FYS3150/4150 - Project 5
Disease Modeling with SIRS-model

Aram Salihi[1], Adam Niewelgowski[1]
[1]Department of Physics, University of Oslo, N-0316 Oslo, Norway

December 17, 2018

Abstract

ABSTRAKTEN HER

Contents

1 Introduction

Having the knowledge how deadly infectious spreads among a closed or open group, can be crucial in order to stop a deadly disease before its to late. In order to have this knowledge we must simply build a mathematical model which can describe this notion. In early seventeenth hundred such models were derived and used to predict of how disease behaved when introducing a vaccine. This field have expanded into mathematical epidemiology.

The most famous and commonly used is the SIR (**S**usceptible, **I**nfected, **R**ecovered) model. The philosophy behind this model is to introduce three different compartment groups into a system, where one of the group carries a infectious disease. Assuming that each individual from each group has the possibility to interact with each other. This will lead to people become ill from the disease, but also recover from the disease depending how infectious and deadly the disease is. In order to describe this notion, we must develop a set of coupled differential equations, which explains this type of behaviour in a closed or open system. From this we can develop a understanding how fast a disease can spread, but also how fast people heal and recover from a infectious disease.

Many models in mathematical epidemiology are derived from this basic idea used in SIR model. Models such as SIRS, MSIR, SEIR, and more. However this model is not only limited to this field of studies. The SIR model can also be expanded and be used in the study of traffic congestion(see e.g (REFERANSE til paperen)). The main idea is to have three different groups, where objects are moved around between groups based on system's conditions.

In this numerical study we will be studying the model SIRS which is explained in depth in the next section. We wish wish to study different events created by this model.

- We will first introduce a non deadly infectious disease, and simply assume no one dies from it. We will also assume that no one from each compartment groups dies of natural cause and gives birth. Thus simulating a closed system where the total population number is fixed.
- We will then continue our studies to a open system, where we allow people to die from natural cause and the disease itself, but also introduce a birth rate.
- We will also study how some seasonal diseases impact the model by introducing a new time depdendent osciallating transmission rate. For diseases such as influenza, the rate of transmission depends largely on the time of year we are in.
- We also introduce a vaccine for an arbitray disease.

In order to solve these coupled time dependent system we have chosen two types of approach. First we will directly solve the coupled differential equation by using fourth order Runge-Kutta method. We will then compared it with Markov Chain Monte Carlo method, where the transition from one state/group to another state/group be based on probability and random numbers. Keep in mind we will also compare the method against each other.

2 Theory

In this section we will present the theoretical framework and different types of model used in this numerical study. Keep in mind that the algorithm Runge-Kutta of fourth and probability transition in Monte Carlo is also derived in this section. For implementation of Monte Carlo and Runge kutta please see (ref her)

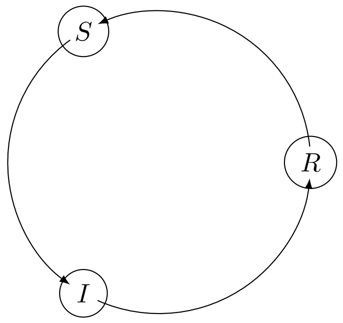
2.1 SIR-model

Consider a group of "susceptible" with population S, which has the possibility to be infected by a deadly or non-deadly disease by someone which carries this infection. The person who carries this infection is in the "infected" group with population I. We will now assume that there is some possibility (depends on the disease) that one or more from group I heals and recoveres from the disease, and becomes immune. Thus, the persons who heals moves to the group "recovered" with population R. These three variables represent the number of people in each compartment at particular time t. Keep in mind that these three group are time dependent even though the number of people in each compartment are constant. Thus the total population in this system follows

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

2.2 SIRS-model

We will now consider a bit more general model compared with previous one presented in (**??**) called SIRS. Assuming no babies born into these three group and the possibility of dying being zero. We will now consider a group of recovered people from a disease $\mathcal{D}$. We will now assume that these people will have the ability to loose immunity from this disease and become susceptible. Meaning that they can be infected by someone with disease $\mathcal{D}$. We will now have circular system. The idea behind is demonstrated in the figure down below



In other words we have consider a case with a closed system. The change in the population for this system is described by following coupled differential equations

$$\frac{\mathrm{d}S}{\mathrm{d}t} = cR - \frac{aSI}{N} \tag{2}$$

$$\frac{\mathrm{d}I}{\mathrm{d}t} = \frac{aSI}{N} - bI \tag{3}$$

$$\frac{\mathrm{d}R}{\mathrm{d}t} = bI - cR \tag{4}$$

Where a describes the transmission rate of the disease, b describes the rate of recovary, and c describes the loss of immunity. The change described by these equations will always keep the total population $N(t)$ constant. This can be easily seen when adding S', I' and R' together.

Vital dynamics (open system) We will now consider a open system. Meaning we will have a even more realistic scenario compared with two previous models. We will now introduce a death rate which can occour in group S, I and R. The cause of death is not by the disease itself, but describing other reasons such as natural cause of death such as heart attack. We will define this parameter rate as d. Further we will introduce a death rate of people dying from the disease itself, and call this rate for d_i. To make it even more realistic we will introduce a birth rate parameter e. This parameter explains the rate of babies added to the system. We will now add a few terms to the differential equations introduced in (4)

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

We will further assume that all babies added to the system are susceptible. If we take the sum of the these equation, we will notice that

$$\frac{\mathrm{d}S}{\mathrm{d}t} + \frac{\mathrm{d}I}{\mathrm{d}t} + \frac{\mathrm{d}R}{\mathrm{d}t} = eN - dR - (d_i - d)I \tag{5}$$

From this we can see that the population number N is never constant. It will either decrease or increase depending how the factors are behaving with respect to each other. Notice that in order to keep a increase in population with this model, eN must dominate

$$eN >> dR + (d_i + d)I \tag{6}$$

Considering the other case, simulating an apocalypse with deadly disease/infection with a decreasing population, we must have

$$eN << dR + (d_i + d)I \tag{7}$$

Thus $dR + (d_i + d)I$ must dominate in order to bring the population down.

Seasonal variation As introduced in the introdcution, we also wish to study how the population is affected by oscillating diseases such as influenza. We will slightly change the transmition rate a to be time dependend

$$a(t) = A\cos(\omega t) + a_0 \tag{8}$$

where a_0 is the average transmission rate, A is the maximum deviation from the average and ω is frequency of oscillation.

Vaccination Here we introduce a new constant, f, which will break our circular model, and allow transition directly from **S** to **R**. The set of differential equation will then look like

$$\begin{aligned}\frac{dS}{dt} &= cR - \frac{aSI}{N} - f\\ \frac{dI}{dt} &= \frac{aSI}{N} - bI\\ \frac{dR}{dt} &= bI - cR + f\end{aligned}$$

Keep in mind we have removed the parameter explaining the death and the birth rate the system. Meaning we wish to investigate the maximum potential of a vaccine. In this numerical study we wish to implement a time dependent vaccination rate $f(t)$, to simulate the improvement of the vaccination as more people get vaccinated as the time goes. The function which satisfies this is

$$f(t) = \frac{f_0}{1 + e^{-t}} \tag{9}$$

Where f_0 is the initial rate of how many people gets treated with this type of vaccine.

2.3 Numerical algorithm

2.4 Runge-kutta 2

2.4.1 Runge-Kutta 4

Runge-Kutta of fourth order is widely used numerical algorithm for solving ordinary differential equations ,due to its precision which makes it a superior and relatively fast method. As other numerical ODE algorithms it is also based on Taylor expansion theorem. The idea behind this method is that it provides intermediate steps to calculate y_{i+1}. Consider a general first order ordinary differential on the form:

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y) \tag{10}$$

We wish to solve this ODE, i.e, we want to integrate both sides with respect to t. Keep in mind that y and t can represent all kind of phenomenas.

$$y(t) = \int f(t, y) dt \tag{11}$$

The idea now is to discretize our solution interval $y \to y_i$ and our interval $t \to t_i$. We want to approximate the next solution y_{i+1} by using the previous solution y_i with a step of

$$\int_{t_i}^{t_{i+1}} f(t, y)\,\mathrm{d}t \tag{12}$$

Thus the equation we want to end up with

$$y_{i+1} = y_i + \int_{t_i}^{t_{i+1}} f(t, y)\,\mathrm{d}t \tag{13}$$

We now wish to taylor expand our integration interval around a midpoint, thus defining

$$t_{i+1/2} = \frac{t_i + t_{i+1}}{2} = t_i + h/2 \tag{14}$$

We first use Taylor expansion to approximate $f(t, y)$ around the midpoint of the integration interval, this is at $t_i + h/2$, where h is the step-size. Then we use the midpoint formula, we get

$$\int_{t_i}^{t_{i+1}} f(t, y)dt \approx hf(t_{i+1/2}, y_{i+1/2}) + O(h^3)$$

where we use subscript $y(t_i + h/2) = y_{i+1/2}$ and equivalent for time, $t_i + h/2 = t_{i+1/2}$.

Substitution into (**??**) results in

$$y_{i+1} = y_i + hf(t_{i+1/2}, y_{i+1/2}) + O(h^3)$$

We have to find the value of $y_{i+1/2}$ now. By employing Euler's method we get the approximation

$$y_{i+1/2} \approx y_i + \frac{h}{2}\,\mathrm{d}y\,t = y(t_i) + \frac{h}{2}f(t_i, y_i)$$

We have now derived the second-order Runge-Kutta method. Coefficients are given by

$$k_1 = hf(t_i, y_i)$$
$$k_2 = hf(t_{i+1/2}, y_{i+1/2})$$

and the update function is

$$y_{i+1} \approx y_i + k_2 + O(h^3)$$

For the fourth-order Runge-Kutta method, we simply use the same idea, but instead of approximating the integral with midpoint rulel, we now employ the Simpson's rule (KANSKJE EN REFERANSE TIL DETTE, finne i boka til Morten) and get

$$\int_{t_i}^{t_{i+1}} f(t, y)dt \approx \frac{h}{6}\left[f(t_i, y_i) + 4f(t_{i+1/2}, y_{i+1/2}) + f(t_{i+1}, y_{i+1})\right] + O(h^5)$$

Since we do not know the values of the midpoints evaluation, we simply split it in two steps, so the update function becomes

$$y_{i+1} \approx y_i + \frac{h}{6}\left[f(t_i, y_i) + 2f(t_{i+1/2}, y_{i+1/2}) + 2f(t_{i+1/2}, y_{i+1/2}) + f(t_{i+1}, y_{i+1})\right]$$

Summarizing the method:

1. Find the first coefficient

$$k_1 = hf(t_i, y_i)$$

2. Use Euler's method to find $y_{i+1/2}$, so the next coefficient is

$$k_2 = hf(t_i + h/2, y_i + k_1/2)$$

3. Improve the slope calculation by finding

$$k_3 = hf(t_i + h/2, y_i + k_2/2)$$

4. Last coefficient is computed as follows

$$k_4 = hf(t_i + h, y_i + k_3)$$

5. Finally, the update function is given by

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

2.5 Monte Carlo simulation

The philosophy behind Monte Carlo simulation is to draw a number from a probabilty transition or density, and compared it with random uniformed number in the interval $[0, 1]$. In our case we assume that S, I and R are continous, but in reality S, I and R are discrete. Using the idea of randomness, we can use the philosophy behind Monte Carlo simulation to construct this discrete movement between groups. In order to do this we must a have set of probability transitions explaining the move from $S \to I$, $I \to R$ and $R \to S$. The beauty of this is that it is easily expanded, and probability transition explaining other moves can easily be implemented. The implementation of this is explained in (fullref her) (ref her med tall). In order to make a move from a general group $j \to q$, the probabilty is then, a small timestep Δt multiplied with the number of people moving from $j \to q$. Recalling the factors used in the differential in subsection 2.2 SIRS-model, section 2.2 Vital dynamics (open system) and Figure 10 , we can define probability transition to a another compartment group as following:

- Lets assume that a susceptible person from group S interacts with a infected person from group I. Depending how large the transmission rate parameter a of some disease $\mathcal{D}$ is. The number of people moving from S to I at small time step Δt is

$$S \to I = \frac{aSI}{N} \tag{15}$$

The probabilty transition is then is defined as

$$P(S \to I) = \frac{aSI}{N}\Delta t \tag{16}$$

Using the same philosophy we can define other probabilty transition as following:

- The probabilty transition of making a move from $I \to R$ is

$$P(I \to R) = bI\Delta t \tag{17}$$

- The probabilty transition of making a move from $R \to S$ is

$$P(R \to S) = cR\Delta t \tag{18}$$

As explained earlier, the SIRS model can be easily expanded. Thus we can find the transition probabilities for section 2.2 Vital dynamics (open system) and Figure 10 by using the same philosophy. The probabilty transition are defined as

$$P(B \to S) = eN\Delta t \tag{19}$$
$$P(S \to D) = dS\Delta t \tag{20}$$
$$P(I \to D) = (d + d_I)I\Delta t \tag{21}$$
$$P(R \to D) = dR\Delta t \tag{22}$$

Here we have defined two new subgroups B and D. Where B is a group of babies comming into to system S, and D is the group dead people when dying from the disease itself or a some other natural cause. For the case with vaccination we have

$$P(S \to R) = f\Delta t \tag{23}$$

How we update the number of people in each compartment group is explained in section (fullref til sec) og (ref tall her).

3 Method

In this section we will expalin how the different type method are implement into our code. Keep in mind that the code is writeen in C++.

3.1 Structure of the code

The complete code consist four classes and object oriented in a more pythonic way compared with the traditional C++ object orienting. The main.cpp reads and calls different class instances.

- solver.cpp : This class contains different method which has different task to solve the problem introduced. This class has to solver methods implemented. Runge Kutta method of fourth order and a Monte Carlo function. If one desires to use another solver, it can be easily implemented since most of the functions are independent of each other. Keep in mind to call the new solver in execute_solver method. This class also calculates the standard deviation and the variance.
- read_parameter.cpp : This class reads parameters from parameters.txt
- savetofile.cpp : This class saves result into a .txt file.

3.2 Runge-Kutta

For the sake of accuracy we have chosen to implement Runge-Kutta of fourth order to achieve best possible results. Keep in mind, as explained in the previous section the code is fully object oriented and vectorized. Meaning additional numerical methods for solving the ODE is easily implemented.

3.3 Sampling of Monte Carlo runs

When using Monte Carlo we must expect that for each run, the result will deviate by some decimals from the expected numerical result In order to minimize this deviation, we can sample the same simulation N_{sample} times over M Monte Carlo cycles. The idea is to look at the average of the result when sampling N_{sample} times. This is simply done by adding the point $N_{samples}$ times for each M_{mc} iteration and then divide by $N_{samples}$. From this we will get the average of the result which is signifcantly more smoother and nicer to analyze. From this we can also calculate the variance for each time point, and use this to calculate the standard deviation. The sampling method is implemented in the following way

```
for(int n = 0; n < nsamples; n++){
  for(int mc_step = 0; mc_step < MCcycles; mc_step++){
    SIR(mc_step,0) += S/double(nsamples);
    SIR(mc_step,1) += I/double(nsamples);
    SIR(mc_step,2) += R/double(nsamples);
  }
}
```

3.4 Variance

As explained previously we are interested in the average of the result computed, this means that we can compute the variance. The variance is simply defined as squared of the deviation from the mean value, mathematically

$$Var(X) = (X - \mu)^2 \tag{24}$$

Where X is our random result from each Monte Carlo cycles and μ is the computed average of X. In our case we want to look at the average variance from each $N_{samples}$. This is simply

$$Var(X) = \frac{(X - \mu)^2}{N_{samples}} \tag{25}$$

Notice that X and μ are matrices containing information about the deviation in S I and R data with number of Monte Carlo cycles as length in each dimension. This method is implemented in the following way

```
for(int n = 0; n < nsamples; n++){
  for(int j = 0; j < MCcycles; j++){
      varS(j) += (avgS(j)-S(j))*(avgS(j)-S(j))/(nsamples-1);
      varI(j) += (avgI(j)-I(j))*(avgI(j)-I(j))/(nsamples-1);
      varR(j) += (avgR(j)-R(j))*(avgR(j)-R(j))/(nsamples-1);
    }
  }
```

The goal of this is to have the variance such that we can calculate the standard deviation $\sigma = \{\sigma_S, \sigma_I, \sigma_R\}$ for each compartment group. This number simply tells us where the expected data should lie.

3.5 Standard deviation

As explained previously section (3.4) we want to estimate the standard deviation in order to have a idea of where the result should lie. The standard deviation is defined as

$$\sigma = \sqrt{\frac{\sum_i Var(X_i)}{N}} \tag{26}$$

This method is implemented in the following way

```
for(int n = 0; n < nsamples; n++){
  for(int j = 0; j < MCcycles; j++){
    sigmaS += sqrt(varS(j)/(nsamples*MCcycles));
    sigmaI += sqrt(varI(j)/(nsamples*MCcycles));
    sigmaR += sqrt(varR(j)/(nsamples*MCcycles));
  }
}
```

3.6 Monte Carlo simulation

In the section of theory we explained and derived the probability transitions from one compartment group to another. Using these probability transitions we can simply draw a random uniformed number r in the inverval $[0, 1]$ to check if $r \leq P_{i\to j}$. If the condition is fullfilled we then move a human from one compartment group i to j. This is done by subtract and adding 1 to temporary variables. The reason why use a temporary variables is that have several probability transitions conditions, and every event can happen at once. Thus we must draw a new number each time we check these transitions. When have checked all the conditions, we will then add the corresponding temporary variable to its compartment group. In the code, we have defined three variables as "susceptible", "infected" and recovered in the end are these added to S, I and R. A small part of the Monte Carlo method is presented down below

```
random_device rd;
mt19937_64 gen(rd());
uniform_real_distribution<double> RandomNumberGenerator(0.0,1.0);

 //probabilty transitions of moving from S to I to R
 S_to_I = (parameter[0]*S*I/N)*dt;
 I_to_R = parameter[1]*I*dt;
 R_to_S = parameter[2]*R*dt;

 susceptible = infected = recovered = 0;

 if(RandomNumberGenerator(gen) < S_to_I){
     susceptible -= 1;
     infected += 1;
 }

 if(RandomNumberGenerator(gen) < I_to_R){
     infected -= 1;
     recovered += 1;
 }

 if(RandomNumberGenerator(gen) < R_to_S){
     recovered -= 1;
     susceptible += 1;

 }
 S += susceptible;
 I += infected;
 R += recovered;
```

4 Results

In this section we will present the different result from the different type of SIRS model. Keep in mind that we have used $N_{mc} = 10000$ Monte Carlo cycles with $N_{samples} = 100$ sampling runs. In the simulation where RK4 where used, we used $N = 100000$ iterations with a $h = 1/N$. Important infomartion suc as parameter used are given in the title of the result.

4.1 SIRS model for closed system

In this section we will present the result when using different values of the recovery rate b. We used $b = \{1, 2, 3, 4\}$

4.1.1 b = 1

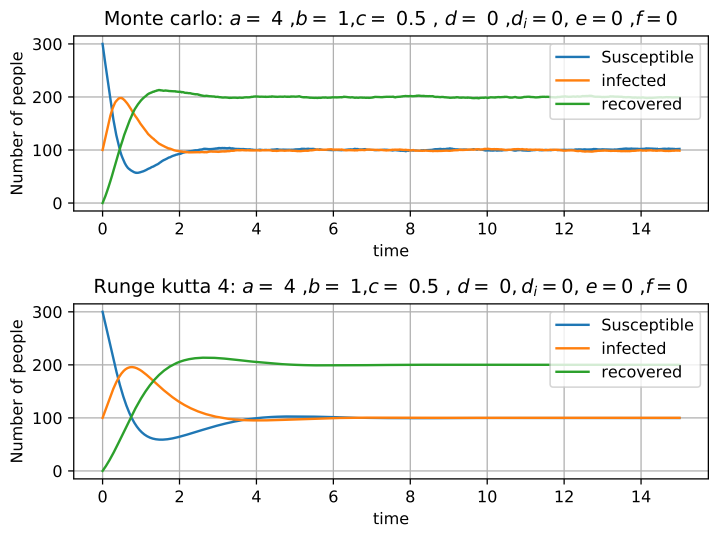


Figure 1: This figure shows to graphs, where the graphs use two different solving methods with same parameters. The above graph uses the Monte Carlo method, and the below graph uses RK4. The parameter used is given in the title of the figures. Notice that susceptible, infected and recovered are $S(t)$, $I(t)$ and $R(t)$. These functions are plotted with respect to time t.

4.1.2 b = 2

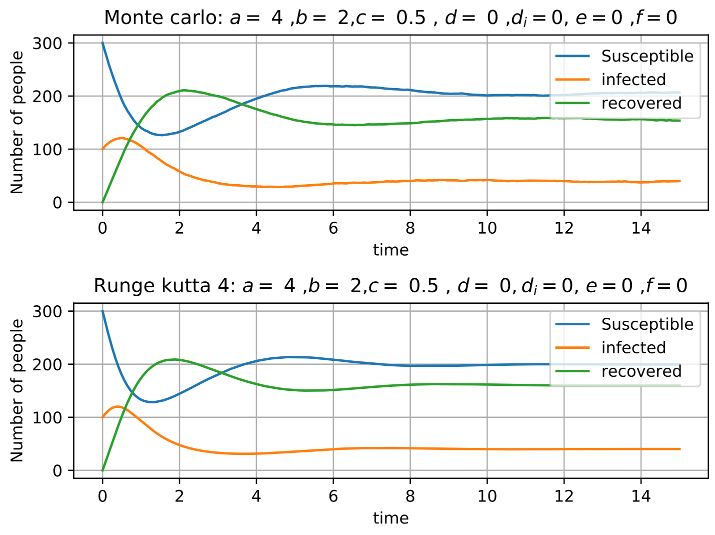


Figure 2: This figure shows to graphs, where the graphs use two different solving methods with same parameters. The above graph uses the Monte Carlo method, and the below graph uses RK4. The parameter used is given in the title of the figures. Notice that susceptible, infected and recovered are $S(t)$, $I(t)$ and $R(t)$. These functions are plotted with respect to time t.

4.1.3 b = 3

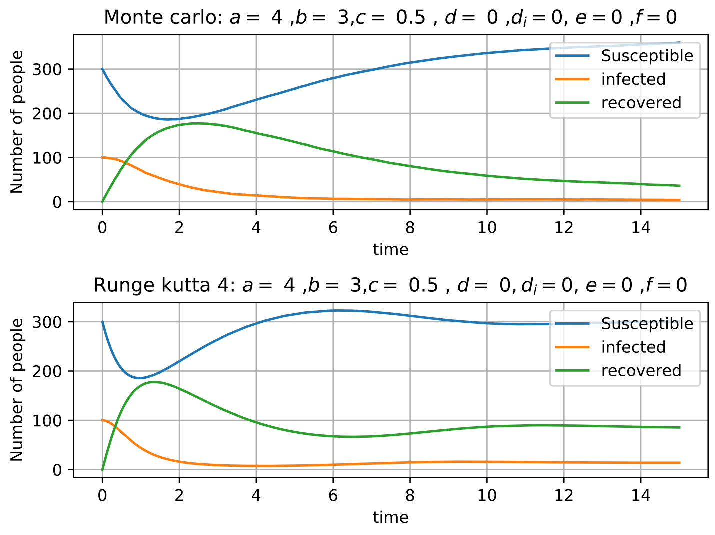


Figure 3: This figure shows to graphs, where the graphs use two different solving methods with same parameters. The above graph uses the Monte Carlo method, and the below graph uses RK4. The parameter used is given in the title of the figures. Notice that susceptible, infected and recovered are $S(t)$, $I(t)$ and $R(t)$. These functions are plotted with respect to time t.

4.1.4 b = 4

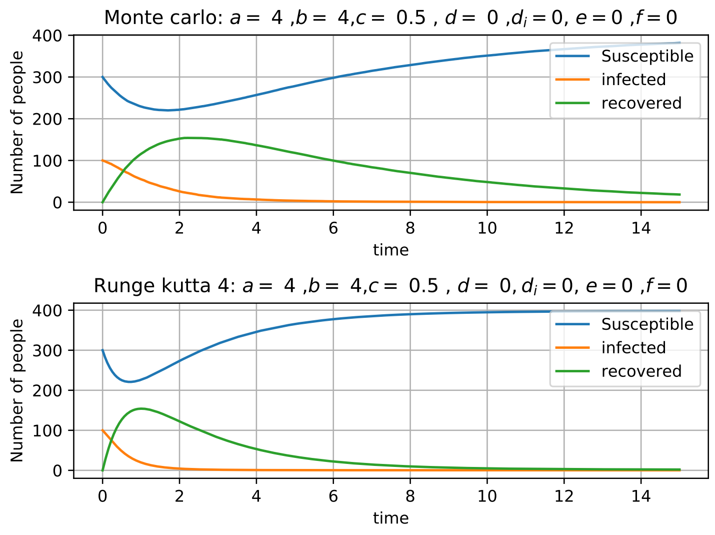


Figure 4: This figure shows to graphs, where the graphs use two different solving methods with same parameters. The above graph uses the Monte Carlo method, and the below graph uses RK4. The parameter used is given in the title of the figures. Notice that susceptible, infected and recovered are $S(t)$, $I(t)$ and $R(t)$. These functions are plotted with respect to time t.

4.2 SIRS model with vital dynamics:

Using the set of rate of recovery $b = \{1, 2, 3, 4\}$ as before

4.3 b = 1

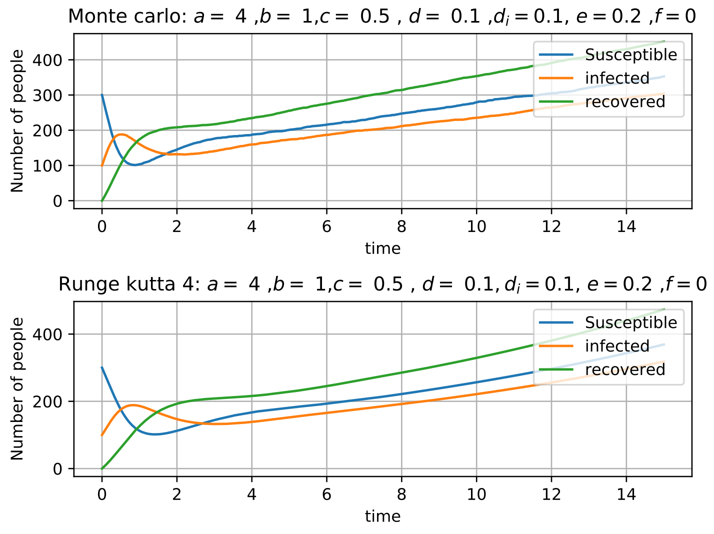


Figure 5

4.4 b = 2

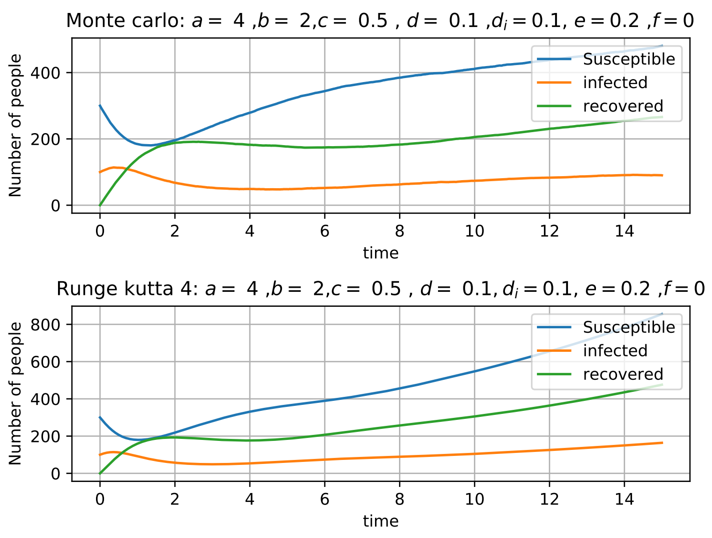


Figure 6

4.5 b = 3

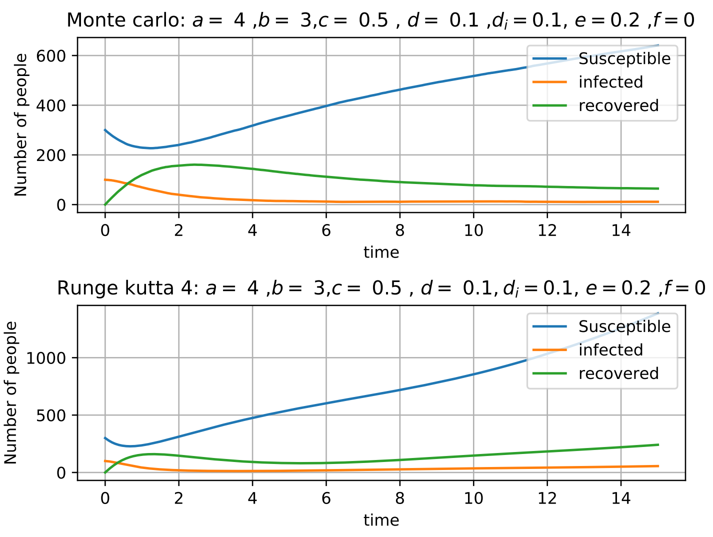


Figure 7

4.6 b = 4

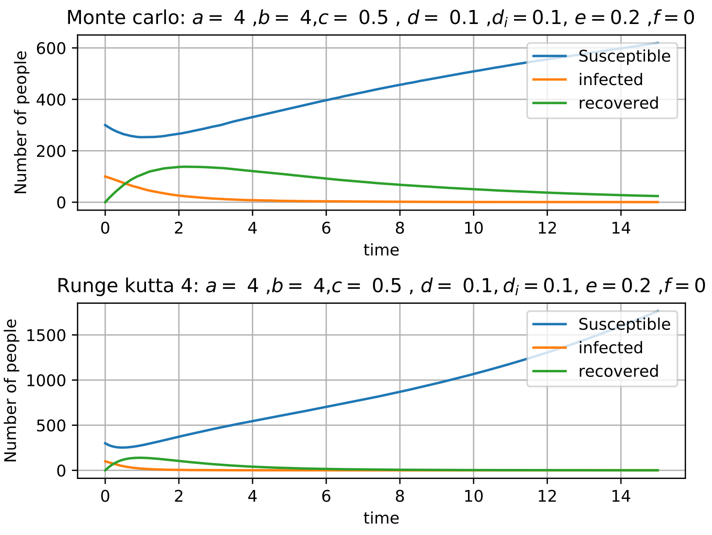


Figure 8

4.7 Seasonal variation, no death and birth rate

$a = a(t)$, $b = 1$, $c = 0.5$, $d_i = d = e = f = 0$

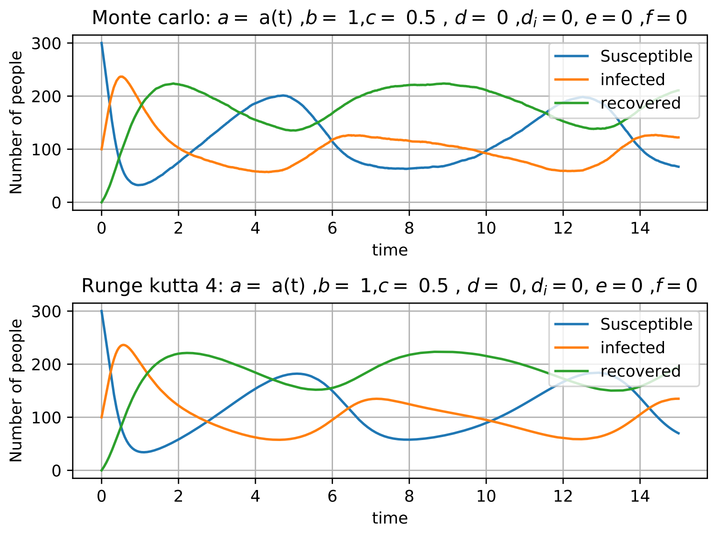


Figure 9

4.8 Vaccination, no death and birth rate

$a = 4$, $b = 1$, $c = 0.5$, $d_i = d = e = f = 0$ FIX THIS!

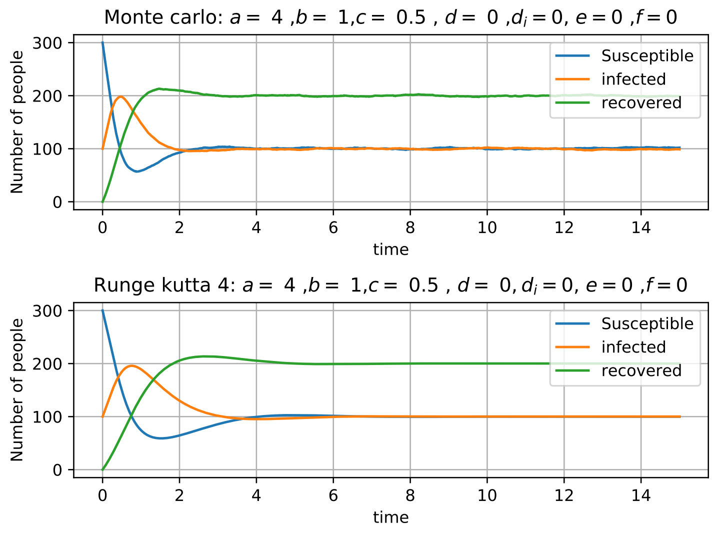


Figure 10

5 Discussion

5.1 SIRS closed system

In previous sections we did RK4 and Monte Carlo simulation for a scenario with no birth and death rate. As we analyze the graphs we notice minimal differences for $b = 1$ and $b = 2$.

5.2 SIRS with vital dynamics

6 Conclusion

We are the best, mUUUtherfuckers.

7 Further work

Theres seems to be a small bug in the C++ code. When analyzing the graphs produced by Monte Carlo and RK4 method, for some parameters there is a signifcantly large difference. This is clearly seen when looking at the y-axis. For future work we want to look deeper into this in order to find the bug, or to get a better understanding of why this difference between these two methods appears.