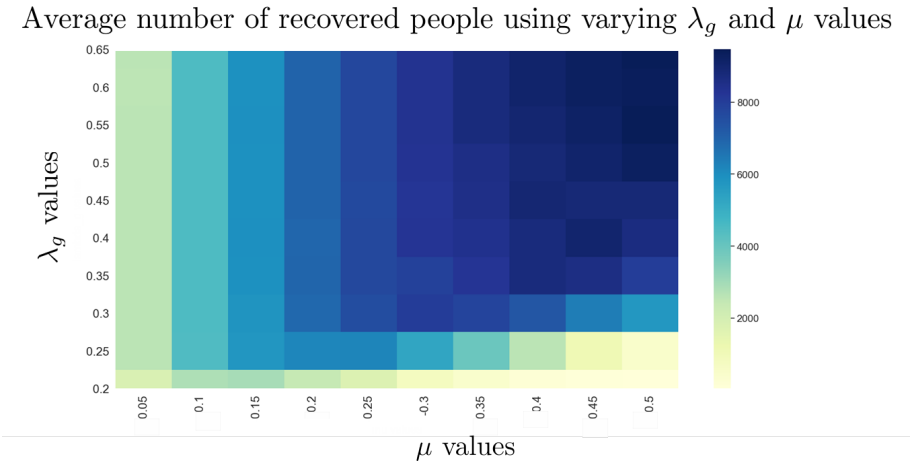

Figure 30: The average number of recovered people at the end of the simulations using different values of λ_g and μ.

Finally, the average number of dead people is shown in Figure 31. Here we see that the largest amounts of deaths happen when μ is small, regardless of the value of λ_g. Similar to figure 30, the influence of the value of μ is clearly present. It can be seen that the number of dead people is approximately constant for fixed values of μ despite the values of λ_g, when λ_g is larger than 0.3. The lowest amount of deaths is seen when λ_g is smallest (0.2) and μ is largest (0.5). The reason why is the same as for the number of recovered people explained earlier.

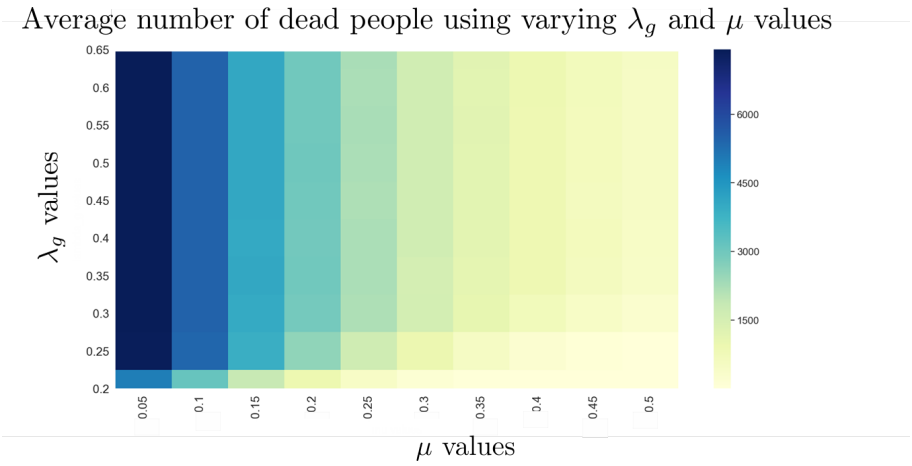

Figure 31: The average number of dead people at the end of the simulations using different values of λ_g and μ.

Together these results provide important insights into the effects of the infection probability rate λ_g and the recovery rate μ on the average length of the epidemic, but also on the amount of susceptible, infected, recovered and dead people.

6 Parameter Estimation

The set up for parameter estimation for the mathematical model in this thesis is explained in this chapter.

6.1 Likelihood and Probabilities

We consider likelihood based inference and therefore derive the likelihood function. Suppose we are dealing with a constant population of n people, where every person is indexed by $i = 1, \ldots, n$. Consider the time intervals $t_0, t_1, \ldots, t_K$ where we observe the state of each person i at every time interval t_k. The time intervals are indexed by $k = 0, 1, \ldots, K$ and we have a total of $K + 1$ time intervals. Let $x_{i,k}$ be the observed state of person i on time t_k. Denote $\mathbf{Y_k} = (x_{1k}, x_{2k}, \ldots, x_{nk})^T$ as the vector consisting of the states of all the n people on time t_k.
Next, since the state of an individual is also dependent on the amount of time spent in the infected state (as we assumed that if a person has spent M time steps in the infected state, he/she dies), it is also necessary to define a vector $\mathbf{Z_k}$. This is a vector that counts the amount of time intervals τ that person i is in the infected state at time t_k. The vector is denoted by: $\mathbf{Z_k} = (z_{1k}, z_{2k}, \ldots, z_{nk})^T$. For every entry $z_{i,k}$, we have the following: $z_{i,k} = \sum_{l=0}^{k} \mathbb{1}_{\{x_{i,l}=2(\text{infected})\}}$. The indicator function essentially counts the number of time intervals that individual i is in the infected state.
The assumption that was given in the mathematical model was that the time to death is given by: $T_d = M \cdot \tau$, where M is a 'large' integer number and τ is the time step. As soon as the person is longer infected than T_d, he/she is assumed to be dead.
Every person's state is in the set {S,I,R,D} or equivalently {1,2,3,4} where 1 represents susceptible, 2 infected, 3 recovered and 4 dead. We will call the set $\{1, 2, 3, 4\} = \Sigma$. Then all the vectors $\mathbf{Y_k} \in \{1, 2, 3, 4\}^n = \Sigma^n$. All the vectors $\mathbf{Z_k}$ are in the set $\{0, 1, \ldots, M\}^n$.
Now that the notation has been set up, Let us define the likelihood function which is going to depend on both vectors $\mathbf{Y_k}$ and $\mathbf{Z_k}$. Since we assumed that the state of the system on time t_k (so the state of all the n people on t_k) is only dependent on the state of the system on time t_{k-1}, we can write the following (using the definition of conditional probabilities):

$$
\begin{aligned}
&\mathbb{P}((\mathbf{Y_0}, \mathbf{Z_0}) = (\mathbf{y_0}, \mathbf{z_0}), (\mathbf{Y_1}, \mathbf{Z_1}) = (\mathbf{y_1}, \mathbf{z_1}), \ldots, (\mathbf{Y_K}, \mathbf{Z_K}) = (\mathbf{y_K}, \mathbf{z_K})) \\
&= \mathbb{P}((\mathbf{Y_K}, \mathbf{Z_K}) = (\mathbf{y_K}, \mathbf{z_K}) | (\mathbf{Y_0}, \mathbf{Z_0}) = (\mathbf{y_0}, \mathbf{z_0}), \ldots, (\mathbf{Y_{k-1}}, \mathbf{Z_{k-1}}) = (\mathbf{y_{k-1}}, \mathbf{z_{k-1}})) \\
&\cdot \mathbb{P}((\mathbf{Y_{k-1}}, \mathbf{Z_{k-1}}) = (\mathbf{y_{k-1}}, \mathbf{z_{k-1}}) | (\mathbf{Y_0}, \mathbf{Z_0}) = (\mathbf{y_0}, \mathbf{z_0}), \ldots, (\mathbf{Y_{K-2}}, \mathbf{Z_{K-2}}) = (\mathbf{y_{K-2}}, \mathbf{z_{K-2}}) \cdot \ldots \\
&\cdot \mathbb{P}((\mathbf{Y_1}, \mathbf{Z_1}) = (\mathbf{y_1}, \mathbf{z_1}) | (\mathbf{Y_0}, \mathbf{Z_0}) = (\mathbf{y_0}, \mathbf{z_0})) \cdot \mathbb{P}((\mathbf{Y_0}, \mathbf{Z_0}) = (\mathbf{y_0}, \mathbf{z_0}))
\end{aligned}
$$

Due to the Markov property this becomes:

$$\mathbb{P}((\mathbf{Y_0}, \mathbf{Z_0}) = (\mathbf{y_0}, \mathbf{z_0}), (\mathbf{Y_1}, \mathbf{Z_1}) = (\mathbf{y_1}, \mathbf{z_1}), \ldots, (\mathbf{Y_K}, \mathbf{Z_K}) = (\mathbf{y_K}, \mathbf{z_K}))$$
$$= \mathbb{P}((\mathbf{Y_K}, \mathbf{Z_K}) = (\mathbf{y_K}, \mathbf{z_K}) | (\mathbf{Y_{K-1}}, \mathbf{Z_{K-1}}) = (\mathbf{y_{K-1}}, \mathbf{z_{K-1}})) \cdot \mathbb{P}((\mathbf{Y_{K-1}}, \mathbf{Z_{K-1}} = (\mathbf{y_{K-1}}, \mathbf{z_{K-1}})) | (\mathbf{Y_{K-}}$$
$$\cdot \mathbb{P}((\mathbf{Y_1}, \mathbf{Z_1}) = (\mathbf{y_1}, \mathbf{z_1}) | (\mathbf{Y_0}, \mathbf{Z_0}) = (\mathbf{y_0}, \mathbf{z_0})) \cdot \mathbb{P}((\mathbf{Y_0}, \mathbf{Z_0}) = (\mathbf{y_0}, \mathbf{z_0}))$$

which we can write in short hand notation:

$$\mathbb{P}((\mathbf{Y_0}, \mathbf{Z_0}) = (\mathbf{y_0}, \mathbf{z_0}), (\mathbf{Y_1}, \mathbf{Z_1}) = (\mathbf{y_1}, \mathbf{z_1}), \ldots, (\mathbf{Y_K}, \mathbf{Z_K}) = (\mathbf{y_K}, \mathbf{z_K}))$$
$$= \mathbb{P}((\mathbf{Y_0}, \mathbf{Z_0}) = (\mathbf{y_0}, \mathbf{z_0})) \prod_{k=1}^{K} \mathbb{P}((\mathbf{Y_K}, \mathbf{Z_K}) = (\mathbf{y_K}, \mathbf{z_K}) | (\mathbf{Y_{k-1}}, \mathbf{Z_{k-1}}) = (\mathbf{y_{k-1}}, \mathbf{z_{k-1}})).$$

Since every individual is conditionally independent from the other individuals this can be factorised into:

$$\mathbb{P}((\mathbf{Y_0}, \mathbf{Z_0}) = (\mathbf{y_0}, \mathbf{z_0}), (\mathbf{Y_1}, \mathbf{Z_1}) = (\mathbf{y_1}, \mathbf{z_1}), \ldots, (\mathbf{Y_K}, \mathbf{Z_K}) = (\mathbf{y_K}, \mathbf{z_K}))$$
$$= \mathbb{P}((\mathbf{Y_0}, \mathbf{Z_0}) = (\mathbf{y_0}, \mathbf{z_0})) \prod_{k=1}^{K} \prod_{i=1}^{n} \mathbb{P}((X_{i,k}, Z_{i,k} = (x_{i,k}, z_{i,k}) | (\mathbf{Y_{k-1}}, \mathbf{Z_{k-1}}) = (\mathbf{y_{k-1}}, \mathbf{z_{k-1}}))$$

When we are dealing with likelihoods, it is common to drop the term $\mathbb{P}((\mathbf{Y_0}, \mathbf{Z_0}) = (\mathbf{y_0}, \mathbf{z_0}))$ (as it is given initial information).
Let us consider the probability $\mathbb{P}((X_{i,k}, Z_{i,k} = (x_{i,k}, z_{i,k}) | (\mathbf{Y_{k-1}}, \mathbf{Z_{k-1}}) = (\mathbf{y_{k-1}}, \mathbf{z_{k-1}})))$ separately. There are two possible cases where the person $X_{i,k-1}$ can be in:

- $X_{i,k-1} = S$ (or equivalently 1) , which means that at time t_{k-1} person i is susceptible.
- $X_{i,k-1} = I$ (or equivalently 2), which means that at time t_{k-1} person i is infected.

Case 1: $X_{i,k-1} = S$ person i is susceptible

There are two possibilities: either he/she stays susceptible or becomes infected in each time step τ (similar to a Bernoulli Random variable, see the Appendix 12 for the definition of a Bernoulli Random variable).

So at time t_k:

$X_{i,k} = S$ with probability $e^{-\lambda_g \cdot (\sum_{j \in N_i^{Inf}(t)} a_{ij}(t)) \cdot \tau}$

$X_{i,k} = I$ with probability $1 - e^{-\lambda_g \cdot (\sum_{j \in N_i^{Inf}(t)} a_{ij}(t)) \cdot \tau}$

For clarity purposes the probability is denoted as follows:

$$p = 1 - e^{-\lambda_g \cdot (\sum_{j \in N_i^{Inf}(t)} a_{ij}(t)) \cdot \tau}$$

The transition is represented in diagram 32.

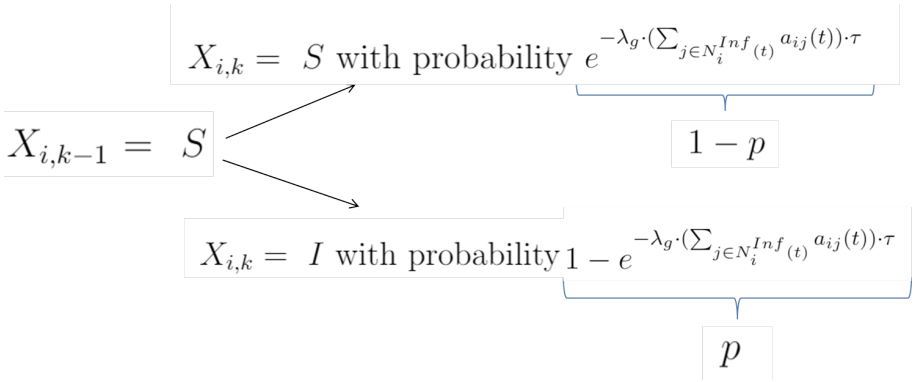

Figure 32: Diagram of the possible transition for a susceptible person.

Therefore, the transition probability from the susceptible state to the infected or susceptible state is given by:

$$\begin{aligned} &\mathbb{P}_{i,k-1}^{S \to \{I,S\}}((X_{i,k}, Z_{i,k} = (x_{i,k}, z_{i,k}) | (\mathbf{Y_{k-1}}, \mathbf{Z_{k-1}}) = (\mathbf{y_{k-1}}, \mathbf{z_{k-1}})) \\ &= p \cdot \mathbb{1}_{\{x_{i,k-1}=S, x_{i,k}=I\}} + (1-p) \cdot \mathbb{1}_{\{x_{i,k-1}=S, x_{i,k}=S\}} \end{aligned} \tag{24}$$

Note that p is dependent on time and the number of infected neighbours and is different for each person on the grid.

Case 2: $X_{i,k-1} = I$ **person** i **is infected**

Next, consider the case where person i is infected at time t_{k-1}. Now there are three possible states that this person can transition to in the next time step: either he/she stays infected or he/she recovers or he/she dies. It all depends on how many time-steps τ the person has been infected. However, if the person has been infected for $M \times \tau$ the probability of dying is one. Therefore, this is again a Bernoulli random variable in the following way:
In each time step τ, if $z_{i,k} \leq M$:

$X_{i,k} = I$ with probability $e^{-\mu \cdot \tau}$
$X_{i,k} = R$ with probability $1 - e^{-\mu \cdot \tau}$

For clarity purposes we can denote the probability as follows:

$$q = 1 - e^{-\mu \cdot \tau}$$

And if $z_{i,k} > M$: $X_{i,k} = D$ with probability 1.
Then the transition diagram is as in Figure 33 and in Figure 34.

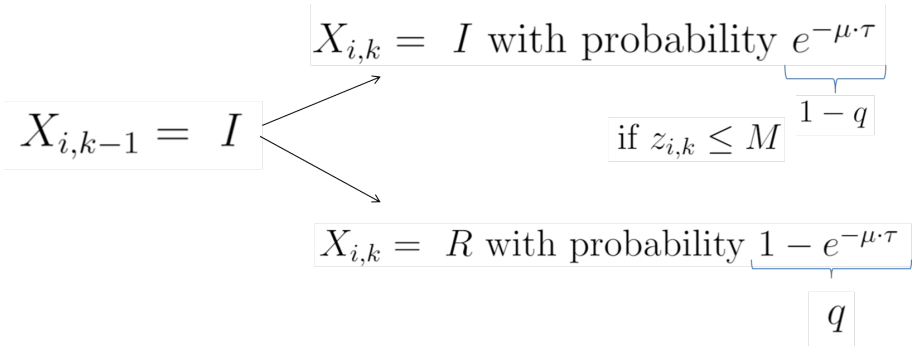

Figure 33: Diagram of the possible transition for an infected person, if $z_{i,k} \leq M$

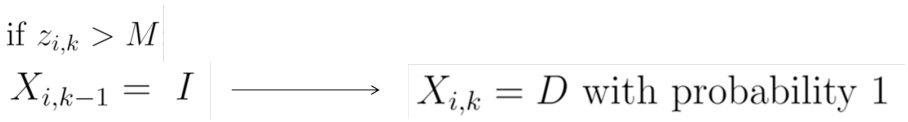

Figure 34: Diagram of the possible transition for an infected person, if $z_{i,k} > M$

Therefore, the transition probability from the infected state to the infected, recovered or dead state is given by:

$$\begin{aligned} &\mathbb{P}_{i,k-1}^{I \to \{I,R,D\}}((X_{i,k}, Z_{i,k} = (x_{i,k}, z_{i,k}) | (\mathbf{Y_{k-1}}, \mathbf{Z_{k-1}}) = (\mathbf{y_{k-1}}, \mathbf{z_{k-1}})) \\ &= q \cdot \mathbb{1}_{\{x_{i,k-1}=I, x_{i,k}=R, z_{i,k} \leq M\}} + (1-q) \cdot \mathbb{1}_{\{x_{i,k-1}=I, x_{i,k}=I, z_{i,k} \leq M\}} + \mathbb{1}_{\{z_{i,k} > M\}} \end{aligned} \tag{25}$$

Hence, every time each person is in either of these two cases depending on the current state at time t_{k-1}.

In summary:

$$
\begin{aligned}
&\mathbb{P}((X_{i,k}, Z_{i,k} = (x_{i,k}, z_{i,k})|(\mathbf{Y_{k-1}}, \mathbf{Z_{k-1}}) = (\mathbf{y_{k-1}}, \mathbf{z_{k-1}}) \\
&= \mathbb{P}_{i,k-1}^{S\to\{I,S\}}((X_{i,k}, Z_{i,k} = (x_{i,k}, z_{i,k})|(\mathbf{Y_{k-1}}, \mathbf{Z_{k-1}}) = (\mathbf{y_{k-1}}, \mathbf{z_{k-1}}))+ \\
&\mathbb{P}_{i,k-1}^{I\to\{I,R,D\}}((X_{i,k}, Z_{i,k} = (x_{i,k}, z_{i,k})|(\mathbf{Y_{k-1}}, \mathbf{Z_{k-1}}) = (\mathbf{y_{k-1}}, \mathbf{z_{k-1}})) \\
&= p \cdot \mathbb{1}_{\{x_{i,k-1}=S, x_{i,k}=I\}} + (1-p) \cdot \mathbb{1}_{\{x_{i,k-1}=S, x_{i,k}=S\}}+ \\
&q \cdot \mathbb{1}_{\{x_{i,k-1}=I, x_{i,k}=R, z_{i,k}\leq M\}} + (1-q) \cdot \mathbb{1}_{\{x_{i,k-1}=I, x_{i,k}=I, z_{i,k}\leq M\}} + \mathbb{1}_{\{z_{i,k}>M\}}
\end{aligned}
\tag{26}
$$

A diagram summarizing the possible transitions transitions can be seen in Figure 35.

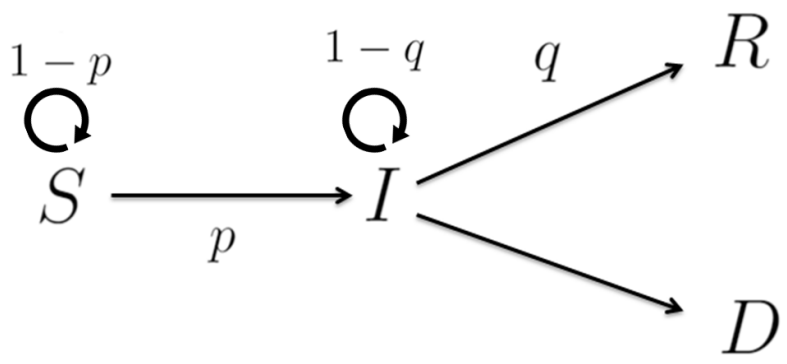

Figure 35: Diagram of the possible transitions.

6.2 The Log-likelihood

For parameter estimation it is often more useful to consider the log-likelihood function instead of the likelihood. This is because all the probabilities are between 0 and 1 and since we are taking the product of these small values, it is better to consider the log of them. Since the log is a convex function, maximizing the likelihood function is the same as maximizing the log-likelihood function. Also, as we are dealing with products of exponential expressions, the expressions become easier when considering the natural logarithm of them. The log-likelihood function is given as follows:
(Note that we omit the pair $(\mathbf{Y_0}, \mathbf{Z_0})$ as that is given by the initial condition).

$$\begin{aligned}
&\log(L(\lambda, \mu; \mathbf{y_1}, \dots, \hat{\mathbf{y}}_\mathbf{K}, \mathbf{z_1}, \dots, \mathbf{z_K})) \\
&= \log(\prod_{k=1}^{K} \mathbb{P}((\mathbf{Y_K}, \mathbf{Z_K}) = (\mathbf{y_K}, \mathbf{z_K}) | (\mathbf{Y_{k-1}}, \mathbf{Z_{k-1}}) = (\mathbf{y_{k-1}}, \mathbf{z_{k-1}}))) \\
&= \sum_{k=1}^{K} \log(\mathbb{P}((\mathbf{Y_K}, \mathbf{Z_K}) = (\mathbf{y_K}, \mathbf{z_K}) | (\mathbf{Y_{k-1}}, \mathbf{Z_{k-1}}) = (\mathbf{y_{k-1}}, \mathbf{z_{k-1}}))) \\
&= \sum_{k=1}^{K} \log(\prod_{i=1}^{n} \mathbb{P}((X_{i,k}, Z_{i,k}) = (x_{i,k}, z_{i,k}) | (\mathbf{Y_{k-1}}, \mathbf{Z_{k-1}}) = (\mathbf{y_{k-1}}, \mathbf{z_{k-1}}))) \\
&= \sum_{k=1}^{K} \sum_{i=1}^{n} \log(\mathbb{P}((X_{i,k}, Z_{i,k}) = (x_{i,k}, z_{i,k}) | (\mathbf{Y_{k-1}}, \mathbf{Z_{k-1}}) = (\mathbf{y_{k-1}}, \mathbf{z_{k-1}})))
\end{aligned} \tag{27}$$

Now it depends on which state the person is in, what the probability is. The probability inside the log is given by equation (26).

7 Maximum Likelihood Estimation

One of the ways to do parameter estimation is by using maximum likelihood estimation. This is a frequentist way of estimating unknown parameters in a model. The maximum likelihood estimator finds the parameter (vector) that gives the maximum probability of observing a particular data $\mathbf{x_0}$ over all the possible values of the parameters. The parameter (vector) that maximizes the likelihood function is called the maximum likelihood estimate. So the maximum likelihood estimate is $\Theta_{MLE} = argmax_\theta L(\theta; \mathbf{X})$, where θ is the parameter (vector), L is the likelihood function and $\mathbf{X}$ is the given data.
Sometimes it is easier to maximize the log of the likelihood when the likelihood has for instance a lot of expressions with exponentials. The log is the natural logarithm (ln) in this case. This is valid since the log is a monotonically increasing function. So the maximum value of the log of the likelihood occurs at the same position as the maximum value of the original likelihood function.
If the (log-)likelihood function is differentiable then the maximum likelihood estimate is obtained by differentiating the (log-)likelihood function with respect to the parameters that are going to be estimated. This derivative is then set to zero and the expression is rewritten in such a way that we obtain the maximum likelihood estimator. After that, the second derivative of the (log-)likelihood is taken when working in one dimension, otherwise the Hessian matrix is considered. If the second derivative is negative for all the values of the parameter that are being estimated then the (log-)likelihood function is maximal in the maximum likelihood estimator obtained in the previous step. For higher dimensions, the eigenvalues of the Hessian matrix can for instance be used. If all the eigenvalues are negative, then the (log-)likelihood function has a maximum. In essence, we maximize the probability density with respect to the parameters for a fixed data set.

Asymptotic properties of the maximum likelihood Estimator
Maximum likelihood estimation is a commonly used way of estimating unknown parameters due to its asymptotic properties. In regular settings (so e.g. the (log-)likelihood is at least twice continuously differentiable and takes its maximum in the interior of the domain of the function) when the sample size goes to infinity (so $n \to \infty$) we have that:

- The maximum likelihood estimator is unbiased.
- The maximum likelihood estimator achieves the smallest possible mean square error of any unbiased estimator

7.1 Simulation results for the estimation of the parameters λ_g and μ using maximum likelihood estimation

The simulation results of the estimation of the parameters λ_g and μ using maximum likelihood estimation will be analysed in this chapter. The set up of the simulations is the same as explained in Chapter 4. The parameters used in the simulated data set can be found in Table 6. It is to be noted that the results of the current simulations are hypothetical as we have only used hypothetical values. These results are to check whether the computational framework works. The implementation of the log-likelihood function in Python can be found in Section 13.

Parameter	**Value**
grid ($n_x \times n_y$)	10×10
τ	1
λ_g	0.5
μ	0.1
end time	100τ
β	1
time lockdown start	30τ
time lockdown end	70τ
Time to death T_d	8τ

Table 6: Table with the parameter values used in the simulation for the estimates of λ_g and μ

Since this section is only focused on parameter estimation, the value of $\beta(t)$ is not considered. $\beta(t)$ is assumed to be known in every simulation (as it refers to the lockdown) and hence it would not change the parameter estimation.
In the analysis of the parameter estimates we will consider the relative error, which is in this case defined as:

$$\textbf{relative error} = \frac{\text{estimate - true value of the parameter}}{\text{true value of the parameter}} \cdot 100\% \qquad (28)$$

7.1.1 One run of maximum likelihood estimation

To begin, we perform one run of maximum likelihood estimation on the simulated data set. The results show that the maximum likelihood estimate for λ_g is $\hat{\lambda}_g = 0.5100$ and the maximum likelihood estimate for μ is $\hat{\mu} = 0.0999$. We see that these estimates give a good indication of where the true parameter lies. The relative error for λ_g is in this case 2.0%, which is reasonably small. If we would consider hypothesis testing with a significance level of for instance 5%, we would not reject this estimate. For the parameter μ, we have a relative error in absolute values of $1.0 \cdot 10^{-8}$ which is very small. Therefore, in this run, maximum likelihood estimation produced relatively accurate results. However, it is to be noted that each simulation also contains uncertainty which could contribute to the error.

7.1.2 100 runs of maximum likelihood estimation

Since in each simulation run the maximum likelihood estimates differ, it is interesting to see how the different estimates differ as some are more accurate than others. In order to do this we have simulated 100 different data sets and calculated their corresponding log-likelihood and maximum likelihood estimates of λ_g and μ. The simulation parameters are the same as stated in Table 6.

To get an understanding of the results, the estimated values for the parameters λ_g and μ have been plotted against each simulation run in Figure 36. From this figure we see that the estimates all fluctuate around the true value of their respective parameters. What is interesting to see is that the range of values for the estimates of λ_g is a lot larger compared to the range of values for the estimates of μ. This is probably due to the fact that it is harder to estimate λ_g compared to μ, as the variable λ_g is different for every individual.

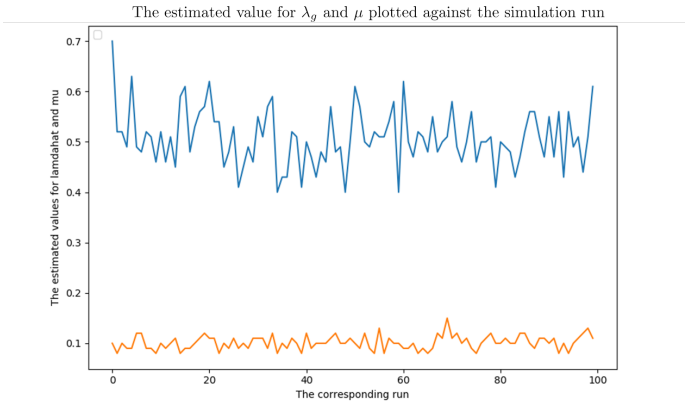

Figure 36: Graph of the estimated values of λ_g and μ plotted against each simulation. Blue are the estimated λ_g values and orange are the estimated μ values

In the limit (letting the sample size tend to infinity) the estimated values for λ_g and μ should converge to the actual values used in the model.
To every estimated pair of parameters λ_g and μ corresponds an estimate for the value of the log-likelihood. Every simulation also has an 'actual' log-likelihood value, so the log-likelihood value that is calculated using the real parameters of $\lambda_g = 0.5$ and $\mu = 0.1$. These values can be considered as the true value of the log-likelihood. To see if there is any significant difference between the 'estimated' log-likelihood values using the maximum likelihood estimates and the 'true' log-likelihood values, these two values have been plotted against each simulation in Figure 37. From this it can be seen that they look very similar with a small difference. The simulated log-likelihood values tend to be a bit lower compared to the actual values, but the difference is very small. Therefore, it can be concluded that parameter estimation by means of maximum likelihood estimation give relatively accurate estimates for the parameters.

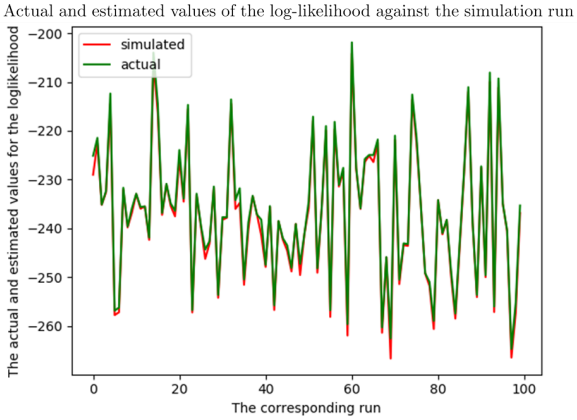

Figure 37: Graph of the actual values and estimated values of the log-likelihood plotted against each simulation. The green curve represents the actual log-likelihood values and the red curve the simulated log-likelihood values.

Furthermore, we consider the frequency of the different estimates. Therefore, two bar charts of the estimated parameters are plotted in Figures 38 and 39. The bar chart in Figure 38 shows that there is quite a large range between the estimated maximum likelihood values for λ_g, which we have also seen in Figure 36. However, all the estimates are centered around the true value of 0.5. The same holds for μ, but with a range between 0.08 and 0.15, which is a lot smaller compared to the range for the estimates of λ_g. This is probably due to the fact that μ is easier to estimate than λ_g, as the calculation of λ_g also involves random interaction with neighbours. The bar chart of μ can be found in Figure 39 and it is centered around the true value of $\mu = 0.1$.

If we would consider 95% confidence intervals for the estimated parameters λ_g and μ in the simulations, we obtain:

- A 95% confidence interval for λ_g is [0.3970,0.6176].
- A 95% confidence interval for μ is [0.0742,0.1282].

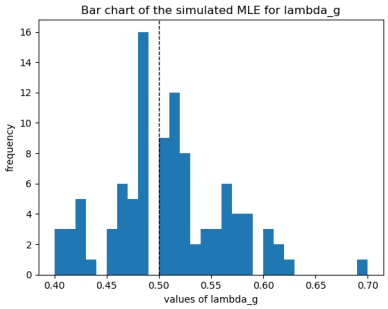

Figure 38: Bar chart of the maximum likelihood estimates for the parameter λ_g, the real value of the parameter is plotted in a dashed line.

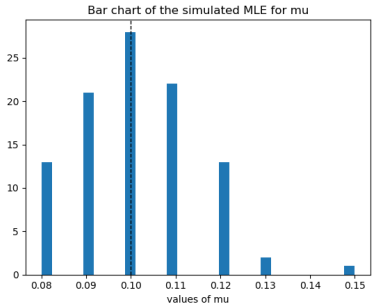

Figure 39: Bar chart of the maximum likelihood estimates for the parameter μ, the real value of the parameter is plotted in a dashed line.

It is interesting to see whether there is any relationship between the estimated values for λ_g and μ. From the scatter plot in Figure 40 we can see that there is not really a clear relationship between the parameters λ_g and μ. The graph doesn't show any kind of correlation or other relationship.

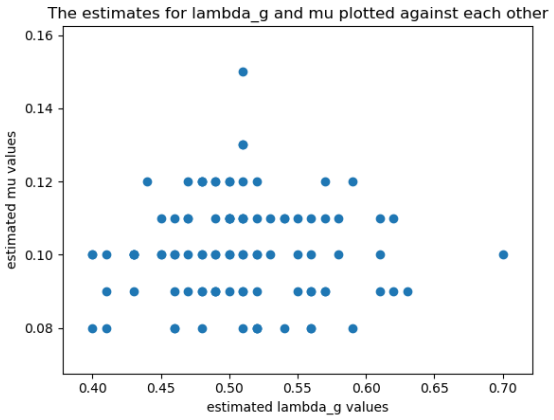

Figure 40: A scatter plot of the estimates for the parameter λ_g and μ against each other.

7.2 Heatmap of the log-likelihood

In the previous section, it has been shown that the parameter estimation by means of maximum likelihood produces quite accurate estimates. However, a question that may arise is whether the estimated values of λ_g and μ actually maximise the log-likelihood. Therefore, a fixed data set has been generated with parameters $\lambda_g = 0.5$ and $\mu = 0.1$ and then the following algorithm has been performed.
We loop over various pairs of different values for λ_g and μ in the interval $(0, 1)$ and calculate the corresponding log-likelihood. The log-likelihood should then be maximal at the pair (λ_g, μ) which is approximately equal to the real pair (λ_g, μ) used in the data set.
In order to interpret the results, a heatmap has been made. This heatmap can be seen in Figure 41. The lighter the colour the less negative the log-likelihood value.

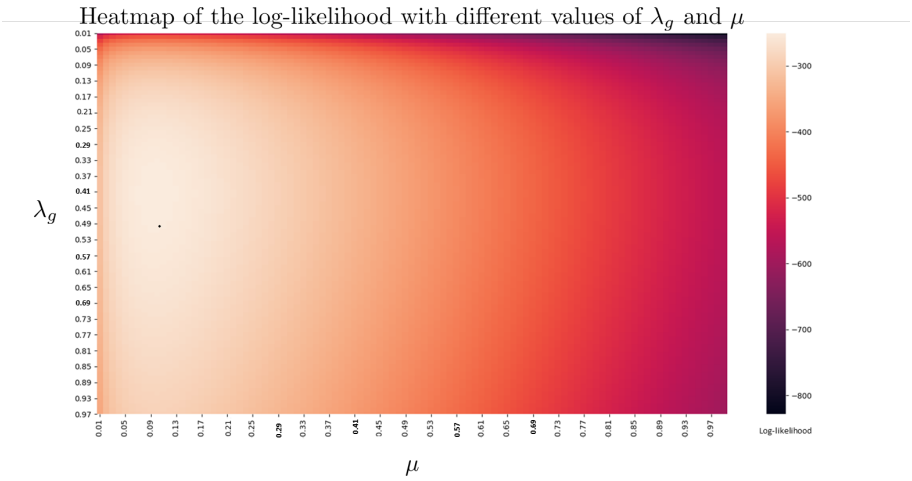

Figure 41: Heatmap of the log-likelihood. The black dot in the heatmap is corresponds to the actual pair of (λ_g,μ) that has been used in the simulation.

From Figure 41 we see that the log-likelihood is indeed maximal at the actual values of (λ_g, μ) as the colour gets lighter and lighter the closer we get to the actual values of both parameters. The area around the black dot (which represents the actual pair of the parameters λ_g and μ) is the lightest colour (almost white). The further away we go from the black dot, the darker the colour becomes, indicating a more negative log-likelihood value.

8 Bayesian Estimation

Maximum likelihood estimation is one way to do parameter estimation. Another way is based on Bayes' rule, shown in equation (29).

$$\mathbb{P}(\theta|X) = \frac{\mathbb{P}(X|\theta) \cdot \mathbb{P}(\theta)}{\mathbb{P}(X)} \tag{29}$$

where X and θ are two random variables or random vectors.

For Bayesian Estimation, we want to find the posterior probability density function $f(\theta|\mathbf{x})$, where θ is the vector of parameters and $\mathbf{x}$ is the vector containing the observed data, given the model $f(\mathbf{x}|\theta)$ and the prior $\pi(\theta)$. We denote the parameter space by Θ, which has a specified prior probability distribution for it. This prior distribution is then adjusted by Bayes' rule to the available data. The new adjusted probability distribution is then called the posterior probability distribution. If we denote $p_\theta(x)$ as the probability density of the random vector X. We can then consider the joint density of (X, Θ) which is given by:

$$p_{X,\Theta}(x, \theta) = p_{X|\Theta=\theta}(x) \cdot \pi(\theta) = p_\theta(x) \cdot \pi(\theta)$$

where $\pi(\theta)$ is the specified prior distribution for the parameter θ.
The marginal density of the vector X is obtained by integrating the joint density over all possible values of θ. Therefore the marginal is defined as follows:

$$p_X(x) = \int p_{X,\Theta}(x, \theta) d\theta = \int p_\theta(x) \cdot \pi(\theta) d\theta$$

Both the joint and the marginal density are then used to define the posterior density, which is just the conditional density of Θ given $X = x$ (by using Bayes rule):

$$p_{\Theta|X=x}(\theta) = \frac{p_{X,\Theta}(x, \theta)}{p_X(x)} = \frac{p_\theta(x) \cdot \pi(\theta)}{\int p_\vartheta(x) \cdot \pi(\vartheta) d\vartheta}. \tag{30}$$

In order to do Bayesian estimation we want to sample from this posterior distribution.

8.1 The Metropolis-Hastings Algorithm

One method to sample from the posterior distribution in order to do the Bayesian parameter estimation, is the Metropolis-Hastings algorithm which is a type of Monte Carlo Markov Chain (or short MCMC) method. The Metropolis-Hastings algorithm allows us to sample from any generic probability distribution, which we will call the target distribution (in this case we want the posterior distribution, but in general it could be any generic probability distribution), even if we don't know the normalizing constant. To do this, we construct a sample from a Markov chain whose stationary distribution is the target distribution that we are looking for. It starts by picking an arbitrary starting value and then iteratively accepting or rejecting candidate samples that are drawn from another distribution, one that is easy to sample. Suppose we want to sample from a target distribution $p(\theta)$. However, we only know this target distribution up to a normalizing constant or proportionality, so we have $g(\theta)$ to work with. The normalizing constant in $p(\theta)$ might for instance be too difficult to integrate or compute.
The Metropolis-Hastings algorithm is then as follows:

1. Select an initial value θ^0.
2. For $j = 1, \ldots, m$ (where m is some large integer number), we repeat the following:
   (a) Draw the candidate θ^* from a proposal distribution called $q(\theta^*|\theta^{j-1})$.
   (b) Compute the following ratio:
   $$\alpha = \frac{g(\theta^*) \cdot q(\theta^{j-1}|\theta^*)}{g(\theta^{j-1}) \cdot q(\theta^*|\theta^{j-1})}$$
   (c) Check α:
   - if $\alpha \geq 1$, we accept the candidate θ^* and set the value of $\theta^j = \theta^*$
   - if $0 < \alpha < 1$, we accept the candidate θ^* and set the value of $\theta^j = \theta^*$ with probability α.
     We do this by drawing a sample u^j from the uniform distribution $U(0, 1)$ and is $u^j < \alpha$ we set $\theta^j = \theta^*$.
   - if $0 < \alpha < 1$, we reject the candidate θ^* and set the value of $\theta^j = \theta^{j-1}$ with probability $1 - \alpha$. We do this by drawing a sample u^j from the uniform distribution $U(0, 1)$ and is $u^j > \alpha$ we set $\theta^j = \theta^{j-1}$.

The superscript ^{j} denotes the j^{th} iteration. Steps (b) and (c) act as a correction, since the proposal distribution q is not the target distribution p. So, at each

step in the Markov chain, we draw a candidate and then decide to either remain where we are now or move the chain to this candidate value. It is a Markov chain since whether we move to the candidate value or not is only dependent on where the chain currently is, not on the history before. The implementation of the Metropolis-Hastings algorithm for the mathematical model explained in this paper can be found in the Appendix 13.

8.2 Simulations

In this section some results will be given from a simulation that has been done using the Metropolis-Hastings Algorithm. In order to do this, a data set has been generated with the parameters stated in Table 7.

Parameter	**Value**
grid $(n_x \times n_y)$	4×4
τ	1
λ_g	0.5
β	1
μ	0.1
end time	100τ
Time to death T_d	8τ

Table 7: Table of parameters used in the simulation of Metropolis-Hastings

As can be seen from Table 7 no lockdown has been implemented in this case as we are only focusing on the parameter estimation right now.

8.2.1 Results of the Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm has been performed on the data set generated with the parameters stated in Table 7. To see if there is any difference in the results when doing different number of simulations, first a simulation of 1000 iterations is done and then a simulation with 90000 iterations of the algorithm is done.
In order to estimate the parameters λ_g and μ, we take the sample average as the point estimator for the simulation. So,

$$\hat{\Lambda}_g = \frac{1}{m}\sum_{i=1}^{m}\lambda_g^i \tag{31}$$

$$\hat{M} = \frac{1}{m}\sum_{i=1}^{m}\mu^i \tag{32}$$

where m is the number of iterations performed by the algorithm.

Simulation 1: 1000 iterations

The first simulation that has been done, performed 1000 iterations of the Metropolis-Hastings Algorithm. Both a graph and a histogram has been plotted for the estimated values for λ_g and μ. First, examine the results of the algorithm for the value of λ_g. The graph can be found in Figure 42 and the corresponding histogram in Figure 43.

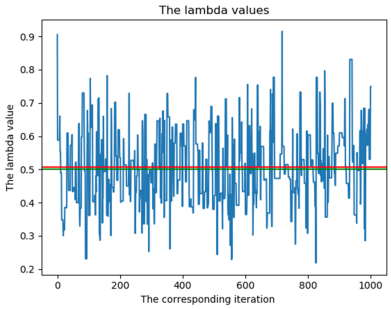

Figure 42: Graph of the values of λ_g generated by the Metropolis-Hastings Algorithm when doing 1000 iterations.

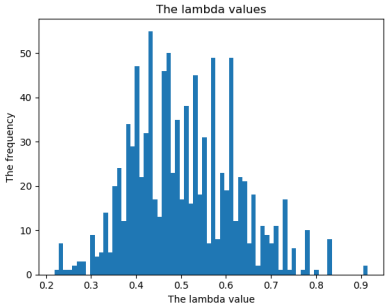

Figure 43: Histogram of the values of λ_g generated by the Metropolis-Hastings Algorithm when doing 1000 iterations.

In Figure 42 the actual simulated values of the algorithm are plotted in blue, the actual value of λ_g used in the data is plotted in green and the average estimated value for λ_g is plotted in red. As can be seen from the graph, the proposed values

of λ_g that are accepted fluctuate around the true value of the parameter. We also see that the average value of λ_g is close to the actual value of λ_g used in the data, which means that the average of the estimates of the values of λ_g is a good estimator for the parameter. The point estimate for λ_g in this simulation was $\hat{\lambda}_g = 0.5065$, while the true value for the parameter λ_g was 0.5. Relatively, the estimated average is about 1.3% off from the true value of the parameter. If we would for instance consider a significance value of 5%, the estimate would be within the limits. Figure 43 gives another indication that the algorithm gives a relatively good estimate for the real parameter λ_g. Most of the estimations for the value of λ_g were around 0.5, with a couple of outliers.

Next, consider the graph and histogram for the values of μ in Figures 44 and 45 respectively.

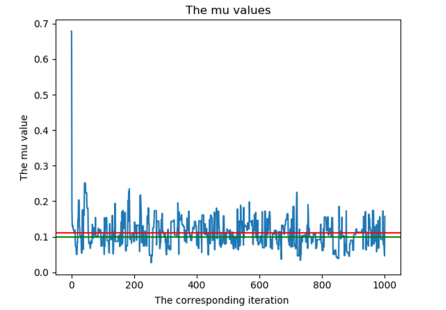

Figure 44: Graph of the values of μ generated by the Metropolis-Hastings Algorithm when doing 1000 iterations.

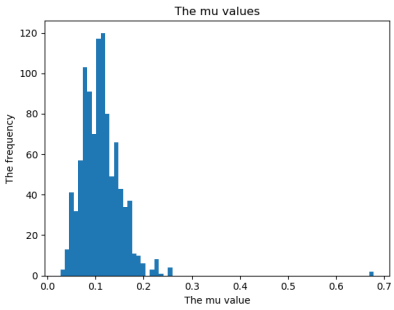

Figure 45: Histogram of the values of μ generated by the Metropolis-Hastings Algorithm when doing 1000 iterations.

Similar to the case for λ_g, the accepted values of μ fluctuate around the true value of the parameter. The histogram in Figure 45 shows that most of the accepted values of μ were around 0.1. From the simulation results we get that the average μ value of this simulation is $\hat{\mu} = 0.1116$, while the true value used in the data set is 0.1. At first sight, this estimate seems to be close to the real value of the data. However, the relative error is about 11.6% which is quite high. Therefore, it might be interesting to see if it makes any difference when we increase the number of iterations of the algorithm.

Simulation 2: 90000 iterations

In order to see if the estimation improves by increasing the number of iterations, the Metropolis-Hastings Algorithm has been performed 90000 times on the same data set. The results for the parameter λ_g can be found in Figures 46 and 47.

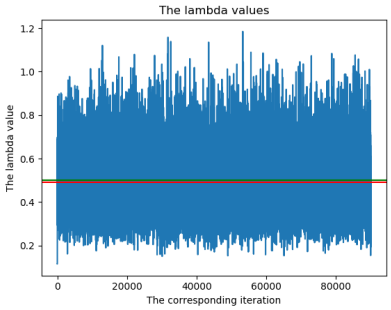

Figure 46: Graph of the values of λ_g generated by the Metropolis-Hastings Algorithm when doing 90000 iterations.

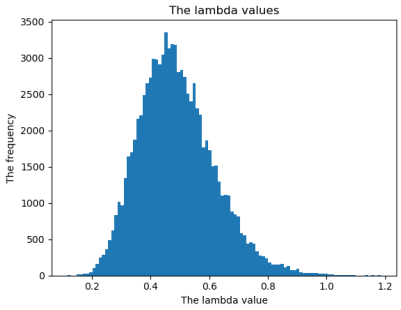

Figure 47: Histogram of the values of λ_g generated by the Metropolis-Hastings Algorithm when doing 90000 iterations.

In Figure 46 we see a similar trend as in Figure 42. The accepted values for λ_g continue to fluctuate around the true value of the parameter. The average value is still relatively close to the true value. The estimated value for λ_g was in this case $\hat{\lambda}_g = 0.4896$ while the true value was 0.5. It underestimated the value of λ_g a little. If we calculate the relative error we are about 2.1% off from the true value in absolute values. This error is a bit larger compared to the previous simulation when we did 1000 iterations. However, due to the randomness of the model and the fact that we have done a lot more iterations (the more we do, the bigger chance for bigger outliers) it would explain the fact why the error is a bit larger. Nevertheless, it is still in the same 'range' of about 1-3%.

The histogram of the values of λ_g is also centered around 0.5 with a slight skewness to the left. This is in line with what has been observed previously in the Figure 46. The range of the accepted λ_g values is a lot larger compared to the range of accepted μ values, which was also seen in the case when there were only 1000 iterations performed, as a result of the definition of λ_g.
The graph and the histogram with the estimated values for the parameter μ can be found in figures 48 and 49 respectively. The graph shows that we quickly converge to a range of μ values fluctuating around the true value of the parameter. The histogram is centered around the true value of $\mu = 0.1$ with a slight skew towards the left.

The estimate for μ is $\hat{\mu} = 0.1134$, while the true value used in the data was 0.1. Similar to the previous case, the estimate does give a good indication where the true value lies, but has a large relative error of 13% in this case. The increase in error is probably due to the increase in the number of iterations.

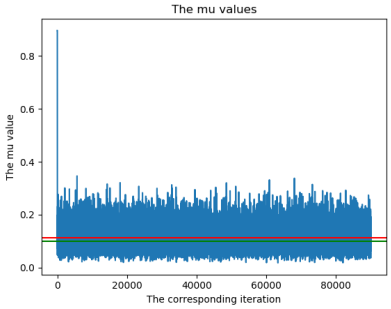

Figure 48: Graph of the values of μ generated by the Metropolis-Hastings Algorithm when doing 90000 iterations.

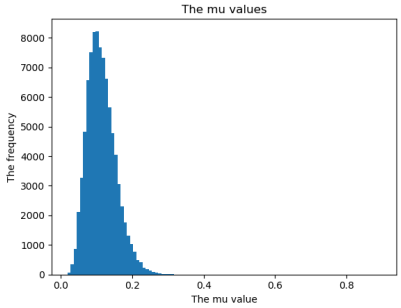

Figure 49: Histogram of the values of μ generated by the Metropolis-Hastings Algorithm when doing 90000 iterations.

All together, the Metropolis-Hastings Algorithm provides a good indication for the value of the estimated parameter (in this case λ_g and μ). However, the estimates are not very accurate. Especially for the parameter μ, we observed a large relative error in both simulations.

8.3 Metropolis-Adjusted Langevin Algorithm (MALA)

In this section we are going to consider another Monte Carlo Markov Chain algorithm called Metropolis-Adjusted Langevin Algorithm or in short MALA to sample from the posterior distribution. In the remainder of the paper, we will use the short-hand notation MALA. First the algorithm will be explained, followed by some simulation results.

8.3.1 The Metropolis-Adjusted Langevin Algorithm explained

The Metropolis-Adjusted Langevin Algorithm (MALA) is another Markov Chain Monte Carlo Method to obtain random samples from any probability distribution for which direct sampling is difficult (in our case: the posterior distribution). It is makes use of two components:

- The Langevin diffusion
- The Metropolis-Hastings algorithm

The Langevin diffusion will make use of the gradient in order to propose candidate values for the parameters and after that the accept/reject algorithm of the Metropolis-Hastings algorithm will be used to either accept or reject the newly proposed values. It is therefore to be expected that MALA will need less iterations of the algorithm to get equally accurate results compared to the Metropolis-Hastings algorithm, which will be done later in this section. The biggest difference between MALA and the Metropolis-Hastings algorithm is that instead of drawing candidate values from any generic proposal distribution, such as normal or exponential, the new candidates in MALA are proposed based on the Langevin diffusion.
The Langevin diffusion is given by a stochastic differential equation given in equation (33).

$$dX_t = -\boldsymbol{\nabla}\log(g(X_t))dt + \sqrt{2}dB_t \tag{33}$$

where, B_t represents standard Brownian motion, X_t is a stochastic process and $g(x)$ is the target density.
Under certain assumptions, the solution process $(X_t)_t$ is a Markov process that has a unique stationary distribution. If we could simulate the process $(X_t)_t$ exactly, then the distribution of X_t should converge to the target density. However, since in general X_t is a continuous time process, we cannot simulate this process exactly. Therefore, MALA discretizes this process using the Euler-Maruyama discretization.

The Euler-Maruyama discretization for a general stochastic differential equation of the form: $\mathrm{d}X_t = a(t, X_t)\,\mathrm{d}t + b(t, X_t)\,\mathrm{d}W_t$ is given in equation (34).

$$X_{n+1} = X_n + a(t_n, X_n)\delta t + b(t_n, X_n)\Delta W_n \tag{34}$$

where $\Delta W_n = W_{t_{n+1}} - W_{t_n}$.
We apply this discretization to the Langevin diffusion given in equation (33) and then we obtain equation (35).

$$X_{n+1} = X_n + h \cdot \boldsymbol{\nabla} \log(g(X_n)) + \sqrt{2h}\zeta_n \tag{35}$$

where $\zeta_n$is drawn from a multivariate normal distribution with mean $\mathbf{0}$ and covariance matrix equal to the identity matrix.

The MALA algorithm is then as follows:

1. Select an initial value θ_0.
2. For $j = 1, \ldots, m$ (where m is some large integer number), we repeat the following:
   (a) Draw the candidate θ^* by: $\theta^* = \theta^{j-1} + h \cdot \boldsymbol{\nabla} \log(g(\theta^{j-1})) + \sqrt{2h}\zeta^{j-1}$, where ζ^{j-1} is drawn from a standard normal multivariate distribution and g is the density.
   (b) Compute the following ratio:
   $$\alpha = \frac{g(\theta^*) \cdot q(\theta^{j-1}|\theta^*)}{g(\theta^{j-1}) \cdot q(\theta^*|\theta^{j-1})}$$
   (c) Check α:
   - if $\alpha \geq 1$, we accept the candidate θ^* and set the value of $\theta^j = \theta^*$.
   - if $0 < \alpha < 1$, we accept the candidate θ^* and set the value of $\theta^j = \theta^*$ with probability α.
     We do this by drawing a sample u^{j-1} from the uniform distribution $U(0, 1)$. If is $u^{j-1} < \alpha$, we set $\theta^j = \theta^*$ and if $u^{j-1} > \alpha$, we set $\theta^{js} = \theta^{j-1}$.

In essence, MALA is similar to the Metropolis-Hastings algorithm with the difference being the value of the candidate that has been drawn in each iteration. Another big difference is that in MALA the proposal densityis fixed to be Gaussian whereas in Metropolis-Hastings the family of the proposal density can be any family of probability distributions (usually centered around the previous value).

8.3.2 Automatic Differentiation

The Metropolis-Adjusted Langevin Algorithm evaluates the gradient of the log-likelihood function. In order to implement this in Python, we made use of automatic differentiation.
Automatic differentiation is a set of techniques to evaluate the derivative of a function numerically. Every computer program is able to make use of elementary arithmetic operations, such as addition, subtraction, multiplication and division, as well as elementary functions, such as sine, cosine, exponential, logarithm etc. The idea is to apply the chain rule repeatedly to these operations in order to compute the derivatives of more complicated functions. Derivatives of any order can be obtained. More background theory about automatic differentiation can be found in [14].

Automatic differentiation using dual numbers

One way to perform automatic differentiation is by forward automatic differentiation using dual numbers. It makes use of an extremely small number, call it ϵ, where it is defined that $\epsilon^2 = 0$. The idea is to replace every real number, call it x, by $x + x' \cdot \epsilon$, where x' is also a real number. The value of x' is called a 'seed' and is chosen arbitrarily. To illustrate the procedure to get a derivative, consider the following simple example.
Suppose it is required to evaluate the first derivative of the function $f(x) = x^2$ using automatic differentiation by dual numbers. First, substitute the number $x + x' \cdot \epsilon$ into the function and simplify the expression:

$$(x + x'\epsilon)^2 = x^2 + 2xx'\epsilon + x'\epsilon^2 = x^2 + 2xx'\epsilon \text{ by the definition of } \epsilon \tag{36}$$

What can be seen in equation (36) is that the second term is the derivative of the function $f(x)$ times by the little extra number that was added to the original number, as $2x$ is the derivative of x^2. This is exactly what is done in automatic differentiation using dual numbers. This procedure can be generalised to arbitrary polynomials and extended to analytic functions.

8.3.3 Simulation results

The Metropolis-Adjusted Langevin Algorithm has been performed on the same data set that had been used for the Metropolis-Hastings algorithm. First a simulation of 10000 iterations is done and later the number of iterations is increased to 90000.

Simulation 1: 10000 iterations

In this case 10000 iterations have been run. In Figures 50 and 51 the graph and the histogram of the simulated values of λ_g can be found.

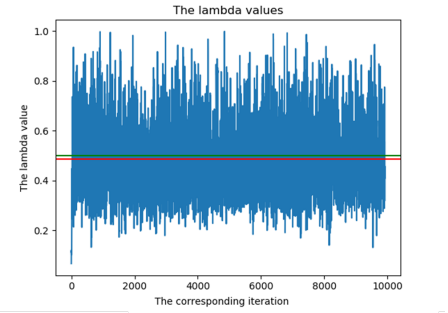

Figure 50: Graph of the values of λ_g generated by the MALA when doing 10000 iterations.

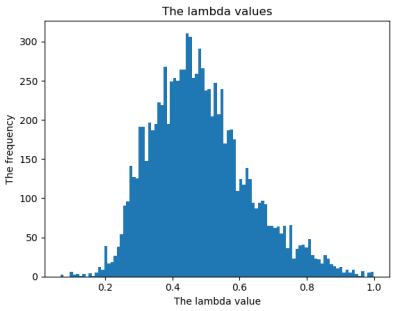

Figure 51: Histogram of the values of λ_g generated by the MALA when doing 10000 iterations.

Similar to the graphs that were in the Metropolis-Hastings Algorithm paragraph, the red line in Figure 50 represents the estimate generated from MALA and the green line represents the actual value of λ_g used in the data set. In this case the actual value of λ_g was $\lambda_g = 0.5$ and the estimated value was $\hat{\lambda}_g = 0.4876$. The graph shows that the estimate is very close to the actual value as the red and green line almost overlap. The histogram in Figure 51 shows that the estimated values are all centered around the true value of $\lambda_g = 0.5$. This result is in line with the graph in Figure 50

The relative error is approximately 2.5% in absolute values. If we compare this estimate to the estimates that we got in the Metropolis-Hastings algorithm, we see that the relative error of MALA is a bit larger compared to the relative error obtained from the Metropolis-Hastings algorithm. If we performed 1000 iterations of the Metropolis-Hastings algorithm we were about 1.3% off from the true value and if we did 90000 iterations we were 2.1% off. However, the relative errors are still within the same 'range' of about 1-3%. Due to the randomness in the algorithm, the relative errors differ in each simulation that is performed, but in general it stays within the range of 1-3%.

The results for the parameter μ can be found in Figure 52 and in Figure 53. From Figure 52 it can be seen that the algorithm quickly converges to a range of values around the true value of the parameter. In this case, the initial guess was about 0.52, but after a few iterations the estimates all lie in the range of 0.05 to 0.3. This is similar to the case when the Metropolis-Hastings algorithm was done. However, the estimates seem to be a bit closer to the true value. The histogram in Figure 53 confirms these observations as well. It is centered around the true value of 0.1 with most estimates in the range from 0.05 till 0.2. The estimate of μ in this simulation run is $\hat{\mu} = 0.0959$. The true value of μ used in the data set was 0.1. This estimate confirms the observations made in the figures.

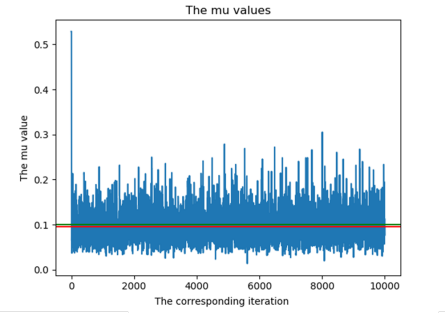

Figure 52: Graph of the values of μ generated by the MALA when doing 10000 iterations.

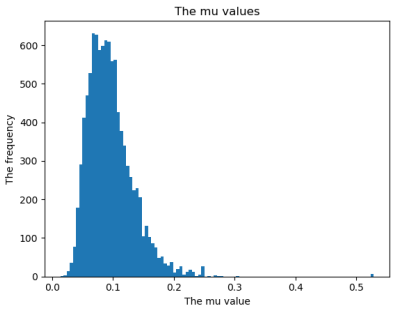

Figure 53: Histogram of the values of μ generated by the MALA when doing 10000 iterations.

The relative error in this estimate is about 4.1% in absolute values. Similar to the estimate for λ_g we are underestimating the true value of the parameter a bit. If we would take a significant level of 5% we would not reject the estimate that we got for μ when doing for instance hypothesis testing. Comparing this estimate to the estimates generated by the Metropolis-Hastings algorithm, it can be concluded that the estimate from MALA is more accurate. In the case that 1000 iterations were done of the Metropolis-Hastings algorithm, the relative error was 11.6% and after increasing the amount of simulation iterations to 90000, the relative error was about 13%. So the Metropolis-Adjusted Langevin Algorithm produced a more accurate estimate for μ. This could be explained by the fact that the 'convergence' to a value close to the real parameter is faster in MALA as a result of a more accurate candidate parameter that is proposed in each iteration.

simulation 2: 90000 iterations

In order to see whether it makes a difference if the number of iterations of the simulation is increased, another simulation with 90000 iterations of the MALA on the same data set is done. The results for λ_g can be found in Figures 54 and 55. From Figure 54 we see that the average value, which is the estimate for λ_g, is very close to the true value of $\lambda_g = 0.5$. Most simulation values range between 0.3 and 0.8. Similar to previous simulations, the estimates fluctuate around the true value of the parameter. The histogram in Figure 55 also shows that most of the estimates are centered around the value of 0.5, but it shows some left skewness. The estimate for λ was $\hat{\lambda}_g = 0.4903$. This confirms the observations made in Figures 54 and 55. The relative error is calculated to be 1.9% in absolute values. If this relative error is compared to the error in the previous simulation of 2.5%, we see that the difference in errors is not significant. Therefore, increasing the number of iterations did not really affect the estimate of λ_g.

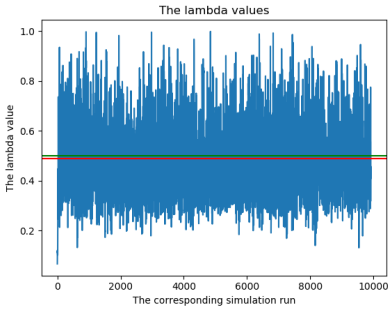

Figure 54: Graph of the values of λ_g generated by the MALA when doing 90000 iterations.

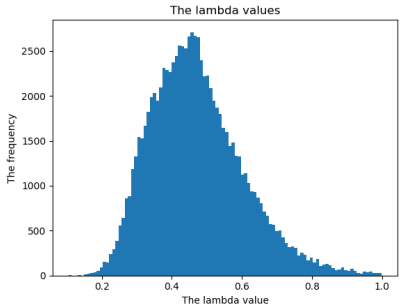

Figure 55: Histogram of the values of λ_g generated by the MALA when doing 90000 iterations.

Furthermore, Figures 56 and 57 show the results for the parameter μ. The graph in Figure 56 shows that the range of values for the parameter μ are between 0.05 and 0.2, similar to the case when there were 10000 iterations done. There are a few outliers, but in general the estimates are close to the true value of the parameter. The red and green line almost overlap, which indicates that the estimate for the value of μ is close to the true value of the parameter. The histogram in Figure 57 shows some positive skewness centered around the value of 0.1.

The estimate for μ was in this case $\hat{\mu} = 0.0957$ with a relative error of 4.3% in absolute values. Comparing this error to the error made when doing 10000 simulation iterations, we see that they are about the same size. The relative error in the case that MALA was done 10000 times was 4.1% in absolute values, while in this case the relative error was 4.3% in absolute values. The difference is not significant. Therefore, doing more iterations did not improve the parameter estimation for the parameter μ.

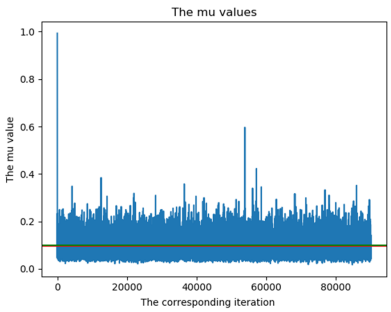

Figure 56: Graph of the values of μ generated by the MALA when doing 90000 iterations.

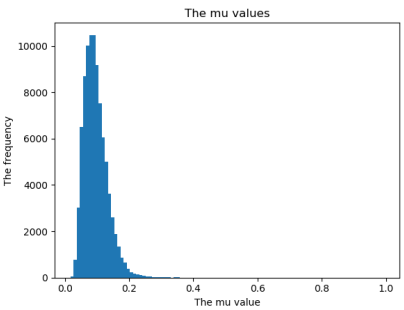

Figure 57: Histogram of the values of μ generated by the MALA when doing 90000 iterations.

To conclude, the results showed that increasing the number of iterations from 10000 to 90000 did not improve the parameter estimation for both parameters. The difference between the relative errors was not significant. However, the estimate of the parameter μ obtained by MALA was better compared to the estimate obtained by the Metropolis-Hastings algorithm.

8.4 The Metropolis-Hastings algorithm compared to MALA

Both the Metropolis-Hastings algorithm and MALA were used to sample from the posterior density in order to do Bayesian parameter estimation. The biggest difference between these two Monte Carlo Markov Chain methods is how the candidate value is proposed in each iteration. Another difference is the choice for the proposal density, in MALA the proposal density is fixed to be Gaussian whereas in the Metropolis-Hastings algorithm the family of the proposal density can be any probability distribution (usually centered around the previous value).
The downside of MALA is the computational effort that it requires as it needs to evaluate gradients of density functions. On the other hand, the big advantage of MALA is that this algorithm would need less iterations to produce the same accurate result compared to the Metropolis-Hastings algorithm, when the initial estimate is for instance far away from the true value. In order to see if this is also the case for the parameters λ_g and μ, both the Metropolis-Hastings algorithm and MALA were run with an initial estimate of $\lambda_g^0 = 10$ and $\mu^0 = 10$. The data set on which the simulations have been performed is the same as the one used in the previous sections, where the parameters are given in Table 7.

8.4.1 The Metropolis-Hastings algorithm

We start by examining the results for the Metropolis-Hastings algorithm when we take the initial values of λ_g and μ to be 10 for both and doing 10000 iterations of the algorithm. The graphs for the values of λ_g that have been generated by the Metropolis-Hastings algorithm are found in figures 58 and 59.

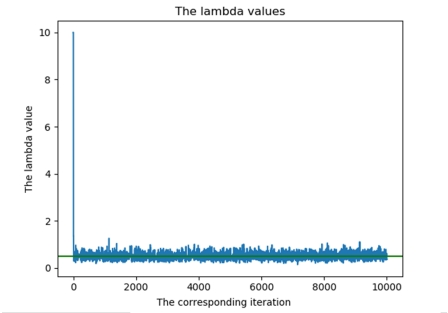

Figure 58: Graph of the values of λ_g generated by the Metropolis-Hastings algorithm when doing 10000 iterations. The initial value is $\lambda_g^0 = 10$.

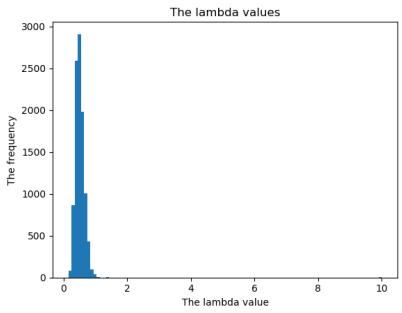

Figure 59: Histogram of the values of λ_g generated by the Metropolis-Hastings algorithm when doing 10000 iterations. The initial value is $\lambda_g^0 = 10$.

Similar to the previous graphs, the green line in Figure 58 represents the actual value of the parameter used in the simulation. We see that the algorithm quickly adjusts the starting value of 10, to a value close to $\lambda_g = 0.5$. The histogram in Figure 59 shows that most of the estimates are between 0.3 and 0.8. The average λ_g value generated in this simulation of 10000 iterations was $\hat{\lambda}_g = 0.5088$, with a relative error of 1.8%, which is small. Surprisingly, despite having a far-off initial guess of 10, the relative error is still in the same range of 1-3%, similar to the previous simulations. The results of the values of μ can be found in figures 60 and 61. The algorithm quickly converges to a value close to the true value of the parameter μ. Figure 60 shows this as the accepted values almost immediately drop from an initial value of 10 for μ to a range of values around the true value of 0.1. The estimate for μ is $\hat{\mu} = 0.1187$ with a relative of about 18.7%, which is relatively large. The histogram in Figure 61 shows that almost all values are close to the true value, but there are a couple of outliers present, which might affected the estimate as we are considering the sample average as the estimator.

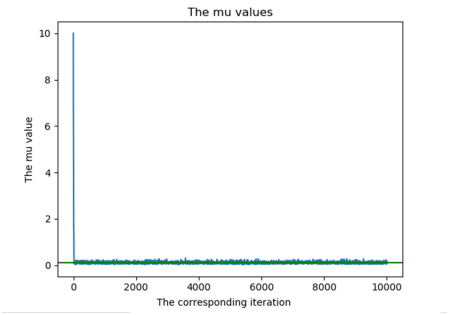

Figure 60: Graph of the values of μ generated by the Metropolis-Hastings algorithm when doing 10000 iterations. The initial value is $\mu^0 = 10$.

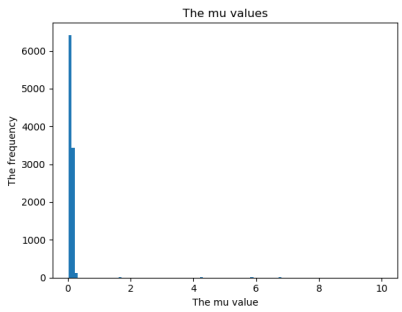

Figure 61: Histogram of the values of μ generated by the Metropolis-Hastings algorithm when doing 10000 iterations. The initial value is $\mu^0 = 10$.

The estimation improved slightly if the number of iterations was increased. Hence, another simulation run has been done with 90000 iterations of the algorithm. The results for λ_g can be found in figures 62 and 63. Both figures look very similar to the figures 58 and 59. The average of the λ_g values was $\hat{\lambda}_g = 0.5025$, which is a bit closer to the true value of $\lambda_g = 0.5$ compared to the estimate before. The begin estimate of $\lambda_g = 10$ was far off from the true value, thus this will contribute heavier to the average value by doing 10000 iterations compared to doing 90000 iterations, which might explain the more accurate estimate. The relative error of this estimate is about 0.5%, which is really small.

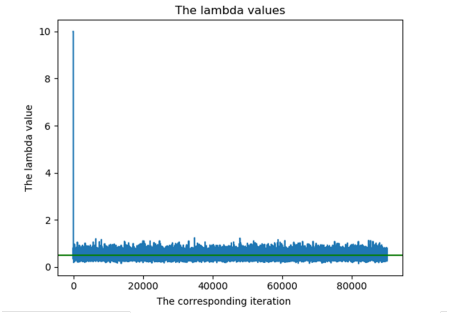

Figure 62: Graph of the values of λ_g generated by the Metropolis-Hastings algorithm when doing 90000 iterations. The initial value is $\lambda_g^0 = 10$.

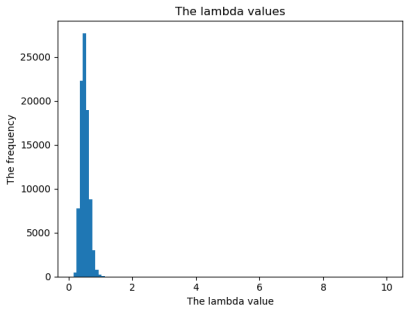

Figure 63: Histogram of the values of λ_g generated by the Metropolis-Hastings algorithm when doing 90000 iterations. The initial value is $\lambda_g^0 = 10$.

In figures 64 and 65 the results for the parameter μ can be found. The observations in the results of λ_g also hold for μ. The estimate for μ is $\hat{\mu} = 0.1097$, which is closer to 0.1 compared to the estimate of 0.1187 obtained after 10000 iterations. A possible explanation of the better estimate is the same as in the case of λ_g. The relative error is calculated to be 9.7% which is still large, but an improvement compared to the error in the previous simulation.
From the simulations we can conclude that if we have an initial guess that is far off from the true parameter of the model, running more simulations of the Metropolis-Hastings algorithm will produce a more accurate estimate. Note that in this case the sample average is taken as the point estimator for the parameter, if the estimator was taken differently then the results might differ.

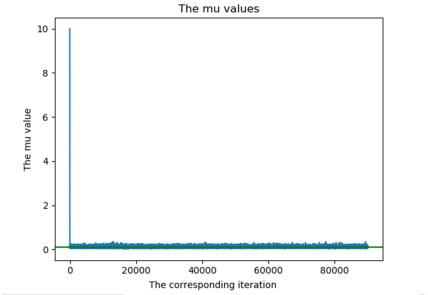

Figure 64: Graph of the values of μ generated by the Metropolis-Hastings algorithm with 90000 iterations. The initial value is $\mu^0 = 10$.

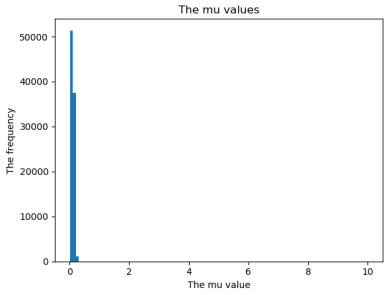

Figure 65: Histogram of the values of μ generated by the Metropolis-Hastings algorithm with 90000 iterations. The initial value is $\mu^0 = 10$.

8.5 The Metropolis-Adjusted Langevin Algorithm

The proposed candidate vector of parameters of MALA is likely to be more accurate compared to the candidate generated by the Metropolis-Hastings algorithm. Therefore, it is expected that MALA will needs less simulation iterations to produce an estimate of the same (or perhaps better) accuracy. To see if this expectation is correct or not, MALA has been run 10000 times on the same data set as the Metropolis-Hastings algorithm. The results for the parameter λ_g can be found in Figures 66 and 67.

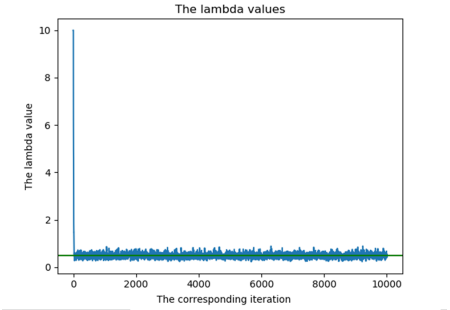

Figure 66: Graph of the values of λ_g generated by MALA when doing 10000 iterations. The initial value is $\lambda_g^0 = 10$.

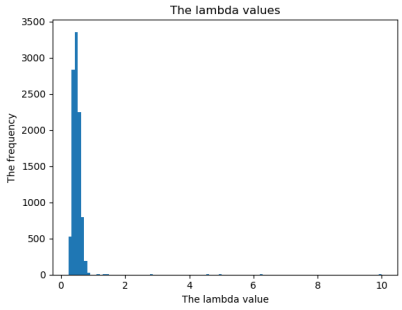

Figure 67: Histogram of the values of λ_g generated by MALA when doing 10000 iterations. The initial value is $\lambda_g^0 = 10$.

Similar to the Metropolis-Hastings algorithm, the convergence to the range where the true value of λ_g lies is quite fast. The initial estimate of 10, drops almost immediately to a value in the interval from 0.3 to 0.8 approximately. For the remainder of the simulation iterations the estimates remain in this interval. The histogram in Figure 67 shows that the histogram is centered around the true value of 0.5, but with a few outliers. These outliers are most probably due to the initial estimate of 10, which is far off from the true value of 0.5.
The estimate obtained for λ_g is $\hat{\lambda}_g = 0.5006$. The relative error is 0.12%, which is the smallest error compared to all the previous errors we got in the simulations. Therefore, we can conclude that for the estimation of the parameter λ_g, the expectations are confirmed. After only 10000 iterations of MALA, we already get a more accurate estimate compared to 90000 iterations of the Metropolis-Hastings algorithm.
Subsequently, consider the results of MALA for the parameter μ. Just like in the previous cases, we see that the value of μ drops almost immediately to a small range around the true value of 0.1 in Figure 68 and stays there for the remainder of the iterations. The histogram in Figure 69 shows that the estimates are

centered around the true value of the parameter, with a few outliers due to the far-off initial estimate. The estimate for μ was in this case $\hat{\mu} = 0.1079$, which is also slightly better compared to the estimates we obtained using the Metropolis-Hastings algorithm. The relative error is about 7.9% which is also the smallest error for μ compared to the previous errors obtained in the Metropolis-Hastings algorithm. Therefore, also for the parameter μ, we can conclude that MALA is a more efficient way of parameter estimation as it needs less iterations to produce a more accurate estimate compared to the Metropolis-Hastings algorithm.

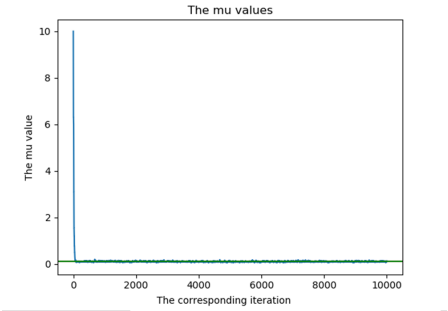

Figure 68: Graph of the values of μ generated by the Metropolis-Hastings algorithm when doing 90000 iterations. The initial value of $\mu^0 = 10$.

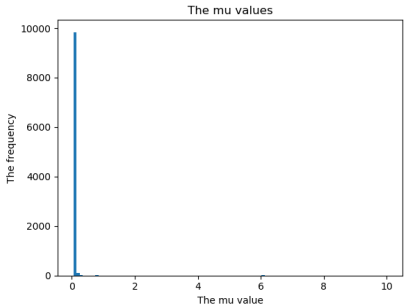

Figure 69: Histogram of the values of μ generated by MALA when doing 10000 iterations. The initial value is $\mu^0 = 10$.

9 Discussion

We have proposed a mathematical model to model the spread of the novel coronavirus epidemic in the world. the model is different than most of the models that have already been proposed, which are based on the general S(E)IR-models [15] [16] [17] [18] [19]. There are two unique features in the presented model: (1) it is based on cellular automata (2) it uses exponentially distributed times between different states, which makes the model stochastic of nature. The model is able to predict the dynamics of the disease under various lockdown scenarios with different infection and recovery rates. However, we also note some limitations in the proposed model as well as it's underlying assumptions. The proposed model is still general as it is not yet adjusted to specific countries or regions and it has not taken into account some factors that play a role in the dynamics of the disease.

To set up the mathematical model, a constant population size of n people is assumed. When extending this model to a specific country or region, birth and death rates can be taken into account. In addition, every individual is assumed to be be in one of four different states, namely: susceptible, infected, recovered or dead. We assumed that the virus is only transferred from infected people to susceptible people, while in reality the virus is also able to spread through surfaces or objects. People who just recovered from the COVID-19 virus, are able spread the disease too. According to a study published in the journal JAMA, patients who recovered from the COVID-19 virus had been tested positive for the virus in every test between days 5 and 13 post-recovery [12].

The severity of the symptoms caused by the coronavirus vary from person to person. Some people only experience very mild symptoms like fever and dry cough, while others have difficulty breathing, chest pain and might even lose their speech or movement and must be hospitalised. Elderly (above 70) also have a higher chance of dying from the coronavirus compared to younger individuals. Certain risk groups including individuals with chronic respiratory or pulmonary problems, heart patients or diabetics also have a higher chance of dying as a result of the coronavirus compared to 'normal people' their age.

Moreover, there is the possibility of reinfection. In this model, the recovery rate μ is assumed to constant for every individual. However, research shows that over a period of about three months, the number of antibodies in recovered people rapidly decreases. Recent studies found that there is a high chance of losing immunity to the COVID-19 virus after a period of time. [20]. Death is assumed when a person has been infected for a certain period of time. In practice, some people might have mild symptoms for a very long time and recover or have very heavy symptoms and die within a few days. This is highly individually. Further research should be undertaken to investigate the probability of reinfection as well as introducing different sub-groups of infected people based on their symptoms and

health conditions. To better quantify these transmission rates, one could assume that they are functions of time, as the rate of recovery not only changes with age, but also with the circumstances at that time (for instance IC capacity).

In order to do simulations, the model has been simplified to a square grid where each individual is assumed to only have four neighbours at most as we only considered the neighbouring nodes as the set of neighbours. This is not realistic. Future studies are encouraged to implement for instance a random process to determine the neighbours of an individual. This allows for a flexible number of contacts between individuals, but also some isolated sub-communities.

The different lockdown simulations showed that a severe lockdown is able to extinguish the outbreak in a limited amount of time, while less severe lockdown policies mainly cause a steady amount of infections as well as recovering and dying people over time. The term *flattening the curve* is often used in the media to describe: (i) reduction in the peak number of infections, to prevent the health care system from being overwhelmed and (ii) increasing the duration of the pandemic over a larger time interval, but with the same number of cases at the end. This phenomenon has been seen back in the simulations for various lockdown scenarios. The time of the epidemic was stretched over a longer period when a lockdown was implemented and the peak number of infections decreased. However, the total number of infections was the same as well as the total number of people who recovered and died, as can be seen in the cumulative graphs. Lifting the lockdown rules resulted into several peaks in the number of infections. The second and possibly third peak were a lot lower compared to the first one, due to a lower number of susceptible people. Nonetheless, governments are encouraged to impose appropriate measures in their lockdown policies to reduce the peak in the number of infections when they are relaxing the lockdown rules.

The model provides a theoretical framework to investigate the spread of the COVID-19 virus. The variability in the results due to the randomness in the model, makes the simulations more realistic as similar lockdown protocols in different countries have different effects on the number of casualties. Culture, population density and public health care are for instance factors that influence this. In the paper by Cooper, Mondal and Antonopoulos in [21] a SIR-model is developed for various different communities, which investigate the time evolution of different populations and the diversity in the parameters for the spread of the disease. Future research on this topic could help modify this model to a specific country or region.

The impact of different pairs of λ_g and μ have been investigated in the case when there is no lockdown implemented. In this way the model can give insights into the effects of these parameters on the spread of the virus. The simulations showed that the length of the epidemic was shortest either for a large value of μ

(0.5) combined with a small value of λ_g (0.2) or for large values of λ_g (above 0.55), regardless of the value of μ. The first pair $(\lambda_g, \mu) = (0.2, 0.5)$ corresponds to a small probability of infection and a large probability of recovery. This is seen back in the average number of susceptible people, which was largest in this case. The average number of recovered and dead people was smallest as well as the average maximum number of infections for this pair of parameters. These findings suggest, that if a country is able to lower the probability infection rate, by taking adequate measures, in combination with a large recovery rate, then both the length of the epidemic and the number of casualties will be smallest.

The effect of the parameter μ is important in determining the average number of people who remain susceptible, as well as the average number of recovered and dead people at the end of the epidemic. A long as the value of λ_g is above 0.3, the value of μ determines the evolution of the outbreak. Therefore, it is of vital importance for countries to provide enough IC capacity as well as good medical treatment for the patients to increase the recovery rate. However, it is to be noted that the maximum number of infections in a certain time interval is dependent on both parameters λ_g and μ. It is recommended that in future studies, they consider various lockdown scenarios with varying pairs of these parameters. Perhaps it is possible to find a pair of values for (λ_g, μ) and a lockdown strategy to eradicate the virus. It is also advisable to consider larger population samples or more simulation runs, as in this case we have only run simulations for a constant population size of 10000 people and have only done 100 simulation runs per pair of (λ_g, μ).

As the presented model is not yet adjusted to a specific country or region neither is it as extensive as many proposed S(E)IR-models. The results cannot be directly compared to previous studies. In future studies, it might be possible to relate the outcomes of this Spatial Cellular Automata Markov Chain model to the S(E)IR-models presented in for instance [15] [16] [17] [18] [19]. A downside of the current model, compared to the classical deterministic models like the S(E)IR-models, is that it is relatively expensive to execute. One could possibly optimise the model within the Monte Carlo framework.

Besides the formulation of the mathematical model, two methods of parameter estimation have been discussed, namely maximum likelihood estimation and Bayesian estimation. In order to sample from the posterior distribution in Bayesian estimation, we used two different Markov Chain Monte Carlo algorithms: the Metropolis-Hastings algorithm and the Metropolis-Adjusted Langevin Algorithm (or short MALA). Maximum likelihood estimation has the downside that the likelihood function needs to be worked out explicitly for the model and differentiable. In this model, the mathematics was relatively easy so we could derive the expressions that were needed. Nonetheless, if the model is extended to a more complicated model, this might not be possible anymore.

In the Metropolis-Hastings algorithm we encountered large relative errors for the parameter μ. These errors were probably large because the range of the values of μ that were accepted using the Metropolis-Hastings algorithm was a lot smaller compared to the range of λ_g, but further research is needed to explain the size of these errors. The Metropolis-Adjusted Langevin Algorithm (or short MALA) produced more accurate estimates for the parameter μ compared to the Metropolis-Hastings algorithm. The relative errors that were observed in the simulations of MALA were at most 10%, while in the Metropolis-Hastings algorithm we had a relative error of almost 19% when we initially started with an estimate far from the true value of the parameter. It is also possible that the mean was not a good estimator for the parameter μ. Further work is required to establish what estimator would give the smallest relative error for the parameters.

In both the Metropolis-Hastings algorithm and MALA, the algorithm was first done for λ_g and then for the proposed values of μ. This method of keeping one of the parameters constant could potentially have affected the parameter estimation. This choice was made under the assumption that the parameters λ_g and μ were independent of each other. In the maximum likelihood estimation we have plotted the estimated values for λ_g and μ against each other in a scatter-plot and didn't see any relationship among them, but that is not sufficient evidence to prove the independence of the two parameters. This is an important issue for further research as well as linking the parameters of this model to the basic reproduction number R used in for instance [16].

A note of caution is that in the simulations of both Metropolis-Hastings and MALA, we have done still a relatively large number of iterations. If one would only do few iterations, say 10 or 100, the estimates are very likely to be less accurate. It could for instance be that in the numerical simulations, there is a certain point where the uncertainty in the model determines the error and that increasing the number of iterations after this point is reached does not produce more accurate results. Future studies that consider the parameter estimation of the current model are recommended to investigate at which number of iterations the error is determined by the uncertainty in the model. In addition, there are many other algorithms and ways of parameter estimation that can be used, so in future studies it is recommended to look into more detailed and other methods of parameter estimation or other algorithms to sample from the posterior distribution function.

Given the current development of COVID-19, it is widely speculated that this pandemic is not soon to be over. The proposed model could help countries to predict future scenarios and potentially offer information about how to reduce the epidemic wave and eradicate the disease.

10 Conclusion

In December 2019, a novel coronavirus emerged in China and spread rapidly across the world causing a pandemic, affecting society and the global economy [22] [23]. Mathematical modelling techniques are effective tools to find patterns in disease outbreaks and provide useful predictions in the evolution of epidemics. In this bachelor thesis, we have developed a spatial Markov Chain Cellular Automata Model for the spread of the COVID-19 virus. This model uses network topologies for the progression of an epidemic by considering each person on a grid individually and using stochastic principles to determine the transition between different states. The model can be used to predict the time-evolution of epidemics under various lockdown policies as well as the maximum number of infected people at a certain time interval. Using both maximum likelihood estimation as well as Bayesian estimation, one can estimate the input parameters from a given data set for this particular mathematical model.

Different lockdown scenarios have been simulated to see the impact of the severity of lockdown policies. Mild lockdowns were not really effective in reducing the spread of the virus, while heavy lockdowns caused the number of infected cases to be approximately constant over the lockdown period and severe lockdown protocols could eradicate the virus. Lifting the lockdown rules caused multiple peaks in the number of active cases as the virus could spread more rapidly when the lockdown rules were relaxed. It is recommended for governments to strictly monitor what is happening when lifting lockdown rules. It might be wise to implement some more stricter rules even if the rules have been relaxed previously, when the spread of the virus is flaming up again. In reality, we see that many governments have indeed taken this approach.

In addition to that, the impact of varying the infection probability rate as well as the recovery rate has been investigated when no lockdown was implemented. An important finding was that the recovery rate parameter μ is crucial for determining the number of susceptible, recovered and dead people, as long as the infection probability rate λ_g was larger than 0.3. On the other hand, λ_g mainly influenced the duration until there were no more active cases in the population. However, the maximum number of infections was dependent on both parameters.

The parameters of the model have been estimated using maximum likelihood estimation as well as Bayesian estimation, where we sampled from the posterior distribution function by two Markov Chain Monte Carlo Methods: the Metropolis-Hastings algorithm and the Metropolis-Adjusted Langevin algorithm (MALA). All the algorithms produced relatively accurate estimates for the parameters λ_g and μ. The relative error of λ_g was generally in the range of 1-3%, while the relative error for μ differed per estimation method. It is to be noted that if the model is extended or adapted for a certain country, maximum likelihood estimation might

be more difficult, as it might not be possible to get a differentiable likelihood. The numerical simulation results demonstrated that estimates using MALA 'converge' faster to the real value of the parameter compared to the Metropolis-Hastings algorithm, when for instance the initial estimate is far off from the true value, and had the smallest relative error. Hence, MALA produced the most accurate results with the least number of iterations.

As more and more data and information on COVID-19 becomes available, the proposed model could be useful for estimating the duration of the epidemic under various (lockdown) scenarios. Parameter estimation methods can be applied in future studies to real data to fit the model to for instance specific countries using the approaches mentioned in this thesis or more advanced methods.

Acknowledgements
The author is thankful for the support provided by the supervisors Frank van der Meulen and Fred Vermolen from the Department of Applied Mathematics, Delft University of Technology, the Netherlands and the Hasselt University, Belgium.

11 Appendix Figures

In section 4 various lockdown scenarios have been considered. The corresponding figures which show the spread of the virus in the various simulations for different values of the lockdown parameter $\beta(t)$ are presented in this appendix.

11.1 Simulation 2: $\beta(t) = 0.5$

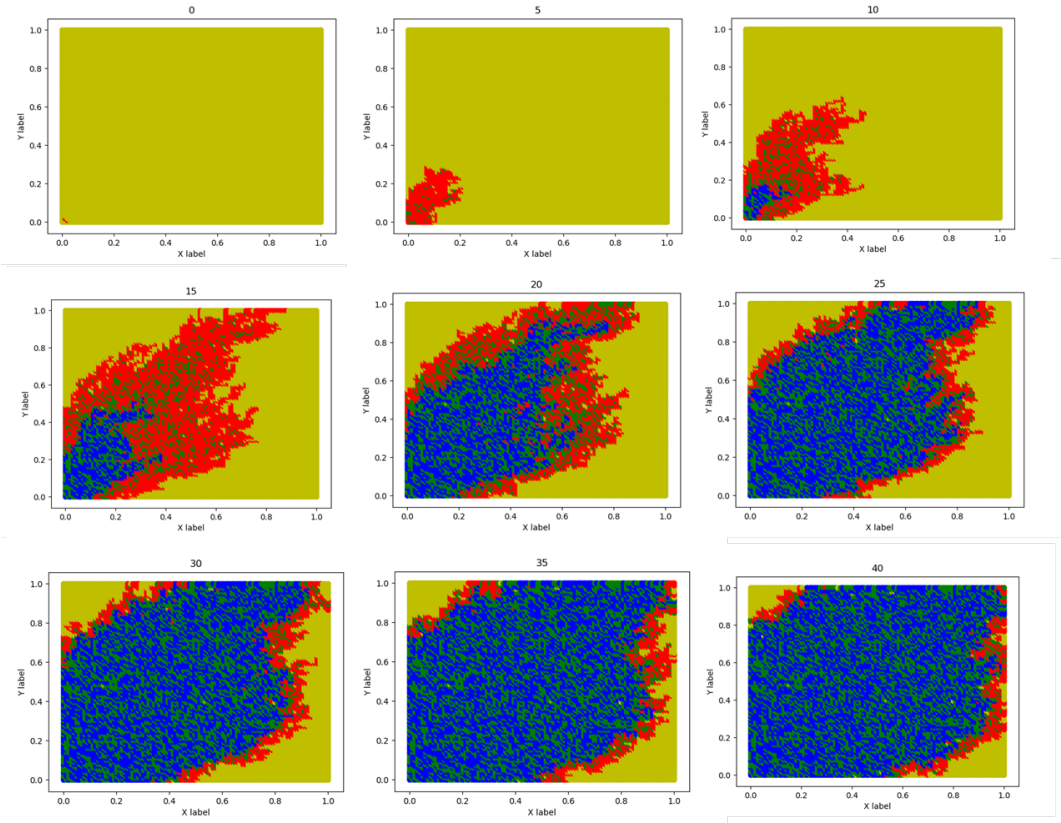

Figure 70: Graph of the state of the system at different times for a 100x100 grid and a medium lockdown of $\beta(t) = 0.5$ where the lockdown starts at time 15τ and ends at time 60τ.

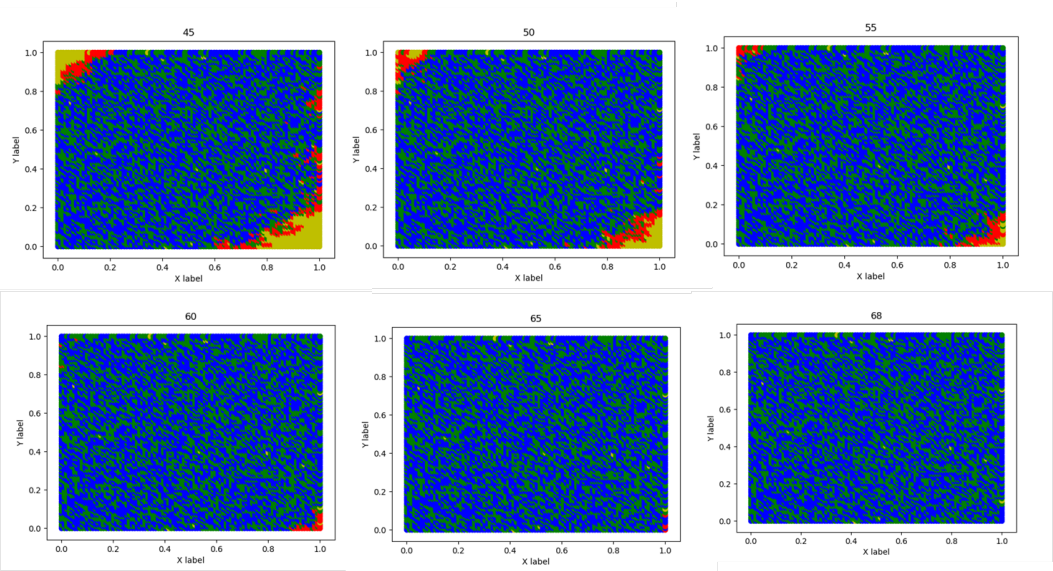

Figure 71: Continuation of the graph of the state of the system at different times for a 100x100 grid and a medium lockdown of $\beta(t) = 0.5$ where the lockdown starts at time 15τ and ends at time 60τ.

From Figures 70 and 71 it can be seen that the spread of the virus is rapid in the beginning, but slows down when the lockdown has been implemented. Especially around the times 20τ, 25τ and 30τ, the decline in the number of active cases is significant. After the lockdown implementation at time 15τ, a large amount of people have already recovered or died from the virus, but there is still a large proportion of susceptible people left.
From time 35τ till time 60τ, the number of infected people is approximately constant as a result of both the lockdown policies and the fact that many people have already been infected and are either recovered or dead. Surprisingly, there is still a proportion of people left, who have never become infected and thus remained susceptible. This is also seen in reality, as some people are lucky to never come in contact with the virus.
Another noteworthy observation is that after the lockdown has been implemented the green area (corresponding to the recovered people) is first larger than the blue area (corresponding to the dead people). The rationale behind this observation is that in this model it takes more time for people to die than to recover, by the way death is assumed. However, after some time blue area expands and we observe a tilted rectangular front consisting of all blue and green area with red edges, showing the spread of the disease. This observation is seen in all the simulations for various values of $\beta(t)$.

11.2 Case 3: $\beta(t) = 0.1$

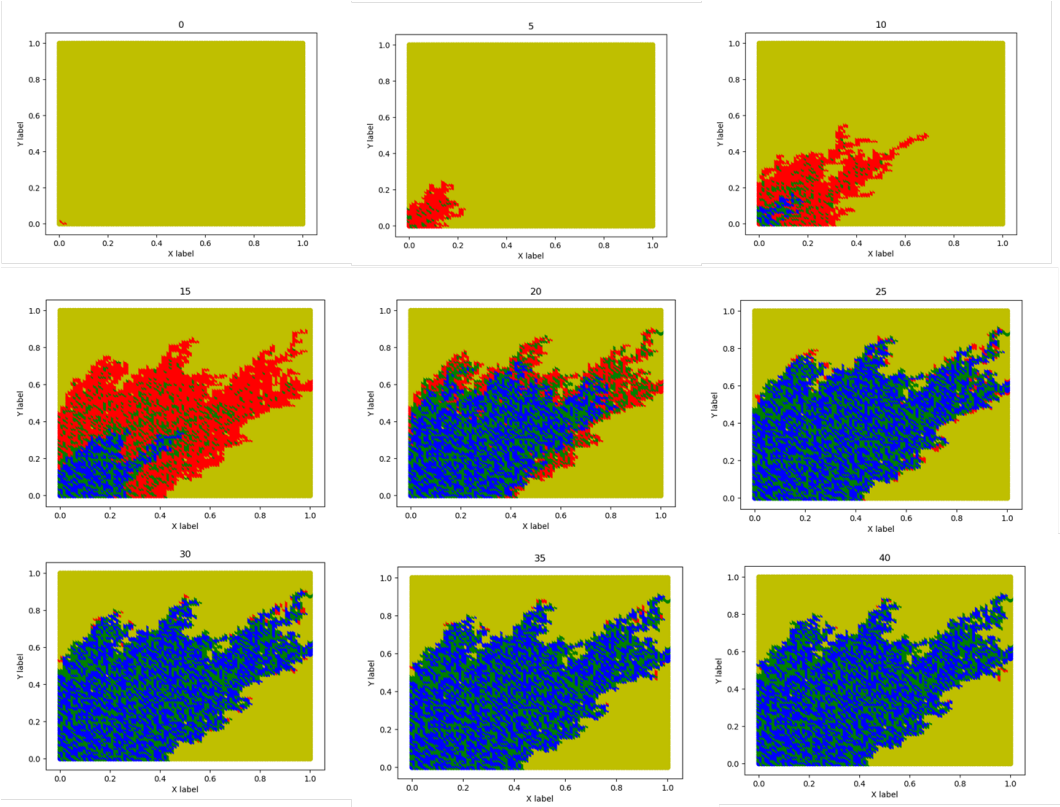

Figure 72: Graph of the state of the system at different times for a 100x100 grid and a severe lockdown of $\beta(t) = 0.1$ where the lockdown starts at time 15τ and ends at time 60τ.

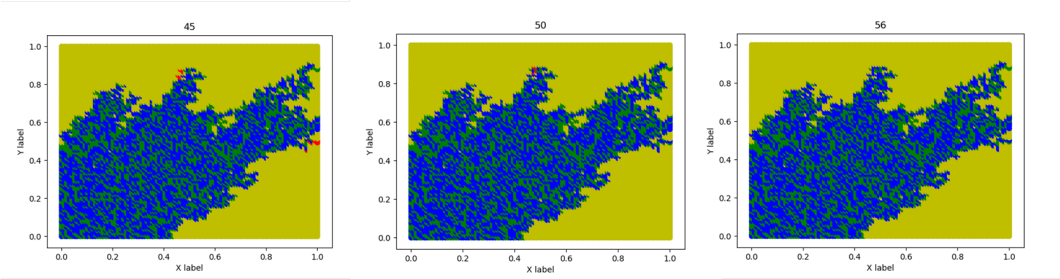

Figure 73: Continuation of the graph of the state of the system at different times for a 100x100 grid and a severe lockdown of $\beta(t) = 0.1$ where the lockdown starts at time 15τ and ends at time 60τ.

From Figures 72 and 73 we see that the spread of the virus is very similar to all the other simulations until time 15τ, as no lockdown has been implemented yet. Due to the severity of the lockdown, the spread decreases rapidly between the times 15τ and 20τ. Only the edges of the in this case 'flame-like' front of recovered and dead people still have active cases. At time 25τ, there is only a very small proportion of the population infected by the virus, which results in less infections over the next time period. Eventually, at time 56τ, the spread of the virus has ceased. There are no more infected people are left. Interestingly, there is still a significant number of susceptible people in the population.
These results suggest that if a country (or region) implements a very severe lockdown, it is able to stop the virus from spreading. However, since the model relies on heavy assumptions, we can only say that a severe lockdown will drastically reduce the amount of active cases, but we cannot promise eradication of the virus. Another thing to be mentioned is that in this simulation, we are not considering travelling between different regions or countries.
Furthermore, since the virus has already stopped spreading (in this simulation) at time 56τ, the lifting of the lockdown at time 60τ did not influence the results. Nonetheless, if one would maintain the lockdown for a shorter period of time, then the end time of the lockdown might be of influence potentially causing a second peak in the number of infections.

11.3 Case 4: $\beta(t) = 0.3$

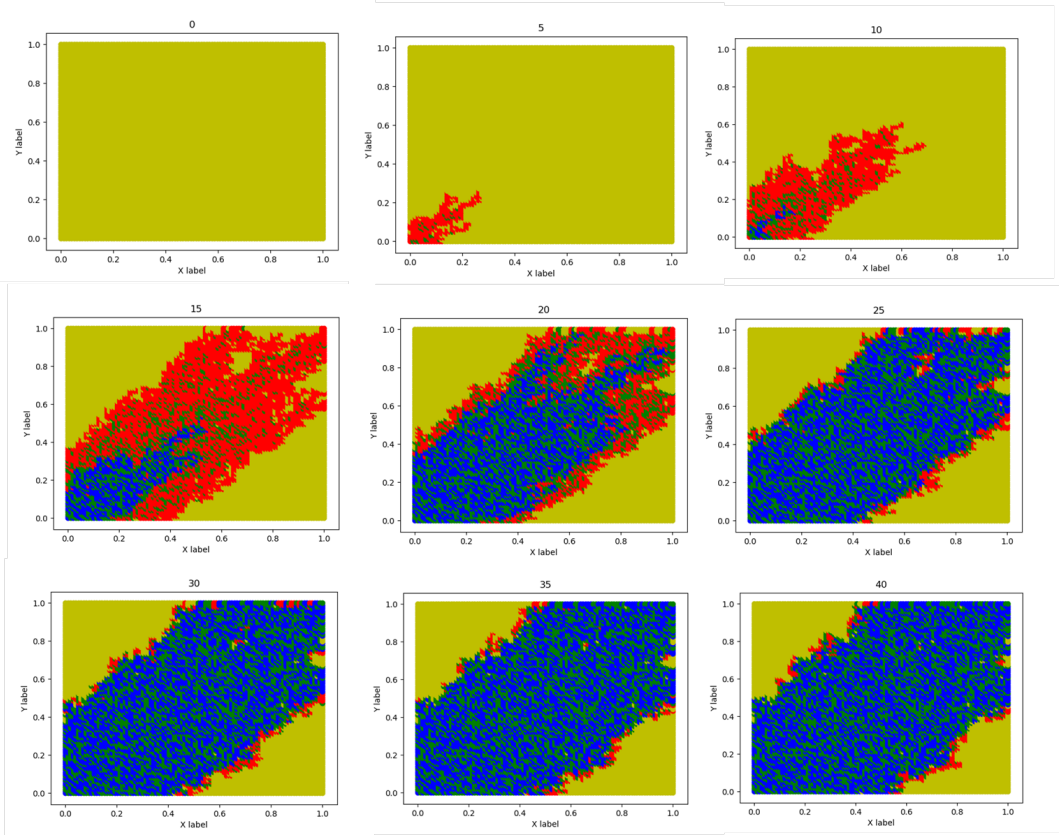

Figure 74: Graph of the state of the system at different times for a 100x100 grid and a heavy lockdown of $\beta(t) = 0.3$ where the lockdown starts at time 15τ and ends at time 60τ.

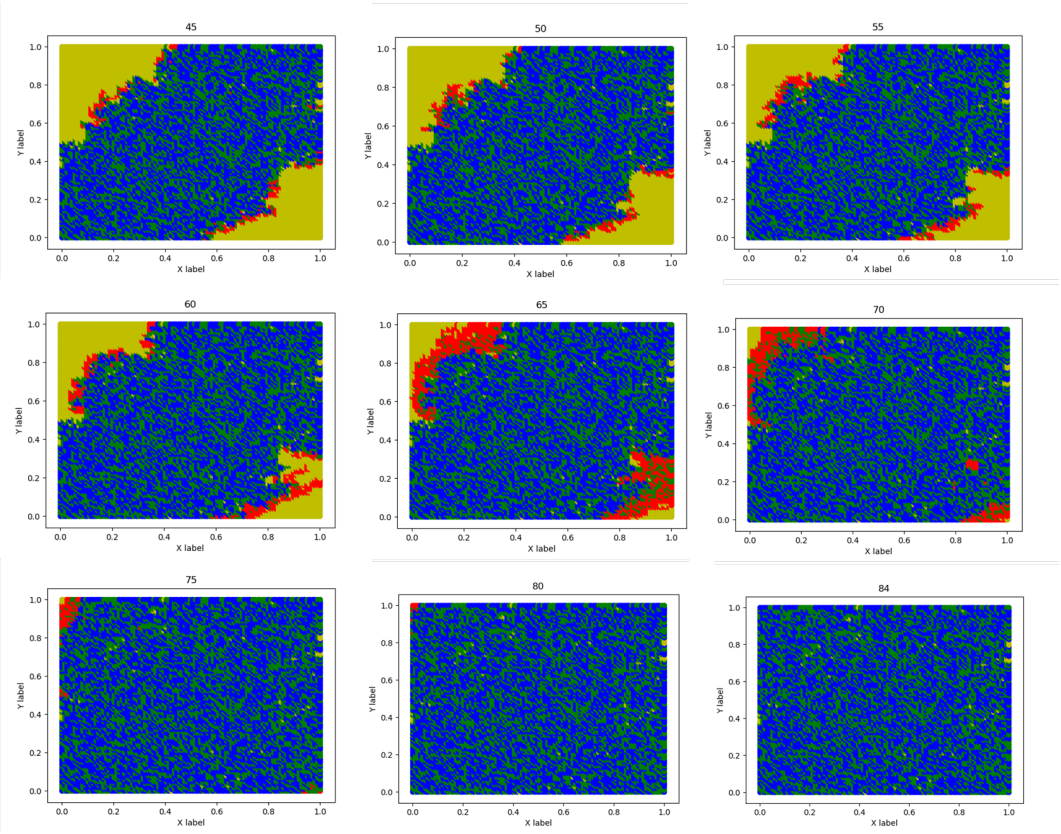

Figure 75: Continuation of the graph of the state of the system at different times for a 100x100 grid and a heavy lockdown of $\beta(t) = 0.3$ where the lockdown starts at time 15τ and ends at time 60τ.

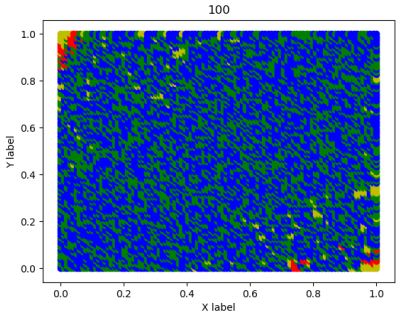

Figure 76: Graph of the end state of the system for a 100x100 grid and a heavy lockdown of $\beta(t) = 0.3$ where the lockdown starts at time 15τ and is valid until time 100τ.

The first observation made is that a lockdown of $\beta(t) = 0.3$ is insufficient to stop the spread of the virus, while a lockdown of $\beta(t) = 0.1$ was able to do this. However, the effects are still clearly present. At time 15τ, the lockdown results in a drastic decrease in the number of infections. At time 20τ, the number of active cases has significantly reduced. Over the course of the next time steps, the spread of the virus is visibly slower. The amount of active cases remains constant over the entire lockdown period. Most of these are at the edges of the 'rectangular' front of recovered and dead people. The little 'islands' of susceptible people are also present in this simulation. These people are lucky to never become infected by the disease, despite being surrounded by recovered or dead people.
The immediate lifting of the lockdown at time $60\tau$is undeniably seen in the simulation results: the virus spreads rapidly among the remaining susceptible population until the end of the simulation. This can be regarded as a 'second peak' in the number of active cases. In reality, a strict lockdown is never relieved immediately, but more gradually. There are still some infected people left at the end of the simulation, as well as susceptible people. If one would have run the simulation longer than time 100τ, then these people would also have recovered or died. All in all, the results do provide an insight in the effects of a heavy lockdown policy.

11.4 Case 5: $\beta(t) = 0.7$

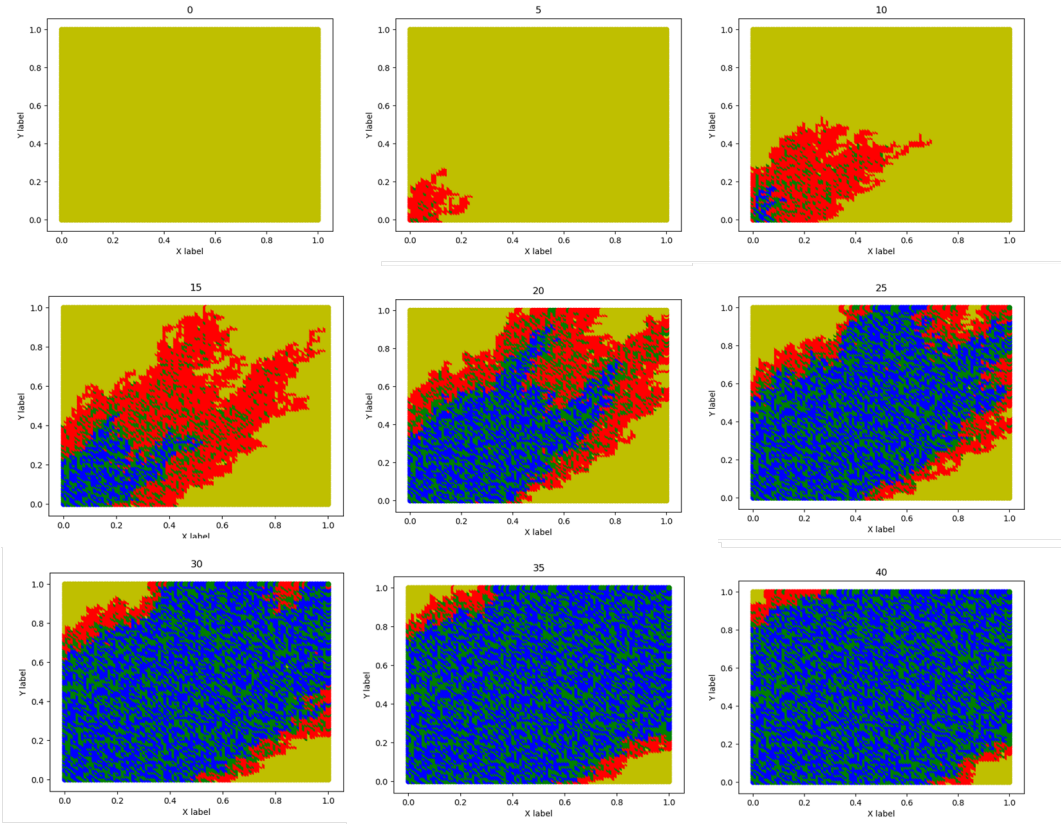

Figure 77: Graph of the state of the system at different times for a 100x100 grid and a mild lockdown of $\beta(t) = 0.7$ where the lockdown starts at time 15τ and ends at 60τ.

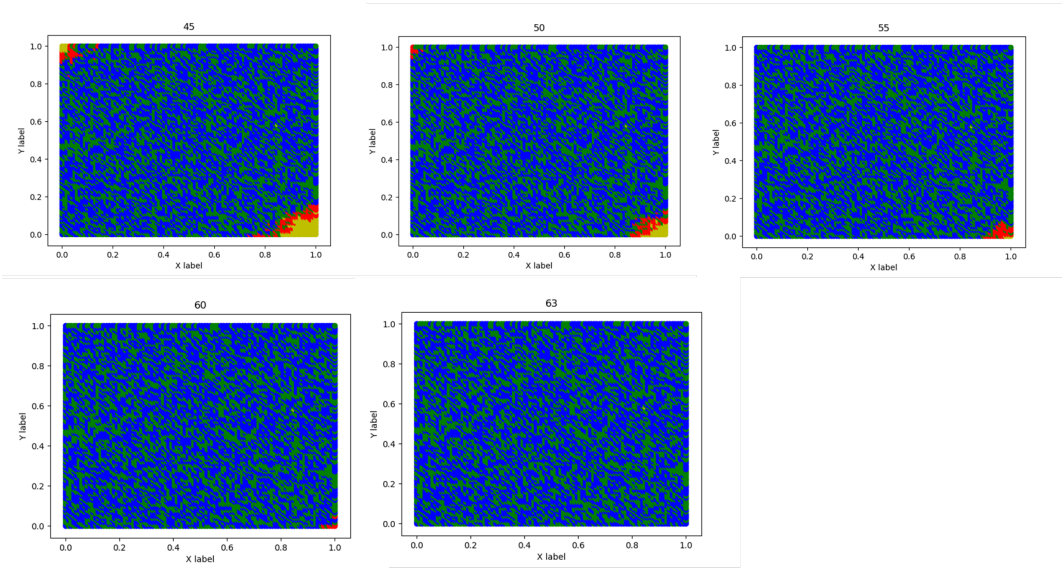

Figure 78: Graph of the state of the system at different times for a 100x100 grid and a mild lockdown of $\beta(t) = 0.7$ where the lockdown starts at time 15τ and ends at 60τ. This is the extension of the previous graph.

The dynamics of the spread of the virus in Figures 77 and 78 is similar to the previous cases. The only difference compared to the previous lockdown scenarios is that the 'band width' of the red area after the lockdown is implemented is wider. This indicates that the amount of active cases is larger, which is logical as we are now considering a lockdown of $\beta(t) = 0.7$.

Interestingly, if the end time of this simulation at 63 is compared to the other lockdown scenarios, we see that this end time is close to the end time of a medium lockdown of $\beta(t) = 0.5$, which had an end time of 68τ. This suggests that the duration of the epidemic with a lockdown of 0.5 or a lockdown of 0.7 is almost the same. Comparing this result to the case when there is no lockdown implemented, where the simulation ended at time 44, we see that a lockdown does seem to have a significant effect in stretching the epidemic over a longer period of time. In summary, these results show that the difference between a mild lockdown of $\beta(t) = 0.7$ and a medium lockdown of $\beta(t) = 0.5$ is minor. However, the lockdown policies do have a significant effect in lengthening the duration of the epidemic.

11.5 Lifting the lockdown rules, end plots of the simulations

11.5.1 Case 1: severe to medium to no lockdown

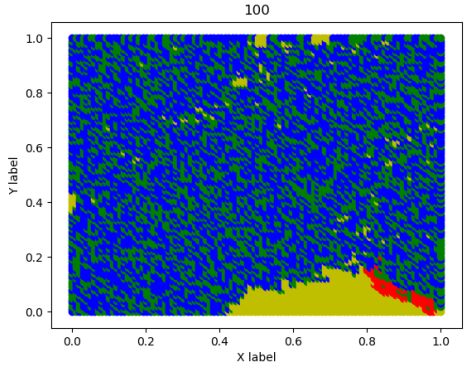

Figure 79: Graph of the state of the end of the system for a 100x100 grid and a severe lockdown of $\beta(t) = 0.1$ where the lockdown starts at time 15, the lockdown is then lifted at time 35τ to a medium lockdown of $\beta(t) = 0.5$ and at time 60τ to no lockdown.

11.5.2 Case 2: heavy to mild to no lockdown

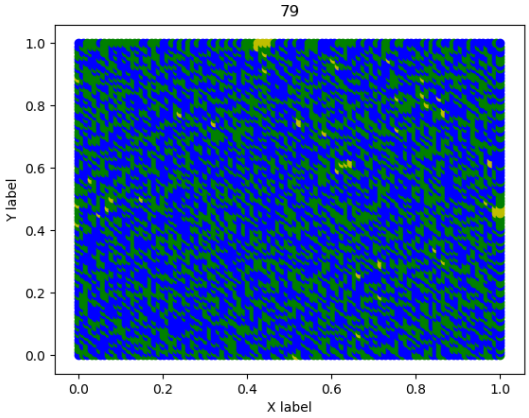

Figure 80: Graph of the state of the end of the system for a 100x100 grid and a heavy lockdown of $\beta(t) = 0.3$ where the lockdown starts at time 15τ, the lockdown is then lifted at time 35τ to a mild lockdown of $\beta(t) = 0.6$ and at time 60τ to no lockdown.

11.5.3 Case 3: heavy to even milder to no lockdown

Note that this is not a separate subsection in section 4, but the results are mentioned.

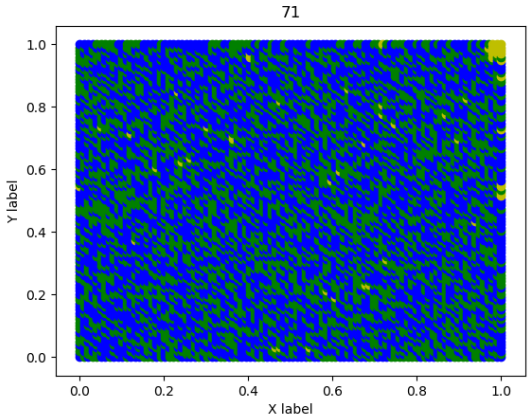

Figure 81: Graph of the state of the end of the system for a 100x100 grid and a heavy lockdown of $\beta(t) = 0.3$ where the lockdown starts at time 15τ, the lockdown is then lifted at time 35τ to a mild lockdown of $\beta(t) = 0.7$ and at time 60τ to no lockdown.

12 Appendix Probability Theory

This Appendix contains probability theory definitions that are used in this paper.

12.1 Bernoulli Random variable

Recall from probability theory the following: If the random variable Z has a Bernoulli distribution with parameter p. Then:

$$Z = \begin{cases} 1 & \text{with probability p} \\ 0 & \text{with probability 1-p} \end{cases}$$

We can write:

$$\mathbb{P}(Z = z) = p^z(1-p)^{(1-z)} \qquad \text{where } z \in \{0, 1\}$$

12.2 Standard Normal Distribution

If it is assumed that $Z \sim N(0, 1)$, so Z is standard normally distributed then the cumulative distribution function $\Phi(z)$ is given by equation (37) and the corresponding density function ϕ is given by equation (38).

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{\frac{-t^2}{2}} dt \tag{37}$$

$$\phi(z) = \frac{1}{\sqrt{2\pi}} e^{\frac{-z^2}{2}} \tag{38}$$

13 Appendix Python implementations

The implementations of certain functions and algorithms in order to do parameter estimation are explained in this appendix.

13.1 Log-likelihood function implementation

The log-likelihood function is implemented as a function in Python. It has arguments: lamhat, mu, dicperson and dicneighbour and timelist. The values of lamhat and mu correspond to the values of λ_g and μ in the mathematical model respectively. Dicperson is a dictionary that contains the information of the state of every person at each time step. The key-value pairs are as follows: (time, xvalue person, yvalue person) = 'state of person'; where the time corresponds to the time step, the x-value and y-value determine which node is considered on the grid and the state of the person tells us in which of the four states: *susceptible, infected, recovered* or *dead* this person is.
The dictionary dicneighbour is similar to dicperson, but instead of storing the state of the person as the value of each key, the value of the intensity of the contact with the neighbours is stored. The reason why this dictionary is implemented is because the interpersonal relations between the people over time change and is also random (as we draw random uniform numbers). Therefore, since the probability of becoming infected is dependent on the value of the infected neighbours, this value is relevant. The key-value pairs are as follows: (time, xvalue person, yvalue person) = 'value of the sum of the interpersonal contact with the infected neighbours'.
The timelist argument in this function is for a list of the time that the simulation data has been run. This differs in every simulation since the time that it takes in every simulation for the number of infections to be zero or the virus to die out is different. This is also the case for the data that is obtained in real life. In certain countries it might take less time for a certain proportion of the population to get infected compared to other countries or regions.
In the function itself, the first step is to iterate over all the time steps and the people in the population. Then the log-likelihood is calculated for each of the four possible states that the person is in.
There are in total five possible transitions for each person:

1. The person is susceptible at time t and stays susceptible at time $t+\tau$. The corresponding log-likelihood is then: $\log\left(e^{-\lambda_g \cdot (\sum_{j \in N_i^{Inf}(t)} a_{ij}(t)) \cdot \tau}\right)$.

2. The person is susceptible at time t and becomes infected at time $t+\tau$. The

corresponding log-likelihood is then: $1 - \log\left(e^{-\lambda_g \cdot (\sum_{j \in N_i^{Inf}(t)} a_{ij}(t)) \cdot \tau}\right)$

3. The person is infected at time t and stays infected at time $t + \tau$. The corresponding log-likelihood is then: $\log(e^{-\mu \cdot \tau})$

4. The person is infected at time t and recovers at time $t+\tau$. The corresponding log-likelihood is then: $1 - \log(e^{-\mu \cdot \tau})$

5. The person is infected at time t and stays infected at time $t + \tau$, but exceeds $M \cdot \tau$ (which is the time to death). The corresponding log-likelihood is then:0. This is because the probability of dying is 1 (under the assumption of this death model).

All the log-likelihood values of the different cases are then summed together to get the final log-likelihood at time t and then we sum this over all the time steps and get the final log-likelihood.
In order to produce the simulation results for the estimation of the parameters λ_g and μ in section 7, the model has been simulated 100 times and each time we estimate the parameters λ_g and μ. In each iteration, we will look at a 'fixed' data set and consider a range of values of λ_g and μ. The range of the values of λ_g and μ are in the interval (0,1) with a mesh-width of 0.01.
We iterate through these values of both λ_g and μ and calculate the corresponding log-likelihood. All these values of the log-likelihoods are stored in a list (in the program it is named *'difvaluesloglh'*). From this list the maximum value is then taken, using the *max* function in Python. This is because the log-likelihood should be maximal around the 'real' parameters.

To this maximum value of the log-likelihood corresponds a pair of λ_g and μ and that pair will be the estimated values for the parameters λ_g and μ (this pair of values are called *'max_x'* in the code).
Every iteration will give a pair of estimated values for the parameters λ_g and μ with a corresponding log-likelihood. All these values are then stored in another list (in the program called '*simulatedlamhatmu*' and '*simulatedlog-likelihood*'). From these values we are going to take the average. This average should then converge to the real parameters used in the data. In this case the real parameters in the simulation data were $\lambda_g = 0.5$ and $\mu = 0.1$.

13.2 Implementation of the Metropolis-Hastings algorithm

In this section, the method and the corresponding python implementation will be explained for this particular mathematical model to estimate the parameters λ_g and μ. The aim of the algorithm is to sample from the posterior probability density given by:

$$p_{\Theta|X=x}(\theta) = \frac{p_{X,\Theta}(x,\theta)}{p_X(x)} = \frac{p_\theta(x)\cdot\pi(\theta)}{\int p_\vartheta(x)\cdot\pi(\vartheta)d\vartheta} \tag{39}$$

where $p_\theta(x)$ is the probability density of the observation x and π is the prior density of the parameter θ, in order to do Bayesian parameter estimation.
We begin by generating a data set and considering the log-likelihood function again. To start the algorithm we take initial values for the parameters we want to estimate, which are in this case λ_g and μ. That is λ_g^0 and μ^0, where the ^{0} superscript is for the 0^{th} iteration. It is to be noted that in this case both values of λ_g and μ are positive, so negative initial estimates are excluded.
The proposal distribution that has been chosen when performing the algorithm is a normal distribution centered around the previous estimated value of λ_g or μ respectively and with a variance of σ^2. This variance can be tuned to a prespecified. However, since generally we want an acceptance rate between 0.25 and 0.5, in this simulation σ^2 is taken to be equal to 1. In the list *lamhatlist* the values of the λ_g parameter are stored and in the list *mulist* the values of the μ parameter are stored when the algorithm is performed. In order to perform the algorithm we will make use of a 'simplified posterior density', which is proportional to the posterior density given by:

$$p_{\Theta|X=x}(\theta) = \frac{p_{X,\Theta}(x,\theta)}{p_X(x)} = \frac{p_\theta(x)\cdot\pi(\theta)}{\int p_\vartheta(x)\cdot\pi(\vartheta)d\vartheta} \tag{40}$$

where $p_\theta(x)$ is the probability density of the observation x and π is the prior density of the parameter θ. Since the denominator of the posterior density is often hard to compute (definitely in this case), we make use of another function $g(\theta)$ which is proportional to the posterior density.

$$p_{\Theta|X=x}(\theta) = \frac{p_\theta(x)\cdot\pi(\theta)}{\int p_\vartheta(x)\cdot\pi(\vartheta)d\vartheta} \propto p_\theta(x)\cdot\pi(\theta) = g(\theta) \tag{41}$$

The prior density that has been chosen for the parameter vector $\theta = (\lambda_g, \mu)$ is the product of the two prior densities of the parameters λ_g and μ. The reason why this has been chosen is because it is assumed that the infection probability rate λ_g is independent of the recovery probability rate μ. For both parameters it is assumed

that they are exponentially distributed with parameter 1. The prior function then becomes:

$$\pi(\theta) = e^{-\lambda_g} \cdot e^{-\mu}(\mathbb{1}_{\{x_{i,k-1}=I\}} + \mathbb{1}_{\{x_{i,k-1}=S\}}) = e^{-(\lambda_g+\mu)}(\mathbb{1}_{\{x_{i,k-1}=I\}} + \mathbb{1}_{\{x_{i,k-1}=S\}}) \quad (42)$$

Recall that the likelihood was given by equation (26) which we derived in section 6.

Thus, all together the function $g(\theta)$ is given by:

$$g(\theta) = e^{-(\lambda_g+\mu)}(\mathbb{1}_{\{x_{i,k-1}=I\}} + \mathbb{1}_{\{x_{i,k-1}=S\}}) \cdot (\prod_{i=1}^{n} \mathbb{P}((X_{i,k}, Z_{i,k} = (x_{i,k}, z_{i,k})|(\mathbf{Y_{k-1}}, \mathbf{Z_{k-1}}) = (\mathbf{y_{k-1}}, \mathbf{z_{k-1}}))) \quad (43)$$

When performing the Metropolis-Hastings algorithm, a ratio needs to be calculated in every iteration . This ratio for the j^{th} iteration is defined as:

$$\alpha = \frac{g(\theta^*) \cdot q(\theta^{j-1}|\theta^*)}{g(\theta^{j-1}) \cdot q(\theta^*|\theta^{j-1})}$$

where θ^{j-1} is in this case a vector of the parameters $(\lambda_g^{j-1}, \mu^{j-1})$ of the previous iteration,θ^* is the vector of the parameters that are proposed in this iteration so (λ_g^*, μ^*), g is a function proportional to the posterior density and q represents the proposal density.
To compute $g(\theta^*)$ and $g(\theta^{j-1})$ we simply substitute the star values and the previous values of the parameters respectively into the expression given in equation (43). To compute the ratio of the proposal densities we need to do some more calculations. The proposal densities are assumed to be normally distributed around the previous value of θ with a variance of σ^2. Hence, we get the following, using the fact that $\frac{\log(\theta^*)-\log(\theta^{j-1})}{\sigma} \sim N(0,1)$. Define $Z \sim N(0,1)$.

$$\begin{aligned}
\mathbb{P}(\theta^* \leq z|\theta^{j-1}) &= \mathbb{P}(\log(\theta^*) \leq \log(z)|\theta^{j-1}) \\
&= \mathbb{P}(\frac{\log(\theta^*) - \log(\theta^{j-1})}{\sigma} \leq \frac{\log(z) - \log(\theta^{j-1})}{\sigma}|\theta^{j-1}) \\
&= \mathbb{P}(Z \leq \frac{\log(z) - \log(\theta^{j-1})}{\sigma}|\theta^{j-1}) \\
&= \Phi(\frac{\log(z) - \log(\theta^{j-1})}{\sigma})
\end{aligned} \quad (44)$$

where the Φ represents the standard normal cumulative distribution function.
To get the proposal density conditioned on the previous value, we need to differentiate equation (44) with respect to z and evaluate it at the value we want, which

is in this case θ^*.

$$\begin{aligned} q(\theta^*|\theta^{j-1}) &= \frac{d}{dz}(\Phi(\frac{\log(z)-\log(\theta^{j-1})}{\sigma})\Big|_{z=\theta^*}) \\ &= \phi(\frac{\log(z)-\log(\theta^{j-1})}{\sigma})\Big|_{z=\theta^*} \cdot \frac{d}{dz}(\frac{\log(z)-\log(\theta^{j-1})}{\sigma})\Big|_{z=\theta^*} \\ &= \phi(\frac{\log(z)-\log(\theta^{j-1})}{\sigma})\Big|_{z=\theta^*} \cdot \frac{1}{z} \cdot \frac{1}{\sigma}\Big|_{z=\theta^*} \\ &= \phi(\frac{\log(\theta^*)-\log(\theta^{j-1})}{\sigma}) \cdot \frac{1}{\theta^*} \cdot \frac{1}{\sigma} \end{aligned} \tag{45}$$

where ϕ is the probability density function of the standard normal distribution. In order to get the value α, we need to evaluate the ratio

$$\frac{q(\theta^{j-1}|\theta^*)}{q(\theta^*|\theta^{j-1})}.$$

By using the expression in equation (45), this ratio can then be simplified as follows:

$$\begin{aligned} \frac{q(\theta^{j-1}|\theta^*)}{q(\theta^*|\theta^{j-1})} &= \frac{\phi(\frac{\log(\theta^{j-1})-\log(\theta^*)}{\sigma}) \cdot \frac{1}{\theta^{j-1}} \cdot \frac{1}{\sigma}}{\phi(\frac{\log(\theta^*)-\log(\theta^{j-1})}{\sigma}) \cdot \frac{1}{\theta^*} \cdot \frac{1}{\sigma}} \\ &= \frac{\theta^*}{\theta^{j-1}} \end{aligned} \tag{46}$$

The simplification makes use of the fact that the Gaussian distribution is symmetric. In Appendix 12, the complete expressions of the cumulative distribution function and the density of a standard normal distribution are given.
The other ratio that needs to be evaluated to get the value of α is:

$$\frac{g(\theta^*)}{g(\theta^{j-1})}$$

In order to get the ratio of the function $g(\theta)$, note that the function given in equation (43) is the product of two functions: the likelihood and the prior.
We perform the Metropolis-Hastings algorithm first for the parameter λ_g and then for the parameter μ. So the Metropolis-Hastings algorithm is then as follows:

1. Select an initial value θ^0, so in this case for λ_g^0 and μ^0.

2. For $j = 1, \ldots, m$ (where m is some large integer number), we repeat the following:

(a) Draw the candidate θ^* from the proposal distribution $q(\theta^*|\theta^{j-1})$. We first keep μ^{j-1} constant and use the proposed value of λ_g^* in the calculation of the ratio of α. The parameter vector θ^* is now equal to $\theta^* = (\lambda_g^*, \mu^{j-1})^T$ and the parameter vector $\theta^{j-1} = (\lambda_g^{j-1}, \mu^{j-1})^T$.

(b) Compute the following ratio:

$$\alpha = \frac{g(\theta^*) \cdot q(\theta^{j-1}|\theta^*)}{g(\theta^{j-1}) \cdot q(\theta^*|\theta^{j-1})}$$

(c) Check α for the value of λ_g:

- if $\alpha \geq 1$, we accept the candidate θ^* and set the value of $\lambda_g^j = \lambda_g^*$
- if $0 < \alpha < 1$, we accept the candidate θ^* and set the value of $\lambda_g^j = \lambda_g^*$ with probability α. We do this by drawing a sample u^j from the uniform distribution $U(0,1)$. If $u^j < \alpha$ we set $\lambda_g^j = \lambda_g^*$ and if $u^j > \alpha$ we set $\lambda_g^j = \lambda_g^{j-1}$.

(d) Now we keep the value of λ_g^j constant and use the proposed value for μ, so μ^* and compare that to the value of μ^{j-1} in the calculation of the ratio α. The parameter vector θ^* is now $\theta^* = (\lambda_g^j, \mu^*)^T$ and the parameter vector $\theta^{j-1} = (\lambda_g^j, \mu^{j-1})^T$.

(e) We calculate:

$$\alpha = \frac{g(\theta^*) \cdot q(\theta^{j-1}|\theta^*)}{g(\theta^{j-1}) \cdot q(\theta^*|\theta^{j-1})}$$

(f) Check α for the value of μ:

- if $\alpha \geq 1$, we accept the candidate θ^* and set the value of $\mu^j = \mu^*$
- if $0 < \alpha < 1$, we accept the candidate θ^* and set the value of $\mu^j = \mu^*$ with probability α.
  We do this by drawing a sample u^j from the uniform distribution $U(0,1)$. If $u^j < \alpha$ we set $\mu^j = \mu^*$ and if $u^j > \alpha$ we set $\mu^j = \mu^{j-1}$.

13.3 Implementation of MALA

The implementation of MALA is similar to the implementation of the Metropolis-Hastings algorithm. The only difference is that instead of drawing a candidate from a general distribution family (such as the normal distribution or exponential distribution), we fix the proposal density to be Gaussian and use the Langevin diffusion stochastic differential equation. The Langevin diffusion stochastic differential equation is given in equation (33).
The candidate in our case is then drawn as follows:

$$\theta^* = \theta^j + h \cdot \boldsymbol{\nabla} \log(g(\theta^j)) + \sqrt{2h}\zeta^j \tag{47}$$

where ζ^j is drawn from a normal distribution with mean zero and variance σ^2 and g is the prior function times the likelihood function in this case, which is given in equation (43). The superscript ^{j} denotes the j^{th} iteration of the algorithm.
For the acceptance ratio of α we need to consider the proposal density, just like in the Metropolis-Hastings algorithm. We now have that:

$$\frac{\theta^* - (\theta^{j-1} + h \cdot \boldsymbol{\nabla} \log(g((\theta^{j-1})))}{\sqrt{2h}} \sim N(0,1) \tag{48}$$

If we define that $Z \sim N(0,1)$ then the following holds:

$$\begin{aligned}
\mathbb{P}(\theta^* \leq z | \theta^{j-1}) &= \mathbb{P}(\frac{\theta^* - (\theta^{j-1} + h \cdot \boldsymbol{\nabla} \log(g(\theta^{j-1})))}{\sqrt{2h}} \leq \frac{z - (\theta^{j-1} + h \cdot \boldsymbol{\nabla} \log(g(\theta^{j-1})))}{\sqrt{2h}} | \theta^{j-1}) \\
&= \mathbb{P}(Z \leq \frac{z - (\theta^{j-1} + h \cdot \boldsymbol{\nabla} \log(g(\theta^{j-1})))}{\sqrt{2h}} | \theta^{j-1}) \\
&= \Phi\left(\frac{z - (\theta^{j-1} + h \cdot \boldsymbol{\nabla} \log(g(\theta^{j-1})))}{\sqrt{2h}}\right)
\end{aligned} \tag{49}$$

where Φ is the cumulative distribution function of a standard normal random variable given in equation (37).
In order to get the proposal density $q(\theta^* | \theta^{j-1})$ we differentiate equation (49) again with respect to z and get the following:

$$\begin{aligned}
q(\theta^* | \theta^{j-1}) &= \frac{d}{dz}(\Phi(\frac{z - (\theta^{j-1} + h \cdot \boldsymbol{\nabla} \log(g(\theta^{j-1})))}{\sqrt{2h}})\Big|_{z=\theta^*}) \\
&= \phi(\frac{z - (\theta^{j-1} + h \cdot \boldsymbol{\nabla} \log(g(\theta^{j-1})))}{\sqrt{2h}})\Big|_{z=\theta^*} \cdot \frac{d}{dz}(\frac{z - (\theta^{j-1} + h \cdot \boldsymbol{\nabla} \log(g(\theta^{j-1})))}{\sqrt{2h}})\Big|_{z=\theta^*} \\
&= \phi\left(\frac{\theta^* - (\theta^{j-1} + h \cdot \boldsymbol{\nabla} \log(g(\theta^{j-1})))}{\sqrt{2h}}\right) \cdot \frac{1}{\sqrt{2h}}
\end{aligned} \tag{50}$$

where ϕ is the probability density function of a standard normal random variable given in equation (38).

The value of the ratio α is then calculated using:

$$\alpha = \frac{g(\theta^*) \cdot q(\theta^{j-1}|\theta^*)}{g(\theta^{j-1}) \cdot q(\theta^*|\theta^{j-1})}.$$

The function g is the prior times the likelihood in the model given in equation (43) and the function q is as we derived above in equation (50).
Just like in the case when performing Metropolis-Hastings, we are going to take the natural logarithm of α and therefore get:

$$\begin{aligned}
\log(\alpha) &= \log\left(\frac{g(\theta^*) \cdot q(\theta^{j-1}|\theta^*)}{g(\theta^{j-1}) \cdot q(\theta^*|\theta^{j-1})}\right) \\
&= \log\left(\frac{g(\theta^*)}{g(\theta^{j-1})}\right) + \log\left(\frac{q(\theta^{j-1}|\theta^*)}{q(\theta^*|\theta^{j-1})}\right) \\
&= \log(g(\theta^*)) - \log(g(\theta^{j-1})) + \log(q(\theta^{j-1}|\theta^*)) - \log(q(\theta^*|\theta^{j-1})) \\
&= \text{loglh}(\theta^*) - \text{loglh}(\theta^{j-1}) + (\lambda_g^{j-1} - \lambda_g^*) + (\mu^{j-1} - \mu^*) \\
&+ \log\left(\phi\left(\frac{\theta^{j-1} - (\theta^* + h \cdot \boldsymbol{\nabla}\log(g(\theta^*))}{\sqrt{2h}}\right) \cdot \frac{1}{\sqrt{2h}}\right) \\
&- \log\left(\phi\left(\frac{\theta^* - (\theta^{j-1} + h \cdot \boldsymbol{\nabla}\log(g(\theta^{j-1}))}{\sqrt{2h}}\right) \cdot \frac{1}{\sqrt{2h}}\right)
\end{aligned} \tag{51}$$

The value of the expression derived in equation (51) will then again be compared to a value $\log(u)$, where u is drawn from a standard uniform distribution on the interval (0,1). The MALA algorithm is first performed on the value of λ_g, while keeping the value of μ constant. Then we reverse this process by taking λ_g constant and performing the MALA on the value of μ. The rest of the implementation is exactly the same as it was for the Metropolis-Hastings algorithm, which is explained in the previous section.
Therefore, MALA is as follows:

1. Select an initial value θ^0, so in this case for λ_g^0 and μ^0.
2. For $j = 1, \ldots, m$ (where m is some large integer number), we repeat the following:
   (a) Draw the candidate θ^* from the proposal distribution $q(\theta^*|\theta^{j-1})$. Since we first start with λ_g and keep μ constant we only generate a proposed value for λ_g. The proposed value of λ_g is:

$$\lambda_g^* = \lambda^{j-1} + h \cdot \boldsymbol{\nabla}\log(g(\theta^{j-1})) + \sqrt{2h}\zeta_1^{j-1}$$

where $\zeta_1^{j-1} \sim N(0,1)$. We keep μ^{j-1} constant and use the proposed value of λ_g^* in the calculation of the ratio of α. The parameter vector θ^* is now equal to $\theta^* = (\lambda_g^*, \mu^{j-1})^T$ and the parameter vector $\theta^{j-1} = (\lambda_g^{j-1}, \mu^{j-1})^T$.

(b) Compute the following ratio:

$$\alpha = \frac{g(\theta^*) \cdot q(\theta^{j-1}|\theta^*)}{g(\theta^{j-1}) \cdot q(\theta^*|\theta^{j-1})}$$

(c) Check α for the value of λ_g:

- if $\alpha \geq 1$, we accept the candidate θ^* and set the value of $\lambda_g^j = \lambda_g^*$
- if $0 < \alpha < 1$, we accept the candidate θ^* and set the value of $\lambda_g^j = \lambda_g^*$ with probability α. We do this by drawing a sample u^j from the uniform distribution $U(0,1)$. If $u^j < \alpha$ we set $\lambda_g^j = \lambda_g^*$ and if $u^j > \alpha$ we set $\lambda_g^j = \lambda_g^{j-1}$.

(d) We propose a value of μ. The proposed value of μ is:

$$\mu^* = \mu^{j-1} + h \cdot \boldsymbol{\nabla} \log(g(\theta^{j-1})) + \sqrt{2h}\zeta_2^{j-1}$$

where $\zeta_2^{j-1} \sim N(0,1)$.

(e) Now we keep the value of λ_g^j constant and use the proposed value for μ, so μ^* and compare that to the value of μ^{j-1} in the calculation of the ratio α. The parameter vector θ^* is now $\theta^* = (\lambda_g^j, \mu^*)^T$ and the parameter vector $\theta^{j-1} = (\lambda_g^j, \mu^{j-1})^T$.

(f) We calculate:

$$\alpha = \frac{g(\theta^*) \cdot q(\theta^{j-1}|\theta^*)}{g(\theta^{j-1}) \cdot q(\theta^*|\theta^{j-1})}$$

(g) Check α for the value of μ:

- if $\alpha \geq 1$, we accept the candidate θ^* and set the value of $\mu^j = \mu^*$
- if $0 < \alpha < 1$, we accept the candidate θ^* and set the value of $\mu^j = \mu^*$ with probability α.
  We do this by drawing a sample u^j from the uniform distribution $U(0,1)$. If $u^j < \alpha$ we set $\mu^j = \mu^*$ and if $u^j > \alpha$ we set $\mu^j = \mu^{j-1}$.

References

[1] C. Wang, P. W. Horby, F. G. Hayden, and G. F. Gao, "A novel coronavirus outbreak of global health concern," *The Lancet*, vol. 395, no. 10223, pp. 470–473, 2020.

[2] W. E. Committee *et al.*, "Statement on the second meeting of the international health regulations (2005) emergency committee regarding the outbreak of novel coronavirus (covid-19). geneva: Who, 2020," 2005.

[3] worldometer. *COVID-19 CORONAVIRUS PANDEMIC*. [Online]. Available: https://www.worldometers.info/coronavirus/?utm_campaign=homeAdvegas1?%22%20%5Cl%20%22countries

[4] WHO. (2020) *Naming the coronavirus disease (COVID-19) and the virus that causes it*. [Online]. Available: https://www.who.int/emergencies/diseases/novel-coronavirus-2019/technical-guidance/naming-the-coronavirus-disease-(covid-2019)-and-the-virus-that-causes-it

[5] W. H. Organisation. (2020, March) *Media Statement: Knowing the risks for COVID-19*. [Online]. Available: https://www.who.int/indonesia/news/detail/08-03-2020-knowing-the-risk-for-covid-19

[6] Google. (2020, Sep) *Coronavirus (COVID-19)*. [Online]. Available: https://news.google.com/covid19/map?hl=en-US

[7] McKinsey and Company. (2020, Sep) *COVID-19 recovery in hardest-hit sectors could take more than 5 years*. [Online]. Available: https://www.mckinsey.com/featured-insights/coronavirus-leading-through-the-crisis/charting-the-path-to-the-next-normal/covid-19-recovery-in-hardest-hit-sectors-could-take-more-than-5-years

[8] B. Ivorra, M. R. Ferrández, M. Vela-Pérez, and A. Ramos, "Mathematical modeling of the spread of the coronavirus disease 2019 (covid-19) taking into account the undetected infections. the case of china," *Communications in nonlinear science and numerical simulation*, p. 105303, 2020.

[9] F. Verachi, L. G. Trussoni, and L. Lanzi, "Covid-19 in italy: a mathematical model to analyze the epidemic containment strategy and the economic impacts," *medRxiv*, 2020. [Online]. Available: https://www.medrxiv.org/content/early/2020/05/30/2020.05.28.20115790

[10] S. Abrams, J. Wambua, E. Santermans, L. Willem, E. Kuylen, P. Coletti, P. Libin, C. Faes, O. Petrof, S. A. Herzog, , P. Beutels,

and N. Hens, "Modeling the early phase of the belgian covid-19 epidemic using a stochastic compartmental model and studying its implied future trajectories," *medRxiv*, 2020. [Online]. Available: https://www.medrxiv.org/content/early/2020/07/01/2020.06.29.20142851

[11] L. J. Allen, "A primer on stochastic epidemic models: Formulation, numerical simulation, and analysis," *Infectious Disease Modelling*, vol. 2, no. 2, pp. 128 – 142, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2468042716300495

[12] L. Lan, D. Xu, G. Ye, C. Xia, S. Wang, Y. Li, and H. Xu, "Positive RT-PCR Test Results in Patients Recovered From COVID-19," *JAMA*, vol. 323, no. 15, pp. 1502–1503, 04 2020. [Online]. Available: https://doi.org/10.1001/jama.2020.2783

[13] M. van Algemene Zaken. *Dutch measures against coronavirus: basic rules for everyone*. [Online]. Available: https://www.government.nl/topics/coronavirus-covid-19/tackling-new-coronavirus-in-the-netherlands/basic-rules-for-everyone

[14] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 5595–5637, 2017.

[15] D. Fanelli and F. Piazza, "Analysis and forecast of covid-19 spreading in china, italy and france," *Chaos, Solitons & Fractals*, vol. 134, p. 109761, 2020.

[16] C. Yang and J. Wang, "A mathematical model for the novel coronavirus epidemic in wuhan, china," *Mathematical Biosciences and Engineering*, vol. 17, no. 3, pp. 2708–2724, 2020.

[17] D. Caccavo, "Chinese and italian covid-19 outbreaks can be correctly described by a modified sird model," *medRxiv*, 2020.

[18] M. Al-Raeei, "The forecasting of covid-19 with mortality using sird epidemic model for the united states, russia, china, and the syrian arab republic," *AIP Advances*, vol. 10, no. 6, p. 065325, 2020.

[19] K. Rajagopal, N. Hasanzadeh, F. Parastesh, I. I. Hamarash, S. Jafari, and I. Hussain, "A fractional-order model for the novel coronavirus (covid-19) outbreak," *Nonlinear Dynamics*, vol. 101, no. 1, pp. 711–718, 2020.

[20] F. Agel. *Antibodies, immunity low after COVID-19 recovery*. [Online]. Available: https://www.dw.com/en/coronavirus-antibodies-immunity/a-54159332

[21] I. Cooper, A. Mondal, and C. G. Antonopoulos, "A sir model assumption for the spread of covid-19 in different communities," *Chaos, Solitons & Fractals*, vol. 139, p. 110057, 2020.

[22] N. Fernandes, "Economic effects of coronavirus outbreak (covid-19) on the world economy," *Available at SSRN 3557504*, 2020.

[23] I. Chakraborty and P. Maity, "Covid-19 outbreak: Migration, effects on society, global environment and prevention," *Science of The Total Environment*, vol. 728, p. 138882, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0048969720323998