Diseases modelling

December 2020

Contents

Abstract

We have simulated the spread of an infectious disease. To solve differential equations we used Monte Carlo and Runge-Kutta methods. We used our simple SIRS model for four different populations (A,B,C and D) with different rates of recovery. Furthermore, we improved our model and made it more realistic by adding vital dynamics, seasonal variations and vaccination. The two methods gave same results for population A and D when we simulated for vital dynamics. With vaccination improvements we concluded that vaccination should happen before the transition rate reaches its maximum. This way we can stop the disease spreading.

1 Introduction

The spread of infectious disease has been a great concern to humanity. Altough medicine, hygiene and new technology is improving from day to day, our population is still in danger of an epidemic outbreak. People have been developing strategies for managing disease threat to humans. One of great strategies is modelling of disease spread. Through models of infectious diseases we can understand how a disease spreads throughout a given population over time. To model how a certain disease establish itself within the population, we have subdivided the affected population into various groups. We have in our work developed a Monte Carlo simulation of the spread of an infection disease. To develop the necessary transition probabilities for the simulation we used a model called SIRS model. The classical SIRS model considers an isolated population of N people which are divided into three separate groups: *Susceptible* (S), *Infected* (I) and *Recovered* (R). We will construct a set of coupled differential equations that form the classical SIRS model. You will find a closely description of our SIRS model in a 'Method' section below.
We will use two solvers to simulate our differential equations. As named, one of solvers we will use is called Monte Carlo solver. We have also developed another method which can be used to predict the transmission probabilities, and gives a great numerical solution to our SIRS model. This method is called fourth-order Runge-Kutta method (RK4). This solver is called ODE-solver and it uses RK4 method.
In present work we will also expend the basic SIRS model to some more realistic models. We will do this by introducing birth and death rate, and the model is then called 'Vital dynamics'. Furthermore the model is being expended by seasonal variation and possibility of vaccination. These methods are applied on four different populations, with different recovery rates. [TA16].

2 Method

In this section we will tell some basics about the SIR model and how to develop the model, both using a numerical integration method called Runge-Kutta 4, and by using Monte Carlo simulation. Some of the methods are inspired by the project description, linked in the reference section. [Hjo20b]

2.1 About the SIRS model

We consider an isolated population of N people, and define them in three categories:

- Susceptible (S), Those without immunity to the disease. Those are capable of infecting everybody.
- Infected (I), those who are currently infected with the disease. When you become infected, you leave the susceptible categories.
- Recovered(R), those who have been infected in the past.

  A person can only move from one group to another in the cyclic order given by the name of the model $S \to I \to R \to S$ [Hjo20b]. This is also illustrated in figure 1

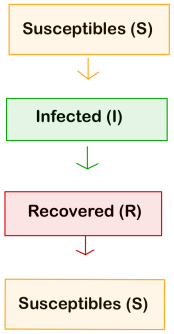


Figure 1: SIRS model

Assumptions:

1. We assume that the population mixes homogeneously and that the total population remains constant. We can express it as

$$N = S(t) + I(t) + R(t) \tag{1}$$

2. We will also assume that the dynamics of the epidemic occur during a time scale mush smaller than average person lifetime. Hence the

effect of of the birth and death of the rate of population is ignored.

3. At the initial time, there are no one at the recovery state.
4. The population is uniformly distributed.

To explain how the model is changing over time, we can present a system of coupled differential equations

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}S &= cR - \frac{aSI}{N} \\ \frac{\mathrm{d}}{\mathrm{d}t}I &= \frac{aSI}{N} - bI \\ \frac{\mathrm{d}}{\mathrm{d}t}R &= bI - cR \end{aligned} \tag{2}$$

This does not have analytical solutions like the SIR-model, but the equilibrium solutions are simple to obtain. We can reduce this three dimensional system into a two dimensional one

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}S &= c(N - S - I) - \frac{aSI}{N} \\ \frac{\mathrm{d}}{\mathrm{d}t}I &= \frac{aSI}{N} - bI \end{aligned} \tag{3}$$

We find the steady state by setting both equations in (3) equal to zero. The fraction on people in each group at equilibrium are

$$\begin{aligned} s* &= \frac{b}{a} \\ i* &= \frac{1 - \frac{b}{a}}{1 + \frac{b}{c}} \\ r* &= \frac{b}{c}\frac{1 - \frac{b}{a}}{1 + \frac{b}{c}} \end{aligned} \tag{4}$$

[Hjo20b]

2.2 Setting up the Differential Equations

In this part we will develop a Runge-Kutta fourth order code to solve the equations above (2). By using this code we look at four different populations, with different rate of recover, b. We can, by this, investigate the effect of rate of recovery, which is the one parameter that can reasonably be affected by the actions of human society, like access to health care, development of new medicines,

etc. [Hjo20b]. We will therefore fix the rate of transmission a and rate of immunity loss between populations. Each population is set to 400 people, where 100 were initially infected and 300 were initially susceptible. The initial recovery will be set to 0 since no one as has recovered or are immune when the disease is at initial time. The parameters for the four different populations are given in table 1.

Rate	**A**	**B**	**C**	**D**
a	4	4	4	4
b	1	2	3	4
c	0,5	0,5	0,5	0,5

Table 1: The rate of transmission is given by the parameter a, recovery by b, and imnuity loss c for populations A,B,C and D.

2.3 Runge-Kutta Method

To find a solution for the ODE to the SIRS model we can use the Runge-Kutta 4 method. This method is based on the Taylor expansion formula, and follow the the basic idea of finding y_{i+1} with intermediate steps. [Hjo20a]. For the Runge-Kutta 4 method we have four steps. The algorithm is given by

$$\begin{aligned} k_1 &= hf(t_i, y_i) \\ k_2 &= hf(t_i + \frac{h}{2}, y_i + \frac{k_1}{2}) \\ k_3 &= hf(t_i + \frac{h}{2}, y_i + \frac{k_2}{2}) \\ k_4 &= hf(t_i + h, y_i + k_3) \end{aligned} \tag{5}$$

Where h is the step, and f as input. This give the result

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

First we calculate k_1 with t_i,then we increase the step size with $\frac{h}{2}$ and calculate k_2, k_3 and k_4. [Hjo20a] The psudokode for the Runge-Kutta method is given in Appendix A.

2.4 Moving to Monte Carlo Simulation

We can use the idea of randomness to avoid the assumption we made in equation 2, that S, I, and R are continuous variables, by defining the probability of moving from one group to another[Hjo20b]. This probability is given by

$$
\begin{aligned}
P(S \to I) &= \frac{aSI}{N}\Delta t, \\
P(I \to R) &= bI\Delta t, \\
P(R \to S) &= cR\Delta t.
\end{aligned} \tag{6}
$$

The time step is given by

$$
\Delta t = min\Big\{\frac{4}{aN}, \frac{1}{bN}, \frac{1}{cN}\Big\}
$$

We will use this to develop a Monte Carlo algorithm which implements equation 6. The main element in this solver is that for each cycle we generate a random number. If the number is less then the probability for the move to happen, than the move is taken [Hjo20b]. This is shown in the psudeocode for Monte Carlo solver, given in Appendix B.

2.5 Expectation Values and Standard Deviation

Variance is defined as the expected value of the square difference between the outcome of the experiment. In our case will one experiment correspond to one Monte Carlo cycle (MC). We want to calculate the variance of people moving from one group to another. Variance (σ^2) is expressed as

$$
\sigma^2 = \frac{\sum_{t_i}^{t_n} (x_t - \langle x \rangle)^2}{n} \tag{7}
$$

where we are looking at a equilibrium period between time t_i to t_n. Value x_i corresponds to average number of people in each group at a time, n is number of time steps and $\langle x \rangle$ is expected number of people at a given time in equilibrium.

To find the average number of people in each group S, I and R at a equilibrium state given time t, we will simply just sum the number of people in a group and divide it by number of MC cycles.

$$
x_t = \frac{1}{MC}\sum_{i=1}^{MC} x_i \tag{8}
$$

The expected number of people in each group is a sum of people in one group for number of MC cycles, divided by number of MC cycles, then sum that for a given time, and divide the sum by number of time steps. It is expressed by equation

$$\langle x \rangle = \frac{1}{n} \sum_{i=1}^{t_i} \frac{1}{MC} \sum_{j=1}^{MC} x_j \tag{9}$$

[Siy]

2.6 Improvements, Vital Dynamics

We are now going to add vital dynamics to our system, so that the model can describe the spread of diseases which occur over longer stretches of time. The modified differential equations are given by:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}S &= cR - \frac{aSI}{N} - dS + eN \\ \frac{\mathrm{d}}{\mathrm{d}t}I &= \frac{aSI}{N} - bI - dI - d_I I \\ \frac{\mathrm{d}}{\mathrm{d}t}R &= bI - cR - dR \end{aligned} \tag{10}$$

where e is the birth rate, d is the death rate, and d_I is the death rate of infected people due to the disease [Hjo20b].

Assumption:

1. All babies born into the population are initially susceptible.

2.7 Seasonal Variations

Some diseases will have an seasonal variation, where the rate of transmission depends on the time of the year. An example of this is influenza, which occur more during the colder months. This result in a rate of transmission which oscillates. We can describe a as

$$a(t) = Acos(\omega t) + a_0 \tag{11}$$

where a_0 is the average transmission rate, A is the maximum deviation from a_0, and ω is the frequency of oscillation [Hjo20b]. We can now implement this to both the ODE solver and Monte Carlo solver. We are using the values given in the table below (table 8).

2.8 Vaccination

For diseases with vaccinations, we break the cyclic structure of the SIRS model, by allowing people to move directly from S to R. The system of differential

equations become

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}S &= cR - \frac{aSI}{N} - f \\ \frac{\mathrm{d}}{\mathrm{d}t}I &= \frac{aSI}{N} - bI \\ \frac{\mathrm{d}}{\mathrm{d}t}R &= bI - cR + f \end{aligned} \tag{12}$$

where f i the rate of vaccinations. [Hjo20b] We will vary the vaccination rate f the same way we varied the transition rate a from previous section. f is then expressed as

$$f = Fcos(\omega t) + f_0; \tag{13}$$

where F is deviation from f_0, ω is frequency, and f_0 is average vaccination rate. We will look at cases where f and a are in the same phase, and different phases.

Assumptions

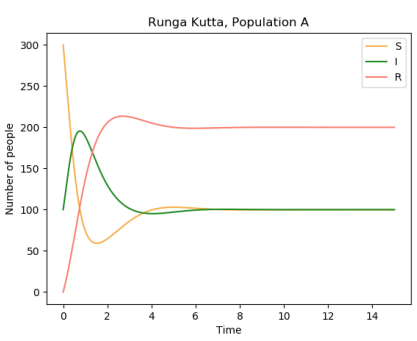
1. A susceptible individual´s choice to become vaccinated does not depend on how many other susceptibles are vaccinated [Hjo20b].
2. The rate of vaccination f can depend on the time, since this rate may oscillate during the curse of a year and/or increase as awareness and medical research increases [Hjo20b].

3 Results

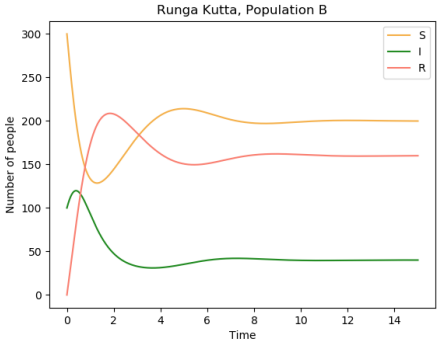
In this section we will present the most important results, and explain what we can observe. Figure 2,3, 1, 5 and 4 show the number of people in each grope at a given time, for four different populations. Each populations consist of 400 people, 100 which were initially infected, 300 of which were initially susceptible and 0 were initially recovered. The rate a, b and c for the different populations A,B, c and D are given in table 1, but to summarize the transmission rate a and the rate of immunity loss, c, is fixed, while the the recovery rate, b , increases.

3.1 SIRS Model with Runge-Kutta Method

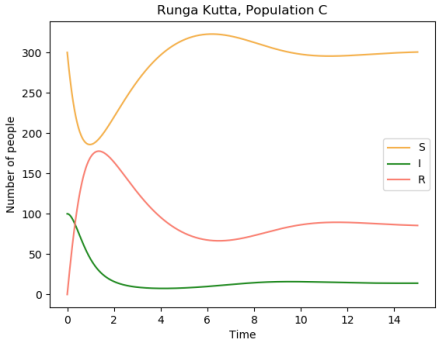
In figure 2 we see the SIRS model solved with Runge-Kutta, for each of the populations.



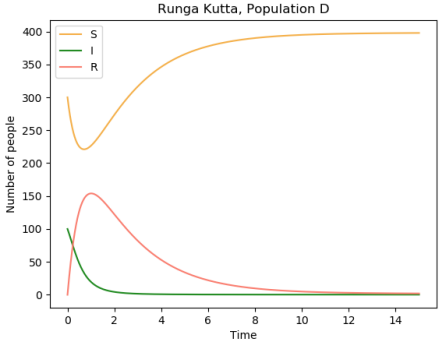
(a) Recovery rate b = 1



(b) Recovery rate b = 2



(c) Recovery rate b = 3

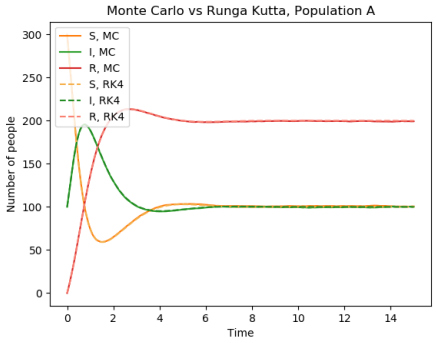


(d) Recovery rate b = 4

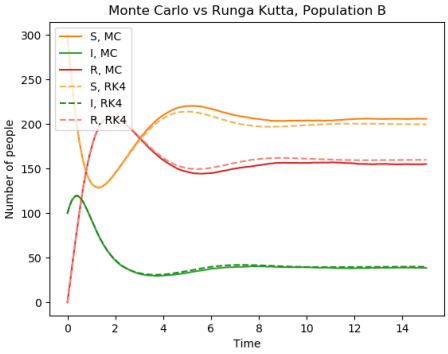
Figure 2: (a), (b), (c) and (d) show the number of people per group at a given time, for populations with different recovery rate. This is solved by using the Runge-Kutta 4 solver. The total population is constant at 400. The step size, Δt is 0.005.

3.2 Moving to Monte Carlo Simulation

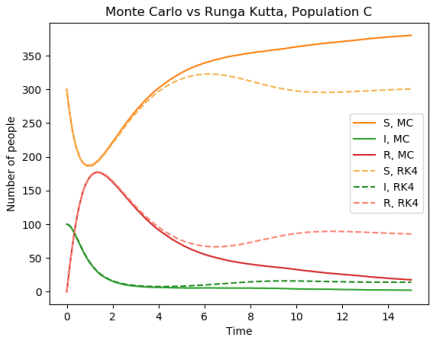
Figure 3 show the SIR model solved with both Monte Carlo- and Runge-Kutta method.



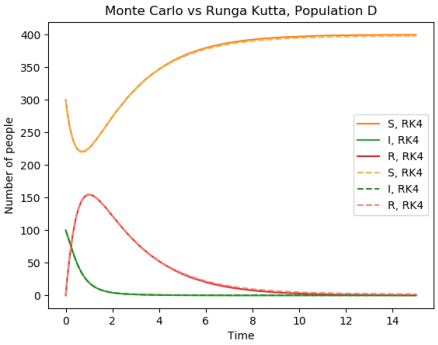
(a) Recovery rate b = 1



(b) Recovery rate b = 2



(c) Recovery rate b = 3



(d) Recovery rate b = 4

Figure 3: (a), (b), (c) and (d) show the number of people per group at a given time, for populations with different recovery rate. The fully drawn line is solved with the Monte Carlo solver, and compered with the methods for Runge-Kutta given by the dotted lines. The time step, Δt, is 0.005.

Fraction of People in Equilibrium.

The fraction of people in each group at equilibrium given by equation 4, and calculated with the parameters in table 1, are shown in table 2.

	A	**B**	**C**	**D**
s*	0,25	0,5	0,75	1
i*	0.25	0,1	0.0357	0
r*	0.5	0,4	0,2142	0

Table 2: Caption

The fraction of people in equilibrium, calculated from the Monte Carlo Simulation are shown in table 3. We have found the fractions by reading from figure 3 how many people there are in each group at equilibrium and divided it by the total number of people in the system.

	A	**B**	**C**	**D**
s*	0,25	0,51875	0,992	1
i*	0.25	0,09575	0.00005	0
r*	0.5	0,3835	0,00825	0

Table 3: Caption

Expectation Values and Standard Deviation

Standard deviation for Monte Carlo is given in table 4

Group	**A**	**B**	**C**	**D**
S	$1{,}5\times10^{-4}$	$4{,}1\times10^{-4}$	$1{,}2\times10^{-3}$	$7{,}4\times10^{-4}$
I	$9{,}2\times10^{-5}$	$1{,}1\times10^{-4}$	$1{,}1\times10^{-4}$	$4{,}2\times10^{-5}$
R	$1{,}8\times10^{-4}$	$4{,}2\times10^{-4}$	$1{,}1\times10^{-4}$	$7{,}1\times10^{-4}$

Table 4: Standard deviation for Monte Carlo simulation at equilibrium for the four populations.

The standard deviation for the Runge-Kutta methods is given in 5.

Group	**A**	**B**	**C**	**D**
S	$3{,}1\times10^{-3}$	$6{,}2\times10^{-3}$	$1{,}2\times10^{-3}$	$9{,}9\times10^{-4}$
I	$5{,}3\times10^{-4}$	$8{,}7\times10^{-4}$	$1{,}2\times10^{-5}$	$6{,}3\times10^{-7}$
R	$1{,}6\times10^{-3}$	$2{,}2\times10^{-3}$	$9{,}6\times10^{-4}$	$7{,}1\times10^{-4}$

Table 5: Standard deviation for Runge-Kutta method at equilibrium for the four populations.

Expectation value at equilibrium is given in table 6.

Group	**A**	**B**	**C**	**D**
S	100,1	205,5	363.8	385,1
I	99,4	38,2	3,5	0,5
R	199.8	156,3	34,3	14,4

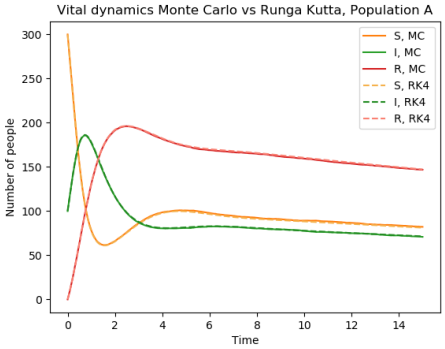
Table 6: Expectation values for the Monte Carlo simuklation at equilibrium for the four populations.

3.3 Improvements, Vital Dynamics

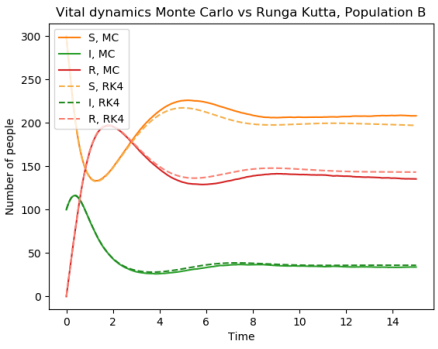
In this section we have added vital dynamics. The results are given in figure 4, for both the Runge-Kutta solver and Monte Carlo solver. The parameter used is given in table. 7 below.

rate	**Population A,B,C, D**
Birth rate, e	0.009
Death rate, d	0,0002
Infected rate, d_l	0.1

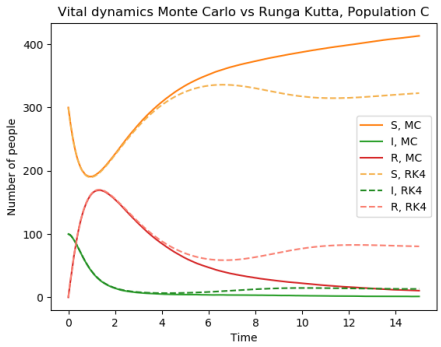
Table 7: The birth rate is based on that there are aproximently 50 000 people born in a year with a population on 5,5 million people. The death of infected people rate indicates that if a 100 people get the disease, 10 people will die. The death rate is from fhi [Fol20].



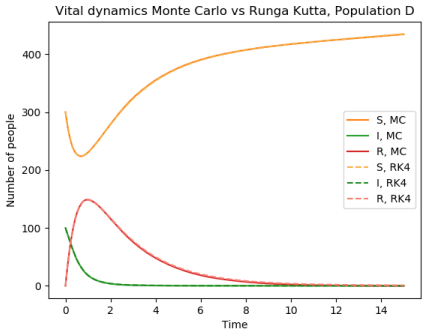
(a) Recovery rate b = 1



(b) Recovery rate b = 2



(c) Recovery rate b = 3



(d) Recovery rate b = 4

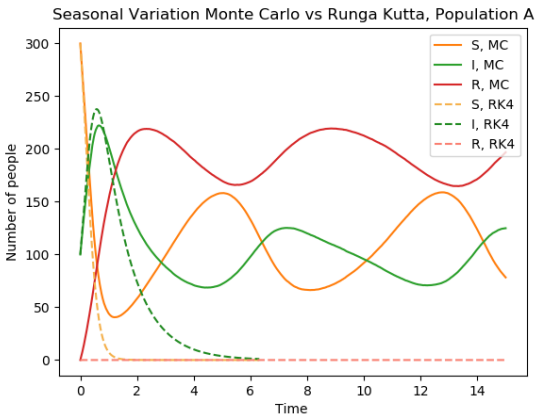
Figure 4: (a), (b), (c) and (d) show the number of people per group at a given time, for populations with different recovery rate, when added vital dynamics. The fully drawn line is solved by using Monte Carlo solver, while the dotted line is solved by Runge-Kutta 4. The time step, Δt is 0.005.

3.4 Seasonal Variations

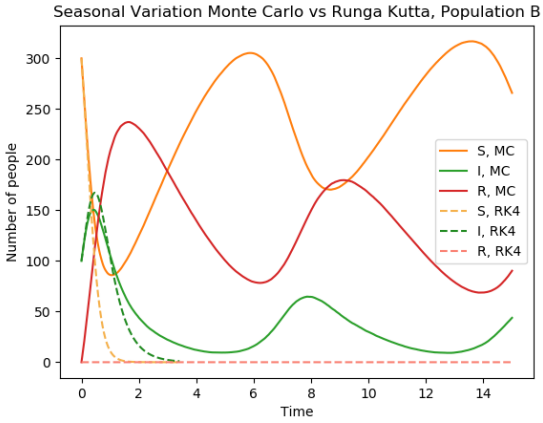
In figure 5 we see the SIRS model for the different population when taking seasonal variation into account. The different parameters we are using is given in tabel 8 below.

Parameter	**1**	**2**
Average transmission rate, a_0	4	4
Maximum deviation, A	1.5	1.5
Frequency of oscillation, ω	0.8	0,4

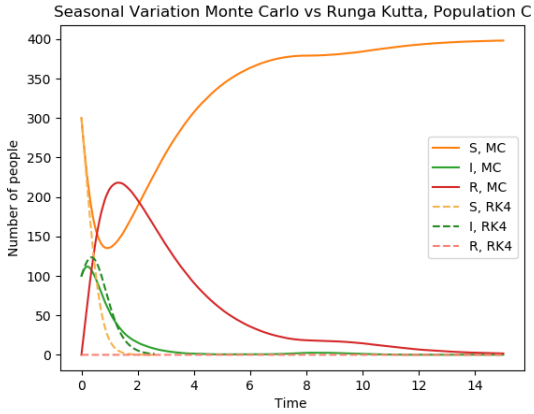
Table 8: The values implemented for the parameters a_0, A and ω.



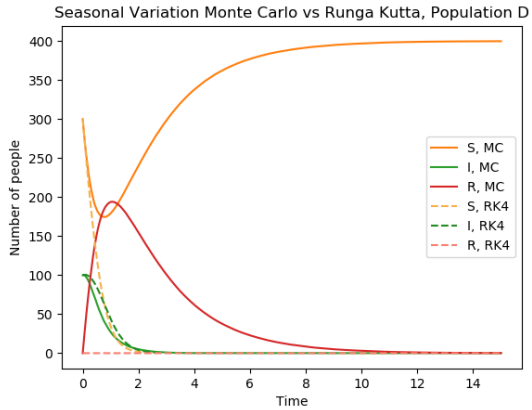
(a) Recovery rate b = 1



(b) Recovery rate b = 2



(c) Recovery rate b = 3



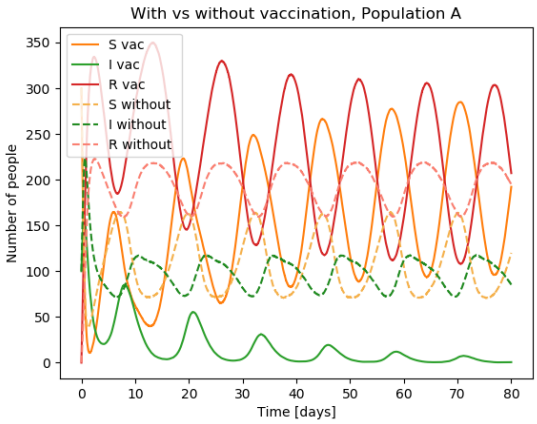
(d) Recovery rate b = 4

Figure 5: (a), (b), (c) and (d) show the number of people per group at a given time, for populations with different recovery rate, when taking the seasonal variation into account. The fully drawn line is solved by using Monte Carlo solver, while the dotted line is solved by Runge-Kutta 4. The time step, Δt is 0.005.

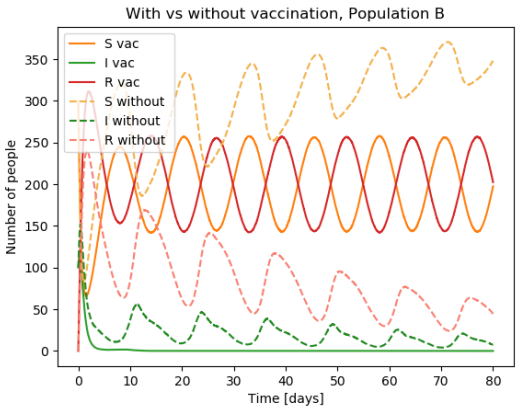
3.5 Vaccination

We have plotted SIRS model with vaccination improvements using Monte Carlo method for populations A and B. We want to compare the effect of vaccination. Figure 6 shows the difference between before and after vaccination for population A and B. In Figure 7 there is a phase difference between f and a, while in figure Figure 2.5 f and a are in the same phase. These plots are obtained with 700

Monte Carlo cycles, $\omega = 0.5$, A=1, a0=2. As vaccination rate we used f=80.

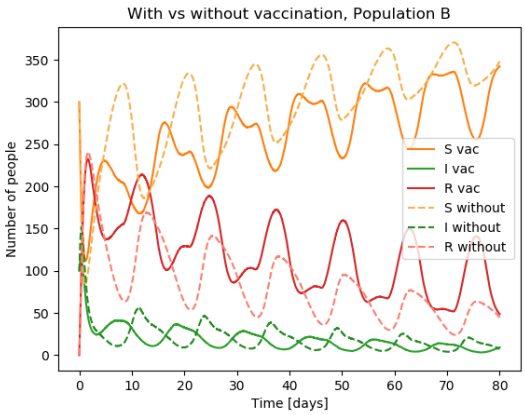


(a) Recovery rate b = 1

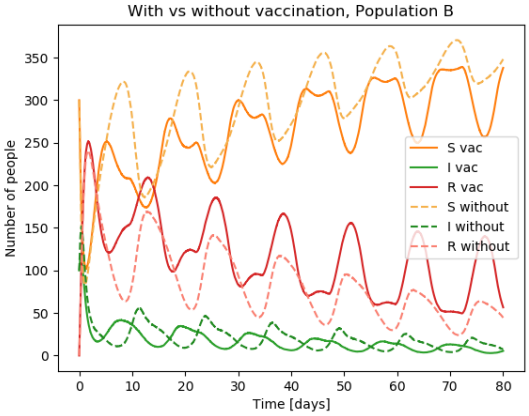


(b) Recovery rate b = 2

Figure 6: f is constant

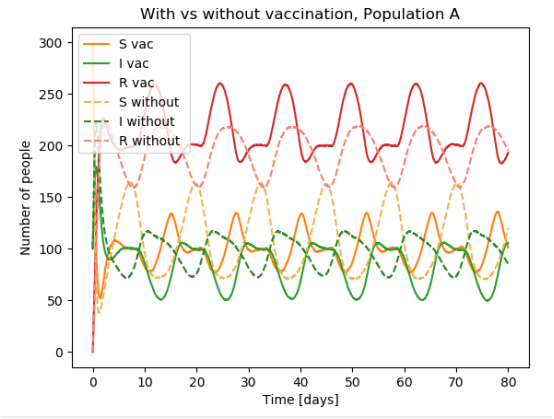


(a) Recovery rate b = 2

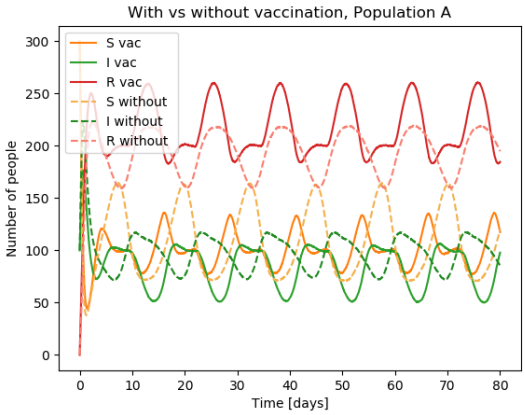


(b) Recovery rate b = 2

Figure 7: There is a phase difference between f and a



(a) Recovery rate b = 1



(b) Recovery rate b = 1

Figure 8: f and a are in the same phase

4 Discussion

In this section we are going to discuss the results presented above, and the methods used to obtain them.

4.1 SIRS Model with Runge-Kutta Method

In figure 2a we see that for low recovery rates the number of infected and susceptible (line I and S) will stabilize at the same number, while the number of recovered (line R) people stabilises at a higher number of people, graph 2a. When the recovery rate gets higher we see that a higher percentage of the people become susceptible again, while the recovery- and infected rate goes towards zero. When this reaches zeros the disease has dies out. From our graphs, we see that all the groups reached an equilibrium.

This is system with a constant population size, in which no one dies or is born, which implies that this is a system that spans over just a short time period.

4.2 Moving to Monte Carlo Simulation

In figure 3, we have presented the results from simulating the SIRS-model using Monte Carlo algorithm, and compared it with the results we obtained from Runge-Kutta. In figure 3a, we see that the Monte Carlo- and Runge-Kutta algorithm give the same results, where Runge-Kutta is denoted as a dotted line. This corresponds with the values for population A in table 2 and 3. In population B, however, we see a small difference in equilibrium states for the two algorithms, while the pre-equilibrium states follow the pattern. We observe that this applies for every population.

We ran our program with 1000 Monte Carlo cycles and $\Delta t = 0.05$ and noticed that it takes much longer time to run Monte Carlo than Runge-Kutta, but both methods didn't use that much time.
From table 4 and 5 we see that standard deviation is smaller for Monte Carlo method. This means that we expect better results for Monte Carlo method.

4.3 Improvements, Vital Dynamics

For these simulations, figure 4, the main difference from before is that we have a dynamic population. This means that we simulate births and deaths of the population, and deaths caused by the disease. This allows us to simulate a larger time span. We are observing a system in which 1/10 of the infected die from the disease. It is worth keeping in mind that since we now allow deaths, we will generally have lower recovery rates for every population. From what we can observe, the two simulations, with and without vital dynamics, gives similar results and stabilizes around the same time. However, a possible reason why they are similar is because we may have simulated a too short time period for the effects of vital dynamics to take effect. The main difference between the two

simulations are the recovery rates, which are generally lower in the simulations with vital dynamics.

4.4 Seasonal Variations

Number of infected people will decrease in colder months of the year, and increase in warmer months. The reason for this is that the disease will be weaker at higher temperatures, and stronger at lower temperatures. It will also have something to say that people are more closely connected in the winter, since they prefer to be indoors rather than outdoors. People are more likely to be infected indoor. We can see this improvements with seasonal variation rate are illustrated in figure 5. We can see that system reaches equilibrium point at same time as the simple SIRS model shown in figure 3. We notice that Monte Carlo simulation gives curved function, just like a harmonic function, which is a sign of seasonal variation. We expected that Runge-Kutta also would have curves, but they did not appear in the figure.

4.5 Vaccination

In the figure 6 we can see the effect of vaccination improvements. With this improvement, we see that the number of infected decreases and the number of immune people increases. This result is as we expected. We have plotted for three different values of vaccination rate. In figure 7 we shifted the transition rate so that a lot of people are being vaccinated in the begining. Because of this we can see that number of susceptibles increases while number of recovered decreases. This means that number of immune is increasing. On the other hand, figure shows that when the transition rate is in the same phase as vaccination rate, number of immune increases but not as fast as in 7. We can conclude that it is very good to vaccinate before the transition rate is at its maximum.

5 Conclusion

In our work we first looked at simple SIRS model and simulated it with two methods, Monte Carlo and Runge-Kutta. We compared the four populations, with different recovery rate, and simulated that population D ends with most immune people, and fewest dead. We can see from figure 3 (a) and (d) that both methods gave same result. figure 2 shows that With b=4 for population D, there are no infected left after some time. Adding birth and death rate will improve our system, number of recovered people decreases because some of infected will die. This improvement will allow us to simulate for a longer period of time, and we will have a more realistic system.

With seasonal varying improvements we can clearly see that for Monte Carlo simulation the system varies as seasons vary, but didn't see that Runge-Kutta reacts the same way. Monte Carlo reaches the equilibrium state the same time as the simple SIRS model does. Finally we improved our model with vaccination. We saw that number of infected decreases and number of immune increases with vaccination. A great way to prevent the infection spread is to vaccinate before transmition rate is at its maximum.

For further studies we might want to simulate for larger number of people. If so, we could make some changes to our program to make it more efficient. We also made some assumptions to our model and simplified it. We assumed that the dynamics of the epidemic occur during a time scale much smaller than the average person's lifetime. To conclude we can through our model of infectious diseases we can understand how a disease spreads throughout a given population over time, and learn how to prevent it.

6 References

References

[TA16] A. S. Talawar and U. R. Aundhakar. "Parameter Estimation of SIREpidemic Model Using MCMC Methods, pp". In: (2016), pp. 81299–1306. DOI: https://www.researchgate.net/profile/Appanna_Talawar/publication/303124617_Parameter_estimation_of_SIR_epidemic_model_using_MCMC_methods/links/5ef05eeaa6fdcc73be9437b3/Parameter-estimation-of-SIR-epidemic-model-MCMC-methods.pdf.

[Siy] Siyavula. *Variance and standard deviation.* URL: https://intl.siyavula.com/read/maths/grade-11/statistics/11-statistics-04.

[Fol20] Folkehelseinstituttet. *Lavere dødelighet i Norge for noensykdommer under pandemien.* 22.09, 2020. URL: https://www.fhi.no/nyheter/2020/lavere-dodelighet-i-norge-for-noen-sykdommer-under-pandemien/.

[Hjo20a] M. Hjorth-Jensen. *Computational Physics Lectures:Ordinary differential equations.* 24.09 2020. URL: http://compphysics.github.io/ComputationalPhysics/%20doc/pub/ode/html/._ode-bs000.html.

[Hjo20b] M. Hjorth-Jensen. *Prosject 5, deadline Desember 16,2020.* Fall 2020. URL: http://compphysics.github.io/ComputationalPhysics/doc/Projects/2020/Project5/DiseaseModeling/pdf/DiseaseModeling.pdf.

7 Appendix A

Pseudocode for Runge-Kutta 4

```
Runga_Kutta_solver(){
//Making vector containing SIR- values
vec SIR = [S,I,R]
 //Creating vector K1,k2,k3 and k4  for holding temporary SIR-values
vec k1,k2,k3,k4

    for (int i < number of steps){
        k1 = calcute (SIR) *dt
        k2 = calculate(SIR + k1/2) *dt
        k3 = calculate(SIR +k2/2) *dt
        k4 = calcukate(SIR + k3) *dt
        SIR += (k1 +2*k2 + 2*k3 + k4)/6

        //Write values to file
    }
}
Calculate(vec SIR){
//creating temporary array containg the calculated SIR-vales
t = [0,0,0]
//We calculate the SIR-values using the diffrental equations
t[0] = c*SIR(2) - a*SIR(0)*SIR(1)/N
t[1]=  a* SIR(0)*SIR(1)/N - b*SIR(1)
t[2] = b*SIR(1) - c*R(2)

return t
}
```

8 Appendix B

Pseudocode for Monte Carlo Solver

```
Monte_Carlo_Solver(){

// Defining dt and n
// Making vector for S, I and R filled with zero.


for i < number of simulations {
    for j < number of steps{

        //Defining the probabilties
        S ->I = s* SIR(0)*SIR(1)/N *dt
        I -> R = b* SIR(1) *dt
        R -> S = c* SIR(2) *dt

        // Count transition probabilities and transfer from/to relevant group
        Number of S->I =0
        Number of I->R =0
        Number of R->S =0

        if random number < S ->I {
            ++ Number of S->I
        }
        if random number < I ->R {
            ++ Number of I->R
        }
        if random number < R ->S {
            ++ Number of R->S
        }


        SIR(0) += Number of R->S - Number of S->I
        SIR(1) += Number of S->I - Number of I->R
        SIR(2) += Number of I->R - Number of R->S

        N = SIR(0) + SIR(1) + SIR(2). // Total number = S + I +R

        S(j) += SIR(0);
        I(j) += SIR(1);
        R(j) += SIR(2);
```

```
    }
}


S /= number of simulations
I/= number of simulations
R /= number of simulations

for i < number of steps{
    //Write to file
}

}
```
