Project 5 in FYS3150

Bendik Steinsvåg Dalen, Ulrik Seip

December 14, 2018

Abstract

In this report have studied the SIRS-model for the spreading of an disease. This has been done using both 4th. order Runge Kutta method and the Monte Carlo method. In addition we will added other aspects to the model to also simulate vital dynamics, variance in the infection rate, and vaccination. We found that the Runge Kutta method is less accurate if the model approaches zero, and our implementation of the Monte Carlo method is impractical for larger populations.

1 INTRODUCTION

In this project we are going to study the SIRS method, which is a popular method for studying the spread of a infectious disease. In the SIRS model we observe an isolated population of N individuals which are divided into three separate groups:

- S for susceptible individuals, people who can become infected from the disease,
- I for infected individuals, people who currently have the disease,
- R for recovered individuals, people who no longer are sick and currently are immune to the disease.

A person can move from S, to I, to R, and back to S again, as they become sick, recover from the disease, and then loose their immunity. This creates a cyclical nature to the spread of the disease, giving us the ability to predict if a certain disease will be able to establish itself permanently in the population.

One problem the SIRS method is that it does not have a analytical solution. We therefore have to use a numerical approximation to solve it. We will be using a 4th. order Runge Kutta approximation, as well as creating a Monte Carlo simulation of the system.

The 4th. order Runge Kutta approximation is a standard method for solving coupled differential equations. One problem with using Runga Kutta is that we must assume S, I, and R are continuous variables, while they in reality are discrete. This may lead to us receiving end results that are not whole numbers, we might see a resurgence of a disease even though the total number of infected people where less than one, or other unrealistic events. It is therefore useful to keep in mind that Runge Kutta is only an approximation, and might not mirror reality to a high degree.

Analysing a system using the Monte Carlo method gives us data that is much closer to what we observe on real life because it is not merely an approximation of the average solution to the problem, but rather a simulation whose average approximates the solution. This means that we get data like the standard deviation of our system, which tells us which numbers are more or less probable. The world is not an ideal system, so using the most precise numerical methods isn't necessarily the best idea, because all they can give you is an answer based on the parameters you give them. With more complex problems, where several parameters are unknown, but we have a statistical probability we can use the Monte Carlo method instead.

2 METHOD

2.a The SIRS method

In the SIRS model an individual can move from S to I, from I to R, and from R to S. The rate at which they do this will be referred to as a, b and c respectively. When an susceptible individual is in contact with an infected individual there is a chance that they become infected, while the rate of recovery and the rate of immunity loss is independent from the other groups. This can then be written as three coupled differential equations:

$$\begin{aligned} S' &= cR - \frac{aSI}{N} \\ I' &= \frac{aSI}{N} - bI \\ R' &= bI - cR \end{aligned} \tag{1}$$

The total population $N = S(t) + I(t) + R(t)$ will at least for now remain constant. This set does not have a analytical solution, but we can easily study the equilibrium solutions. By setting all equations in set 1 to 0 we get

$$\begin{aligned} s^* &= \frac{b}{a}, \\ i^* &= \frac{1 - \frac{b}{a}}{1 + \frac{b}{c}}, \\ r^* &= \frac{b}{c}\frac{1 - \frac{b}{a}}{1 + \frac{b}{c}}, \end{aligned} \tag{2}$$

where s, i and r denotes the fraction of people in S, I and R, respectively, and * means that they are at equilibrium. These should add up to 1, and be between 0 and 1. One thing we can denote from this is that we need $b < a$ for the disease to establish itself permanently in the population. However if $b > a$ the model brakes, as s^* becomes larger than 1, and i^* and r^* becomes less than zero. This makes sense, as this would make the rate of recovery larger than the rate of infection, and the disease wouldn't be able to spread. Equation 2 will serve as a good comparison for our numerical approximations.

2.b Runge Kutta

To solve equation 1 we will first be using the 4th. order Runge Kutta method. From ([1]) we know that one cycle of the method is

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2} k_1) \\ k_3 &= f(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2} k_2) \\ k_4 &= f(t_n + \Delta t, y_n + \Delta t k_3) \\ y_{n+1} &= y_n + \frac{\Delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \end{aligned} \tag{3}$$

where y_n is the current step, y_{n+1} is the next step, t_n is the current time, Δt is the time step, and f is the differential equation.

We will be observing four different populations A, B, C and D. All of them are of size $N = 400$, and we start with 300 susceptible individuals and 100 infected individuals. We also set $a = 4$ and $c = 0.5$, but b will vary from 1 to 4 respectively. We will set $\Delta t = 0.01 days$ and run the simulation until the equilibrium situation has been reached. We will be doing the same for the rest of the Runge Kutta simulations.

We will also study a population E where we set $b = 5$ to se how this affects the situation.

2.c Monte Carlo

A Monte Carlo simulation is based on running a large number of simulations for a system with known probabilities. From these simulations we can extract important statistical information, like the average value of a resulting variable, or the standard deviation of said variable. Because every one of these simulations consist of discrete values they paint a much more realistic picture of the system we are trying to model.

In our SIRS system we have the transitional probability coefficients a, b and c. By cutting our simulation up into sufficiently small time steps we can make sure that at maximum one person can transition from one group to another. We can then make an expression for the probability of said transition, and for every time step dt check if someone goes from one category to another.

Let us start with the expression for dt.

$$(S \rightarrow_{max} I) \Rightarrow \frac{aSI}{N}\Delta t = \frac{aN}{4}\Delta t = 1 \tag{4}$$

$$(I \rightarrow_{max} R) \Rightarrow bI\Delta t = bN\Delta t = 1 \tag{5}$$

$$(R \rightarrow_{max} S) \Rightarrow cI\Delta t = cN\Delta t = 1 \tag{6}$$

The largest Δt we can have must then be

$$\Delta t = \min\left(\frac{4}{aN}, \frac{1}{bN}, \frac{1}{cN}\right). \tag{7}$$

Having found Δt we can now get actual values for the transition probabilities at any given time. For each time step we then generate a random number r, where $r \in [0, 1]$, and if r is smaller than the probability we move one person. This is done once for every possible transition for every time step.

The process that has been described so far equates to a single Monte Carlo run, but to be able to extract useful and precise numbers from the resulting data we need more runs, and so, unless otherwise is stated, we perform a thousand independent runs. We can then use any statistics module to extract the average run, and the standard deviations.

2.d Improving the model

Now that we have a basic model we can extended it to include more details about the population and disease. Following are three different variables that can improve the model to make it more realistic. We are going to implement them independently at first, and then include all of them at the same time.

2.d.i Vital dynamics

First we are going to include vital dynamics, meaning death and birth rate. This can be useful if we are going to model a population over a longer time period. We let e be birth rate, d be death rate and d_i be the death rate from the disease. We assume that individuals from all the groups can give birth, but that children born into the population are initially susceptible. This makes differential equations into

$$\begin{aligned} S' &= cR - \frac{aSI}{N} - dS + eN, \\ I' &= \frac{aSI}{N} - bI - dI - d_I I, \\ R' &= bI - cR - dR. \end{aligned} \tag{8}$$

We tried to base the values for d and e on the death and birth rate in Norway. It can be shown that these become $d = 0.00002242299$ and $e = 0.00002948891$ per day ([2], [3], [4]). We applied these to the populations B, and observed how these were affected by setting $d_i = 0$ and $d_i = 0.1$. We then tried different values for d_i on population A to see if these would result in the entire population dying. Finally, as the values of d and e we found are quite low, we tried seeing how population B would be affected by increasing these by a factor of 10000.

2.d.ii Seasonal Variation

Another concept we can study is seasonal variations in the infection rate. Certain diseases, such as influenza, have a large variance in infection rate based on the time of year. This can be modeled by setting a to be

$$a(t) = A\sin(\omega t) + a_0, \tag{9}$$

where $a0$ is the average transmission rate, A is the maximum deviation from a_0, and ω is the frequency of oscillation. We set $\omega = 2\pi/365.25$ so that a would have a period of a year, like most diseases. First we set $A = 1$ and $a0 = 4$ for population A. Then we observed population B, and set $A = 2$, and $a0 = 4$.

2.d.iii Vaccination

For many diseases a vaccine gets developed. A vaccine causes an susceptible individual to become immune, effectively moving them to the recovered group. We assume that the rate of vaccination f is independent from how many people already have been vaccinated, but that it can vary with time. We also assume that vaccinated individuals loose their immunity at the same rate as people who have recovered from the disease. The differential equations then becomes

$$\begin{aligned} S' &= cR - \frac{aSI}{N} - dS - f, \\ I' &= \frac{aSI}{N} - bI, \\ R' &= bI - cR + f. \end{aligned} \tag{10}$$

Exactly what f is can vary. We will be studying three cases:

- Constant f, meaning $f = f_c S$, where f_c is some constant. We sat $f_c = 1$ for population B. For some diseases a vaccine is offered to children at a specific age, which would constitute a constant vaccination rate. However, these vaccines usually induce a permanent or a long lasting immunity, so this method might not fit well with our model.
- Linear f, meaning $f = (t \cdot f_l + f_0) \cdot I$, where f_0 is the initial value of f, and f_l is the rate at which f increases with time. We sat $f_0 = 0$ and $f_l = 0.01$ for population B. This is meant to model that as awareness and medical research increases, more people take the vaccine and the vaccine becomes more effective. If having this increase linearly is the most realistic way is unknown, but we assume it's a good approximation. By setting $f_0 = 0$ we start the simulation at the point when vaccination starts.
- Vaccination campaign f. This method is meant to model a how many governments respond to an outbreak of an epidemic, by distributing vaccines during a certain time period, often called an vaccination campaign. This can be written as $f = f_c S$ if $t \in [t_0, t_1]$ and $f = f_n S$ if $t \notin [t_0, t_1]$, where $f_c >> f_n$. We sat $f_n = 0$, $f_c = 0.3$, $t_0 = 2$ and $t_1 = 9$ for population B.

2.e more monte carlo

Implementing all these new variables to the Monte Carlo simulation is fairly simple. We have some new transition variables, and so we create new probability expressions, and check if a person is transferred from the relevant category. These probability expressions, with G substituted when the transition can happen to any group, combined with the original ones, become

$$P(S \to I) = \frac{aSI}{\Delta t}, \tag{11}$$
$$P(I \to R) = bI\Delta t, \tag{12}$$
$$P(R \to S) = cR\Delta t, \tag{13}$$
$$P(G_{death}) = dG\Delta t, \tag{14}$$
$$P(I_{infectiondeath)} = d_i G\Delta t, \tag{15}$$
$$P(S_{birth}) = eN\Delta t, \tag{16}$$
$$P(S \to_{vaccination} R) = fS\Delta t. \tag{17}$$

We then apply these probabilities as we did with a, b and c in the Runge Kutta system, and can still let the probabilities vary as described.

These new variables in the simulation do bring one problem though. If the total population exceeds the original one, our calculations for Δt are going to be useless because we then have a system that should let more than one person transition per time step. This means that even if we base Δt on all the transition variables we are still not ensured a working system, and so we need a new method for deciding Δt. We can either make Δt very small, so that we have a large buffer for population growth, whilst taking a hit to performance, or we can go by trail and error until we get consistent results with different values for Δt. In this project we have decided to go with the first alternative, and just let Δt be very small. We found $\Delta t = 10^{-4} days$ to be sufficient.

Because we now have such a small Δt we need to make our code as efficient as possible. In addition calculating the values of variables as few times as possible we can pass the $@inbounds$ argument to skip bounds checking of the for loops. Most importantly though, we can use multithreading to reduce the run time.

3 RESULTS

3.a Runge Kutta

The resulting plots for population A to E can be seen in figure 1 to 5. The resulting end values can also be seen in table 1 to 5.

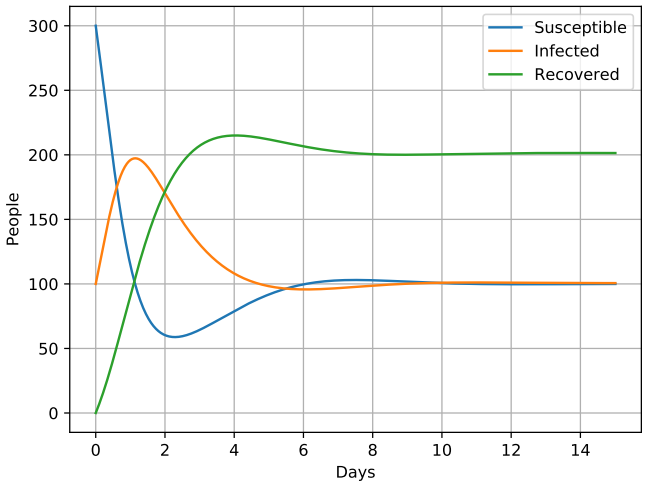

Figure 1: A plot of the population distribution for the SIRS-model using Runge Kutta, for population A, where $a = 4$, $b = 1$ and $c = 0.5$.

Group	Expected	Numerical	Number
s^*	0.25	0.2499	100.0
i^*	0.25	0.2516	100.6
r^*	0.5	0.5035	201.4

Table 1: The corresponding end values for figure 1. The expected values are derived from equation 2, the numerical values are the potion of the population found by RK4, and number are the total number of people this corresponds to.

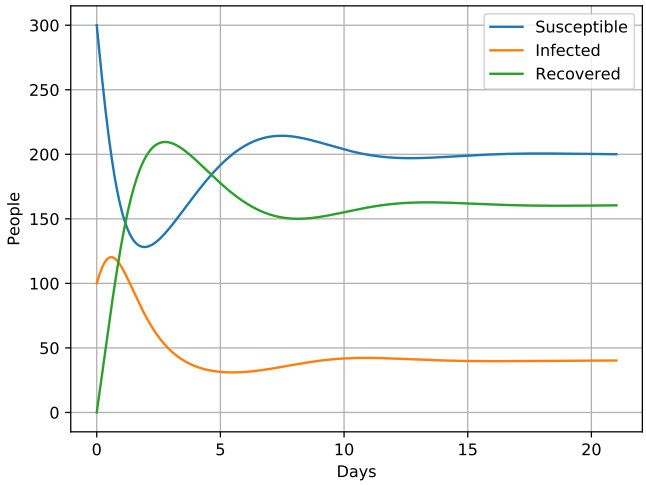

Figure 2: A plot of the population distribution for the SIRS-model using Runge Kutta, for population B, where $a = 4$, $b = 2$ and $c = 0.5$.

Group	Expected	Numerical	Number
s^*	0.5	0.5001	200.1
i^*	0.1	0.1006	40.2
r^*	0.4	0.4011	160.5

Table 2: The corresponding end values for figure 2. The expected values are derived from equation 2, the numerical values are the potion of the population found by RK4, and number are the total number of people this corresponds to.

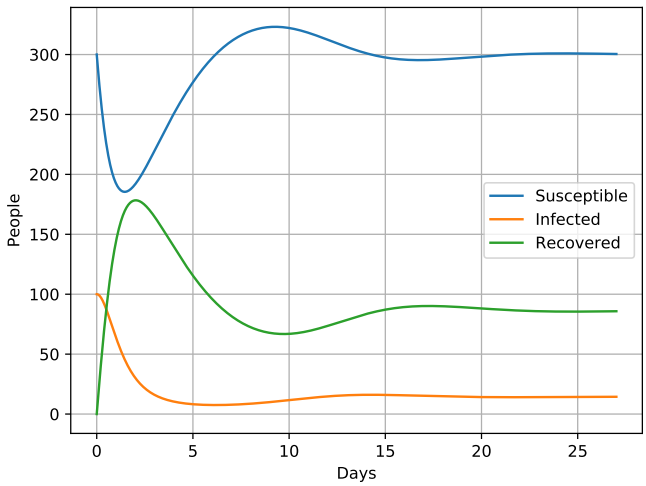

Figure 3: A plot of the population distribution for the SIRS-model using Runge Kutta, for population C, where $a = 4$, $b = 3$ and $c = 0.5$.

Group	Expected	Numerical	Number
s^*	0.75	0.7511	300.5
i^*	0.0357	0.0360	14.4
r^*	0.2143	0.2145	85.8

Table 3: The corresponding end values for figure 3. The expected values are derived from equation 2, the numerical values are the potion of the population found by RK4, and number are the total number of people this corresponds to.

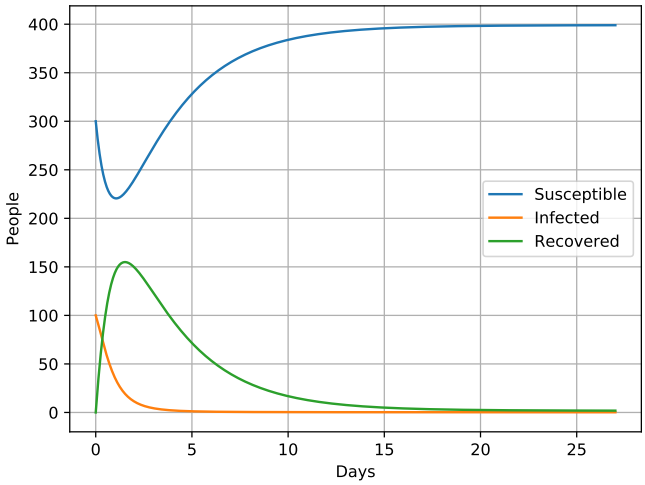

Figure 4: A plot of the population distribution for the SIRS-model using Runge Kutta, for population D, where $a = 4$, $b = 4$ and $c = 0.5$.

Group	Expected	Numerical	Number
s^*	1.0	0.9974	399.0
i^*	0.0	0.0006	0.2
r^*	0.0	0.0047	1.9

Table 4: The corresponding end values for figure 4. The expected values are derived from equation 2, the numerical values are the potion of the population found by RK4, and number are the total number of people this corresponds to.

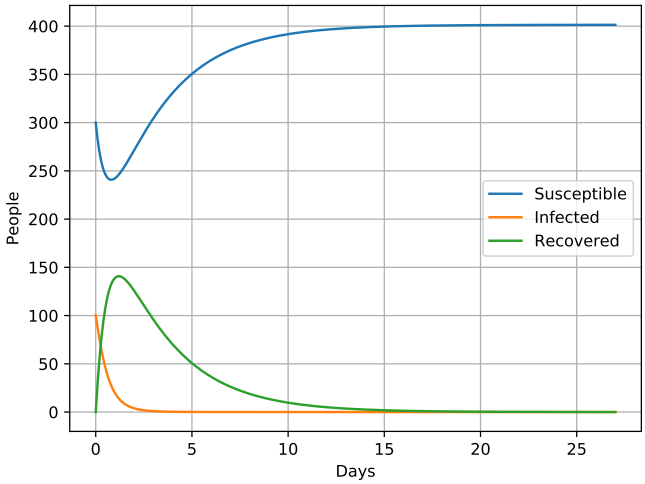

Figure 5: A plot of the population distribution for the SIRS-model using Runge Kutta, for population E, where $a = 4$, $b = 5$ and $c = 0.5$.

Group	Expected	Numerical	Number
s^*	1.25	1.0034	401.4
i^*	-0.0227	$4.7119 \cdot 10^{-11}$	$1.9 \cdot 10^{-8}$
r^*	-0.2273	$8.4064 \cdot 10^{-5}$	$3.4 \cdot 10^{-2}$

Table 5: The corresponding end values for figure 5. The expected values are derived from equation 2, the numerical values are the potion of the population found by RK4, and number are the total number of people this corresponds to.

3.b Monte Carlo

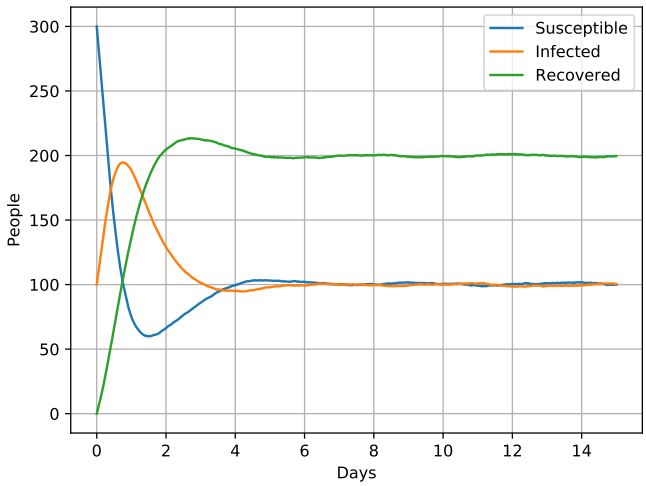

Figure 6: A plot of the population distribution for the SIRS-model using Monte Carlo, for population A, where $a = 4$, $b = 1$ and $c = 0.5$.

Group	Expected	Numerical	Number	σ
s^*	0.25	0.2499	99.98	11.88
i^*	0.25	0.2511	100.43	10.33
r^*	0.5	0.4989	199.60	8.84

Table 6: The corresponding end values and standard deviations for figure 6. The expected values are derived from equation 2, the numerical values are the portion of the population found by the Monte Carlo simulation, and number is the total number of people this corresponds to.

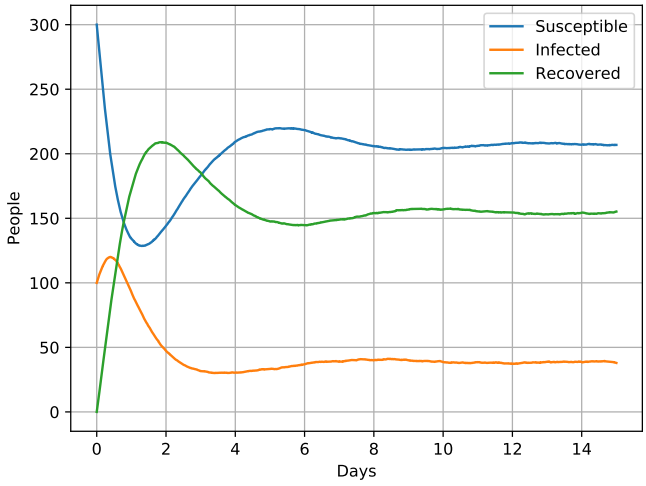

Figure 7: A plot of the population distribution for the SIRS-model using Monte Carlo, for population B, where $a = 4$, $b = 2$ and $c = 0.5$.

Group	Expected	Numerical	Number	σ
s^*	0.5	0.5171	206.85	31.13
i^*	0.1	0.0949	37.94	11.79
r^*	0.4	0.3880	155.21	24.56

Table 7: The corresponding end values and standard deviations for figure 7. The expected values are derived from equation 2, the numerical values are the portion of the population found by the Monte Carlo simulation, and number is the total number of people this corresponds to.

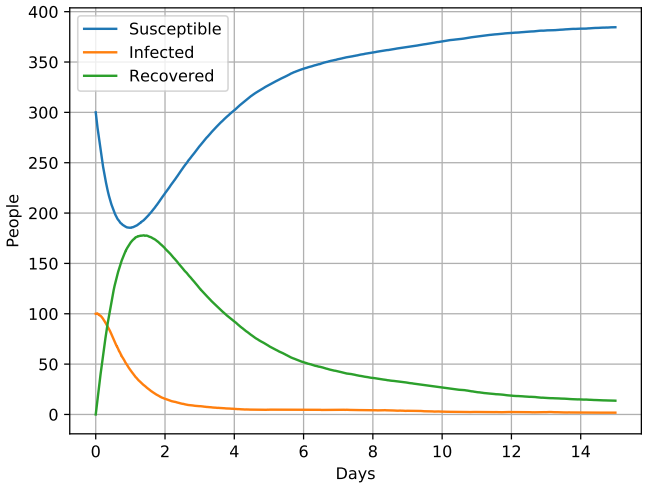

Figure 8: A plot of the population distribution for the SIRS-model using Monte Carlo, for population C, where $a = 4$, $b = 3$ and $c = 0.5$.

Group	Expected	Numerical	Number	σ
s^*	0.75	0.9614	384.55	34.83
i^*	0.04	0.0045	1.78	5.31
r^*	0.21	0.0342	13.67	30.25

Table 8: The corresponding end values and standard deviations for figure 8. The expected values are derived from equation 2, the numerical values are the portion of the population found by the Monte Carlo simulation, and number is the total number of people this corresponds to.

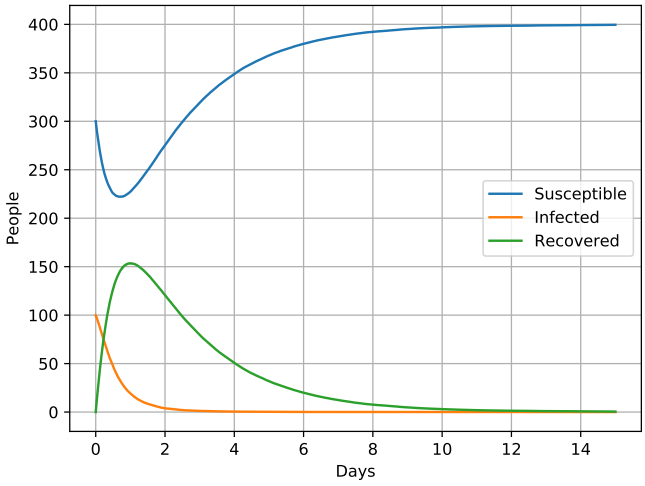

Figure 9: A plot of the population distribution for the SIRS-model using Monte Carlo, for population D, where $a = 4$, $b = 4$ and $c = 0.5$.

Group	Expected	Numerical	Number	σ
s^*	1.0	0.9989	399.55	3.19
i^*	0.0	0.0	0.0	0.0
r^*	0.0	0.0011	0.45	3.19

Table 9: The corresponding end values and standard deviations for figure 9. The expected values are derived from equation 2, the numerical values are the portion of the population found by the Monte Carlo simulation, and number is the total number of people this corresponds to.

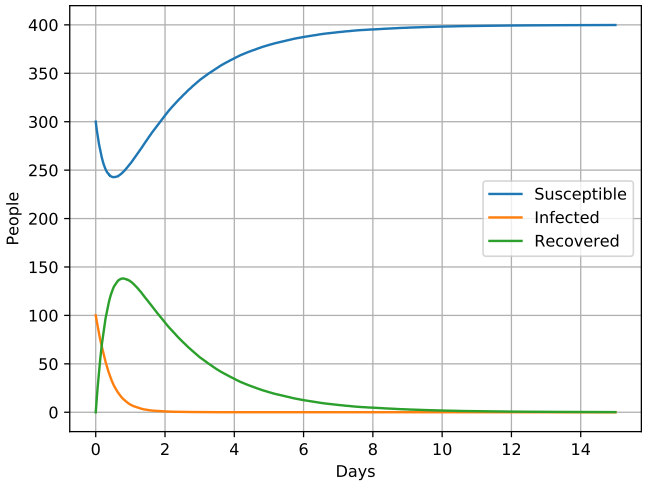

Figure 10: A plot of the population distribution for the SIRS-model using Monte Carlo, for population E, where $a = 4$, $b = 5$ and $c = 0.5$.

Group	Expected	Numerical	Number	σ
s^*	1.25	0.9996	399.84	0.40
i^*	-0.023	0.0	0.0	0.0
r^*	-0.023	0.0004	0.16	0.40

Table 10: The corresponding end values and standard deviations for figure 10. The expected values are derived from equation 2, and clearly break down for this population, the numerical values are the portion of the population found by the Monte Carlo simulation, and number is the total number of people this corresponds to.

3.c Improving the model

3.c.i Vital dynamics

Plots of population B when accounting for vital dynamics, found using Runge Kutta, can be seen in figure 11. In figure 12 you can see plots of population A where d_i either too high or low to realistically kill the entire population, while in figure 13 you can see a plot where d_i is the correct size to kill the entire population. Finally, in figure 14 you can see a plot where d and e have been increased by a factor of 1000000.

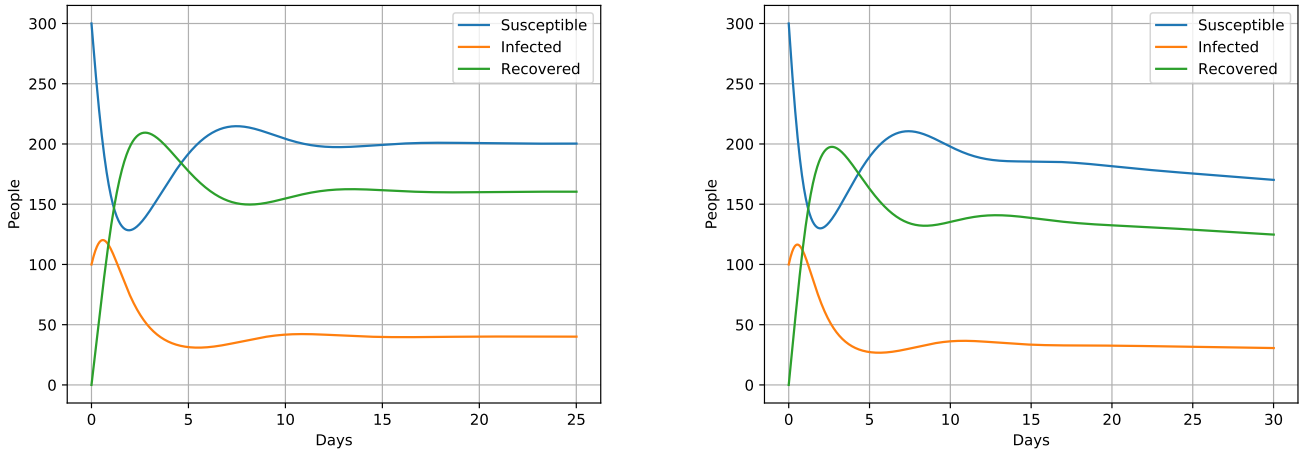

Figure 11: Plots of population B when accounting for vital dynamics, found using Runge Kutta. On the left $d_i = 0$ and on the rigth $d_i = 0.1$.

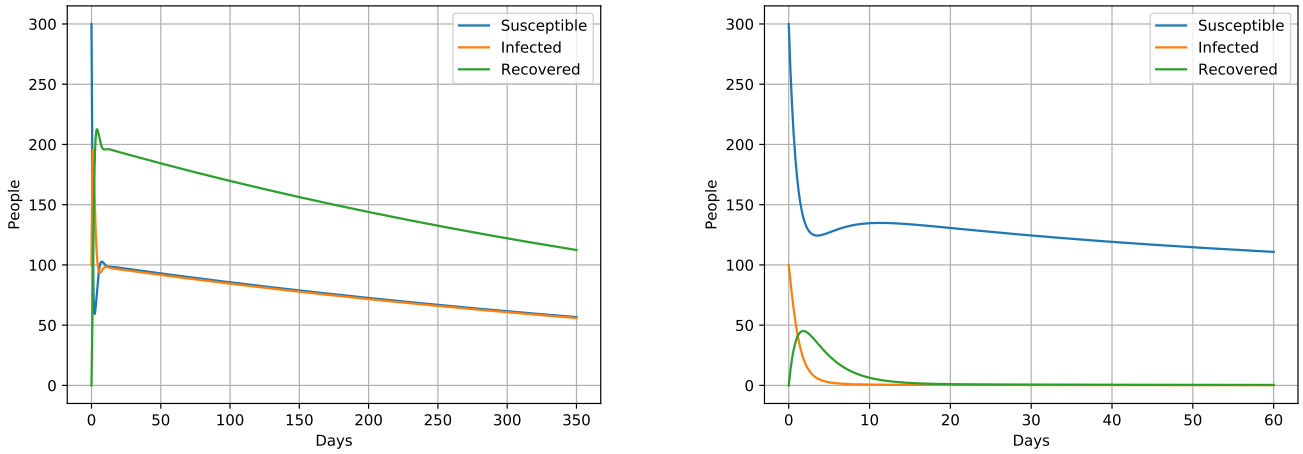

Figure 12: Plots of population A when accounting for vital dynamics, found using Runge Kutta. On the left $d_i = 0.01$ and on the rigth $d_i = 3$. The left d_i is too low to effectively kill the entire population, while the one is too large.

Plots of population B when accounting for vital dynamics, found using Monte Carlo, can be seen in figure 15. In figure 16 you can see plots of population A where d_i either too high or low to realistically kill the entire population, while in figure 17 you can see a plot where d_i is the correct size to in some cases kill the entire population, but definitely reduce its size significantly. Finally, in figure 18 you can see a plot where d and e have been increased by a factor of 1000000.

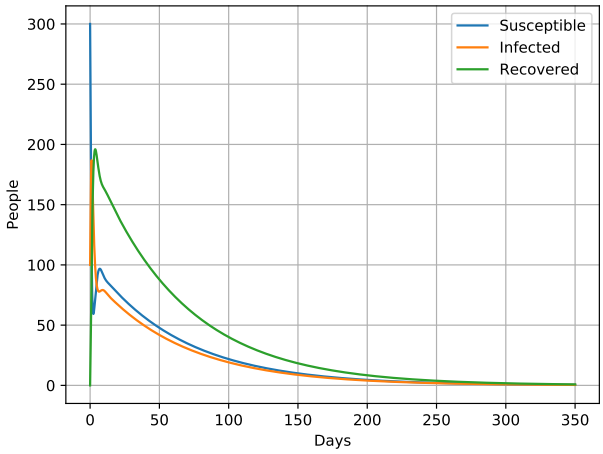

Figure 13: Plots of population A when accounting for vital dynamics, found using Runge Kutta, with $d_i = 0.1$. Here d_i is the correct size to kill the entire population.

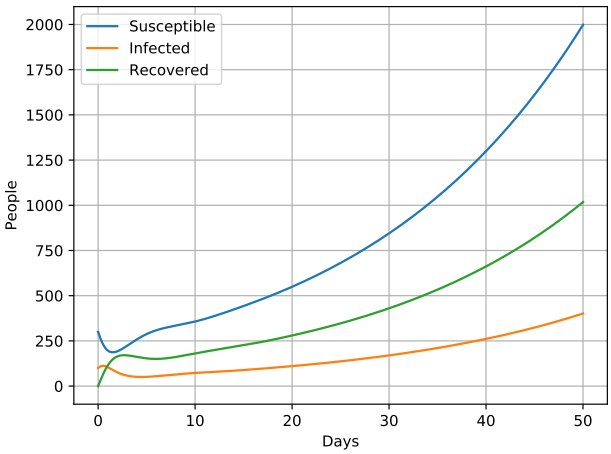

Figure 14: Plots of population B when accounting for vital dynamics, found using Runge Kutta, with $d_i = 0.1$, and d and e are increased by a factor of 1000000 .

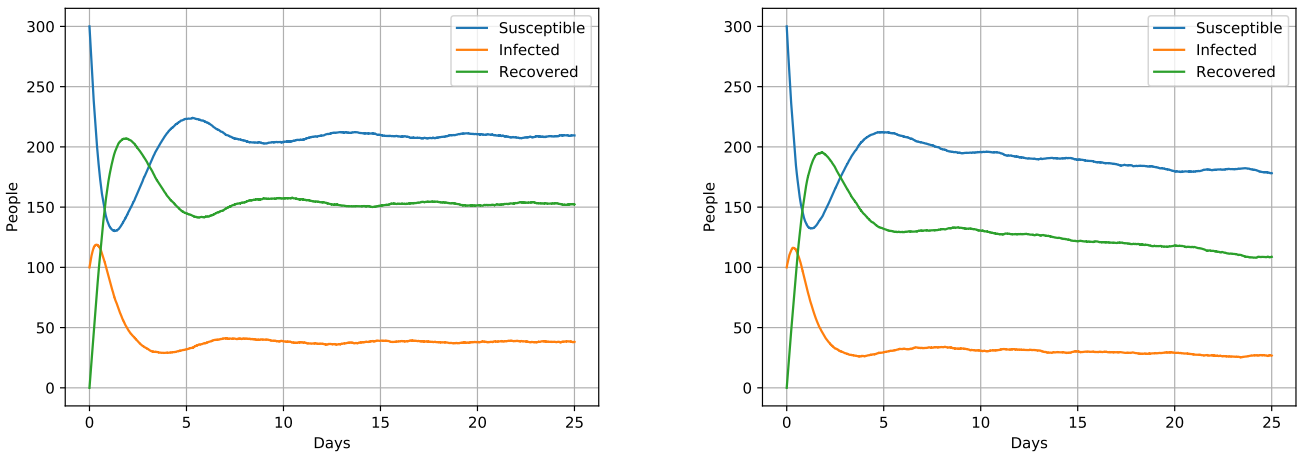

Figure 15: Plots of population B when accounting for vital dynamics, found using Monte Carlo. On the left $d_i = 0$ and on the rigth $d_i = 0.1$.

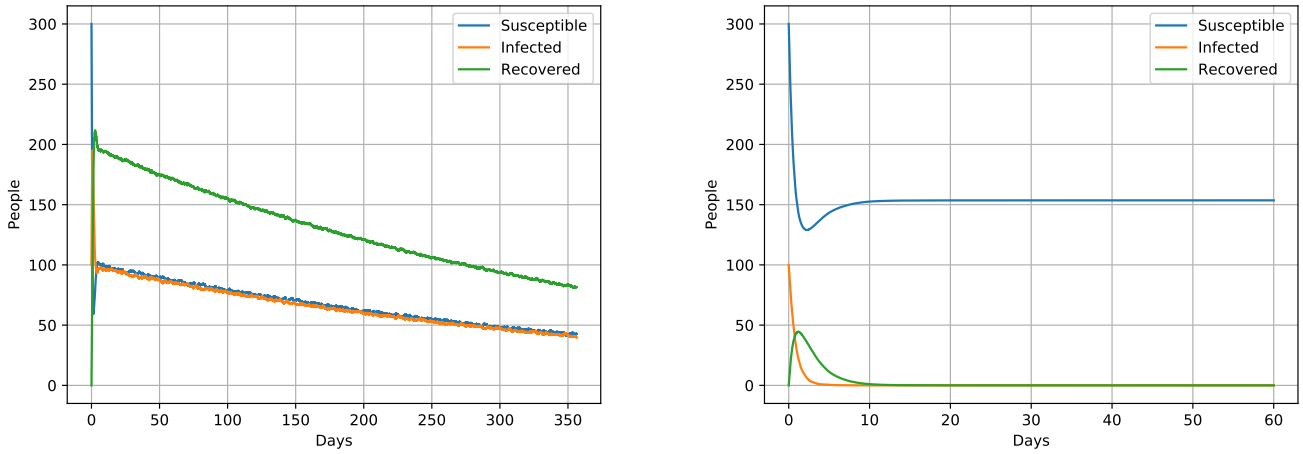

Figure 16: Plots of population A when accounting for vital dynamics, found using Monte Carlo. On the left $d_i = 0.01$ and on the rigth $d_i = 3$. The left d_i is too low to effectively kill the entire population, while other the one is too large.

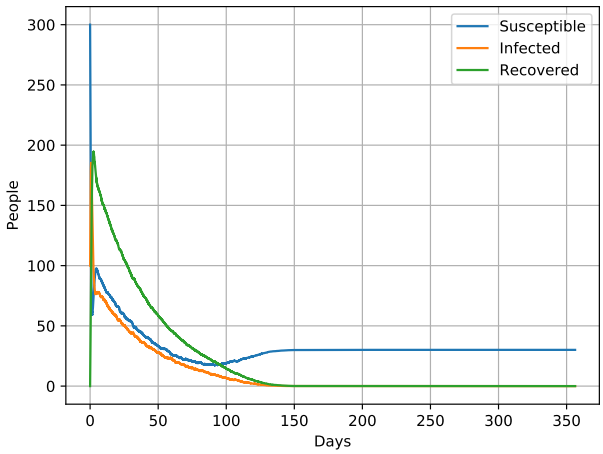

Figure 17: Plots of population A when accounting for vital dynamics, found using Monte Carlo, with $d_i = 0.1$. Here d_i is the correct size to kill most of the population.

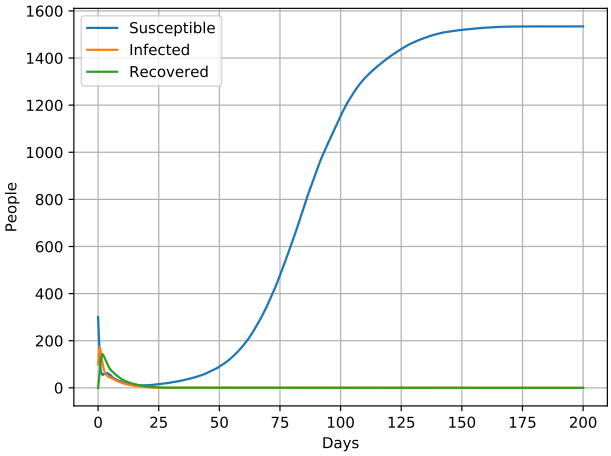

Figure 18: Plots of population B when accounting for vital dynamics, found using Monte Carlo, with $d_i = 0.1$, and d and e are increased by a factor of 1000000 .

3.c.ii Seasonal Variation

In figure 19 you can see a plot of population A and C where a varies with time.

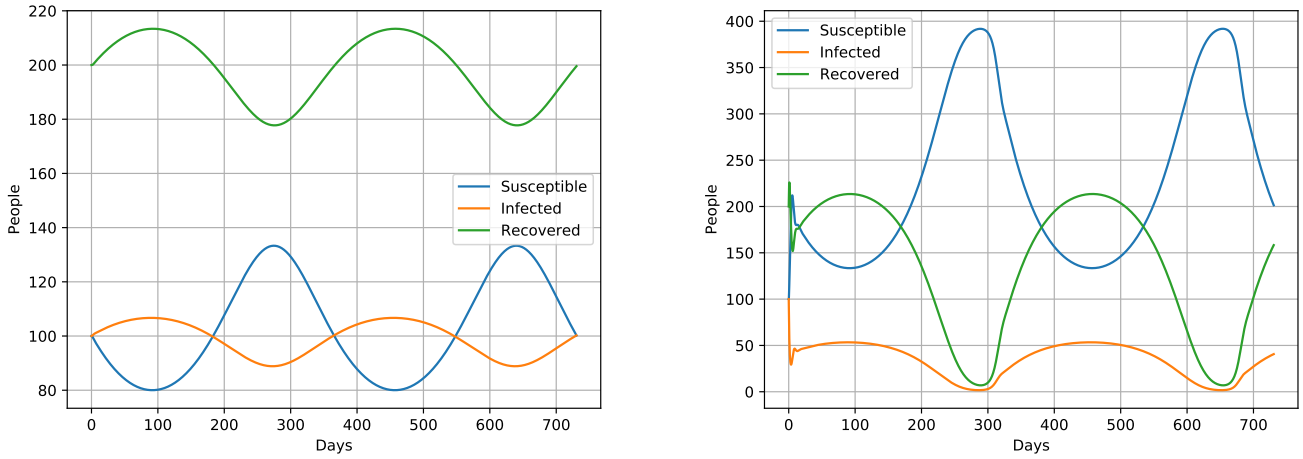

Figure 19: Plots of populations when a varies with time, found using Runge Kutta. Population A on the left with $A = 1$ and $a0 = 4$, and B on the right with $A = 2$ and $a0 = 4$.

In figure 20 you can see a plot of population A and C where a varies with time.

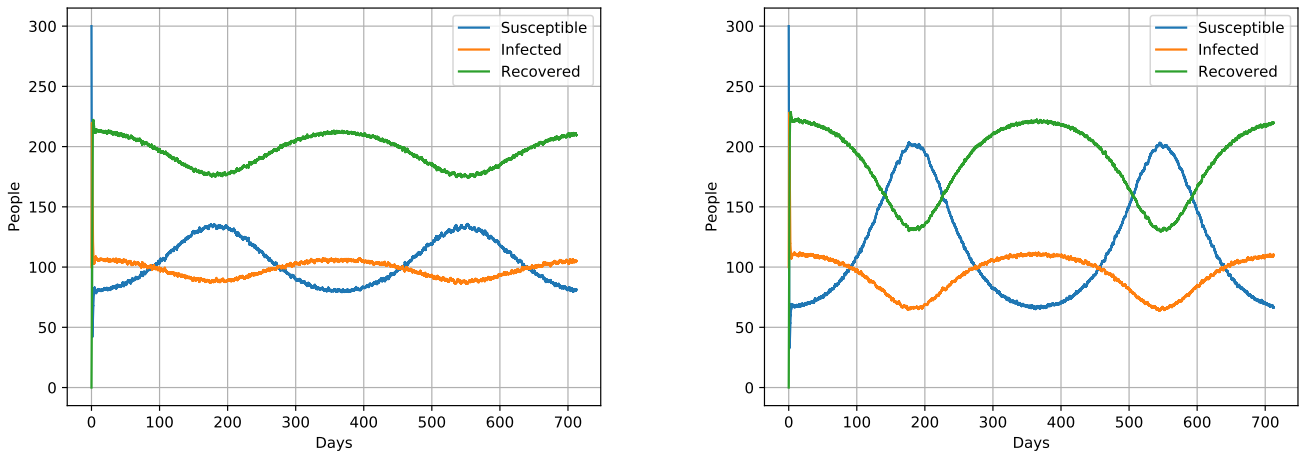

Figure 20: Plots of populations when a varies with time, found using Monte Carlo. Population A on the left with $A = 1$ and $a0 = 4$, and B on the right with $A = 2$ and $a0 = 4$.

3.c.iii Vaccination

In figure 21 you can see a plot of population B with constant vaccination. In figure 22 you can see a plot of population B with linear vaccination. In figure 23 you can see a plot of population B with a vaccination campaign.

In figure 24 you can see a plot of population B with constant vaccination. In figure 25 you can see a plot of population B with linear vaccination. In figure 26 you can see a plot of population B with a vaccination campaign.

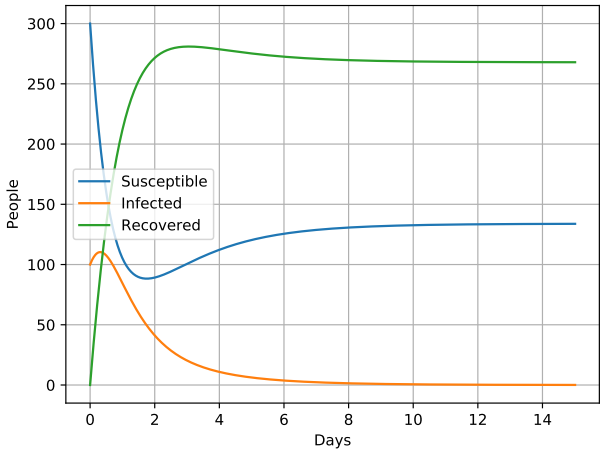

Figure 21: Plots of population B, found using Runge Kutta, with constant vaccination $f = 1$.

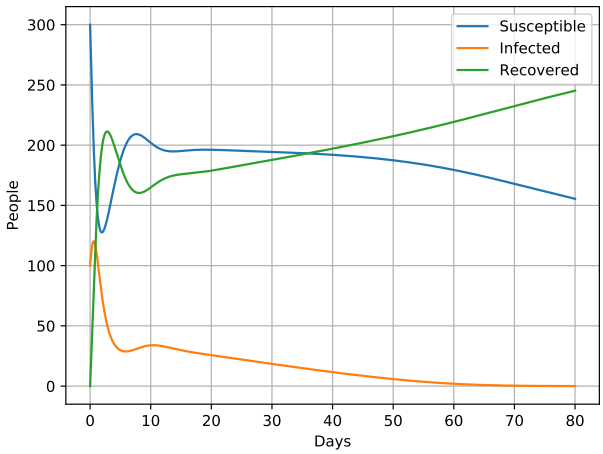

Figure 22: Plots of population B, found using Runge Kutta, with linear vaccination $f = 0.1t$.

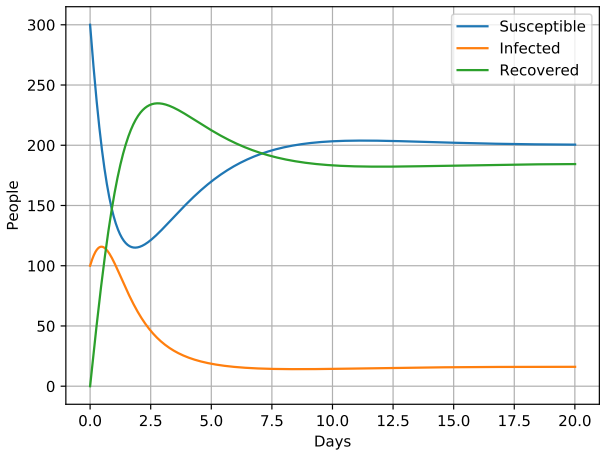

Figure 23: Plots of population B, found using Runge Kutta, with a vaccination campaign from day 2 to 9, where $f = 0.3$, while it's 0 the rest of the time.

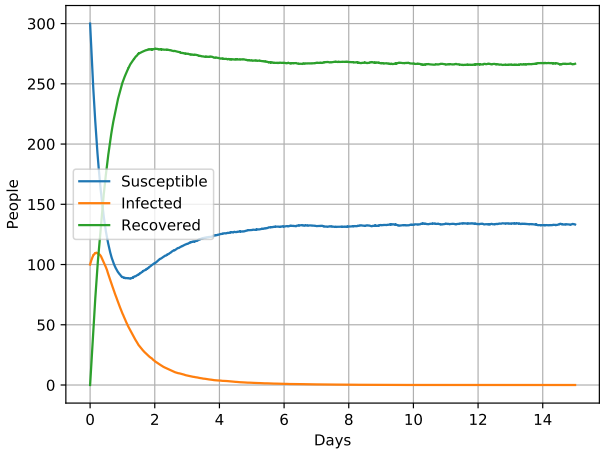

Figure 24: Plots of population B, found using Monte Carlo, with constant vaccination $f = 1$.

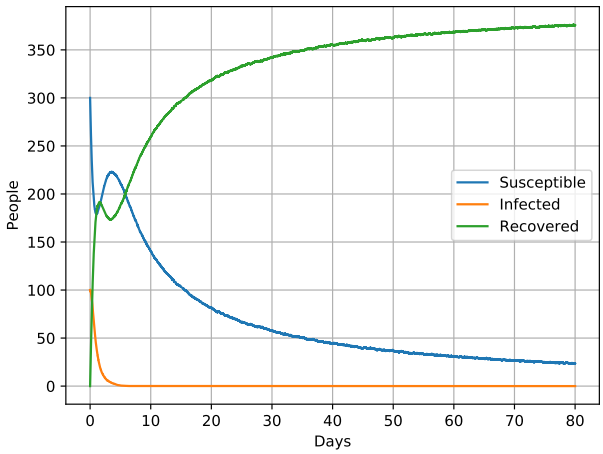

Figure 25: Plots of population B, found using Monte Carlo, with linear vaccination $f = 0.1t$.

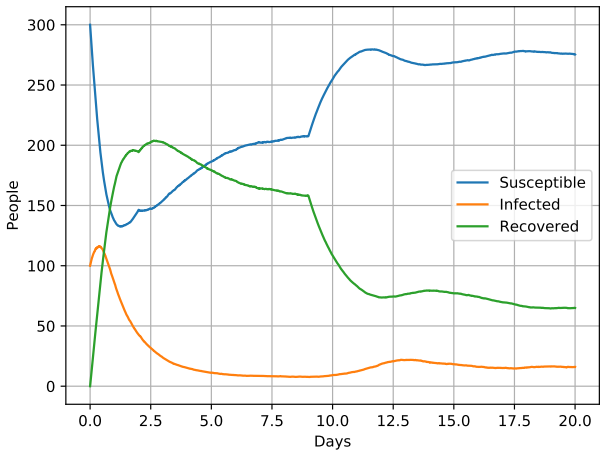

Figure 26: Plots of population B, found using Monte Carlo, with a vaccination campaign from day 2 to 9, where $f = 0.3$, while it's 0 the rest of the time.

3.d EFFICIENCY

Simulating with and without multithreading the only variables really effecting run time are number of timesteps $\frac{T}{\Delta t}$ and total simulations $sims$. Based on several runs these values are representative for the average values.

Values	Seconds without multithreading	Seconds with multithreading
$sims = 1000$, $\frac{T}{\Delta t} = \frac{5 \cdot 365.25}{0.0025}$	110.24	66.54

4 DISCUSSION

4.a Runge Kutta against Monte Carlo

4.a.i The SIRS model and Runge Kutta

From figure 1 to 4 and table 1 to 4 we can see that our Runge Kutta algorithm produces about the same values as the expected analytical values. This suggest that our model is a good approximation. We can also see that the number of people doesn't equal exact integers, but rather a decimal number. This is expected as Runge Kutta is only a numerical approximation and is indiscreet. What is not expected is that the total population increases with about 1 to 2 people. This is likely due to numerical errors, but since this constitutes less than 1% of the population it is probably safe to ignore this. A smaller Δt would probably make this error smaller.

From figure 5 and table 5 we can see that the plot for population E and D is about the same, with the amount of infected decreasing faster for population E. The numerical end values are also about the same as for population D, however the expected values are outside the acceptable values for s^*, i^* and r^*.

On figure 11 we can see that the values we chose for death and birth rate are too low to have a noticeable effect. If we ran the simulation for several years there might be an noticeable effect. From the plot on the right we can see that a value of $d_i = 0.1$ is quite significant and leads to steady decrease of the population.

From figure 12 we can see that when $d_i = 0.01$ the total population decreases quite slowly, with over half the population left after almost a year. It is unrealistic to assume that, when faced with such a decease, people wouldn't take more drastic measures to prevent the decease from spreading. We can also see that when $d_i = 3$, all the infected people die within a couple of days, yet the total population decreases slowly after that. This is likely due to the fact that the amount of infected is not exactly zero, but close to it. This causes some people to become infected, and then immediately dying. If the simulation took account for this, all the infected people would die, and then the total population would remain constant.

In figure 13 we can see a plot where the entire population dies within a year. In this case $d_i = 0.1$, and it is more reasonable to assume that people wouldn't be able to take more drastic measures here, than for $d_i = 0.01$, as people die at a much faster rate. In this plot the amount of infected, susceptible and recovered people reach zero at about the same time, meaning we don't have the same situation as for $d_i = 3$, where people become infected even though there aren't any more infected people left. So if you are a bio-terrorist you should aim to create a disease with parameter similar to this one.

In figure 14 you can see a population with an extreme population growth. This is of course high unrealistic, but is made to illustrate a population increasing over time.

In figure 19 on the left, you can see a stable system where the amount of people in each group oscillates. However the amount of oscillation is quite low, with the amount of infected people not chaining with more than about $10 - 15$ people. Most disease that oscillates like this would probably have a higher amplitude. On the right you can see a plot with a higher amplitude, however this one has the problem of reaching zero infected people. At about day 300 the disease should have died out, yet it remains due to the limitations of Runge Kutta. The other curves also appear discontinuous at the points where infected people reach about zero.

In figure 21 it appears as if the disease dies out after about 8 days. A this point the vaccination would probably stop, but other than that this seems like a realistic plot. In figure 22 the vaccination also manages to kill of the disease, but at slower rate. After this the amount of susceptible people goes to zero, while the amount of recovered goes towards 400. Again, the vaccination would probably stop here, but other than that this seems like a reasonable plot. In figure 23 the vaccination campaign was not strong enough to kill of the disease, however it did make it so that the permanent infected population would be at a lower point than if it hadn't been done. So the campaign was a partial success.

4.a.ii Monte Carlo

For the discussion of the Monte Carlo simulations we will point to the key differences between the Runge Kutta method and the Monte Carlo method.

The first noticeable difference can be found by comparing figure 12 and 16. There is a clear tendency for the Runge Kutta simulation to have a decreasing population. On the other hand the Monte Carlo simulation gives a steady population. This could be because I never truly becomes zero for the Runge Kutta simulation, and so someone will inevitably die. The Monte Carlo method, on the other hand has $I = 0$ from about day 5.

Another big difference can be found between figure 13 and 17. For some reason the population doesn't die completely with a Monte Carlo simulation. We do not know why.

Figure 14 and 18 displays a major problem with the Monte Carlo simulation. In 18 the population stabilises at about 2000, because Δt is too large.

Finally we have figures 21 through 26 , which are widely dissimilar. This is probably due to a bug in the implementation, rather than an inherent problem with one of the methods. We can however say from figure 26 that vaccination campaigns work.

4.b EFFICIENCY

We would expect the efficiency increase from multithreading to be about $\frac{1}{3.5}$ the original time. We are running this program on computers with 4 cores and no hyper threading. Subtracting some efficiency for the multithreading overhead and the slightly lower CPU clock speeds at all-core full load we get about $\frac{1}{3.5}$. Our data didn't quite match up to expectations here. We saved about 40% of our run time, which is a significant saving, but we were nowhere near the expected results.

5 CONCLUSION

The Runge Kutta method is a good approximation to use on the SIRS method, but it has certain limitations, especially when the amount of infected people is close to zero.

The Monte Carlo method also has its limitations when we encounter low levels of infected people. Whenever I is too low, there will be some simulations in which the population actually reaches zero, which means that the disease is eradicated, and any further increase in transmission rates has no effect. What this means is that we get extremely high standard deviations in the distribution of simulations that at any point gets close to zero infected. This doesn't mean that we get bad results, but rather that the average itself cant tell us as much.

Another limitation is large populations. In our implementation Δt is highly dependant on population size, and with a population the size of a country's we would get a Δt so small that the simulation would last for ages, and we would need stupendous amounts of memory.

6 APPENDICES

All the calculations were done using the programming language Julia. The programs used can be found at: https://github.com/UlrikSeip/Projects/tree/master/prosjekt5.

References

[1] Computational Physics, Lecture Notes Fall 2015, Morten Hjort-Jensen p. 419-424

[2] https://en.wikipedia.org/wiki/List_of_sovereign_states_and_dependent_territories_by_mortality_rate

[3] https://www.ssb.no/fodte/

[4] https://data.worldbank.org/indicator/SP.POP.TOTL?end=2017&start=2015&year_high_desc=true