UiO : Kjemisk institutt
Det matematisk-naturvitenskapelige fakultet

# FYS4150, project 5

**Sondre Torp - Sondrt@student.matnat.uio.no**

# 1 Abstract

This report deals with the numerical solution of the coupled ordinary differential equations provided by the classical SIRS model of epidemiology. The traditional 4th order Runge-kutta(RK4) scheme is compared with the novel approach of using neural networks(NN), and Monte Carlo simulations(MCs). Autograd is used for the NN due to its ability to handle coupled ODEs.

While RK4 is better both in terms of accuracy and runtime, the NN also showed results, giving a mean squared on the order $10^{-6}$ after $10000$ epochs.The swish method was determined to be the best activation function. The MCs were a solid and closer to a real scenario of the disease. Of this reason the report proceeds with improving the SIRS model for the RK4 solution and MCs by including; Vital dynamics, Seasonal Variation, and Vaccination. Where it was again showed that vaccines, with propper management, are a crucial tool to hinder the potential spread of dangerous diseases.

# Innhold

# 2 Physical Theory

## 2.1 The SIRS

The SIRS model is the one of the simplest compartmental models, and is very similar to the SIR model, with one exception, this exception stopes the SIRS model from being solvable analythically.

The **SIRS** model considers an isolated population $N$, that consists of three compartments:

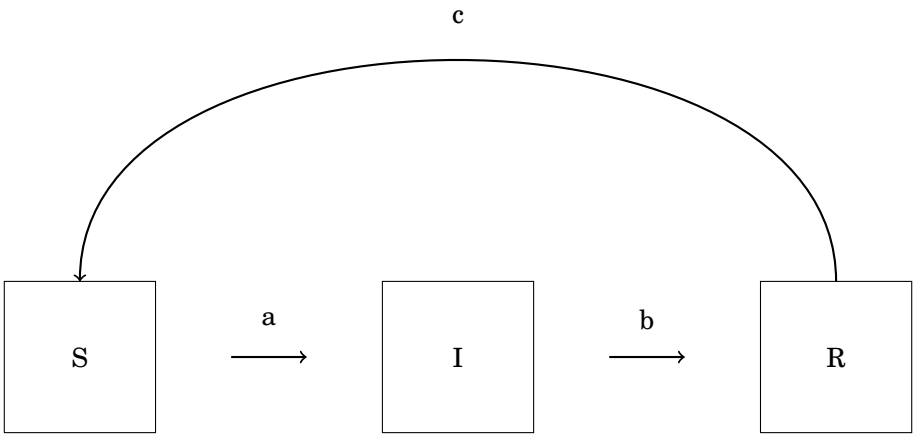
- **S**: The number of people without immunity to the disease.
- **I**: The number of people who are currently infected
- R: The number of past infected whom have developed an immunity to the disease.

These variables (**S, I, R**) represent the number of people in each compartment, and since the size of these are dependent on time, we make the precise number a function of time.($\mathbf{S}(t), \mathbf{I}(t), \mathbf{R}(y)$). For a specific disease in a specific population, these functions can be worked out to predict possible outbreaks and perhaps bring them under control.

The model is dynamic in that the number in each compartment may change over time. This is dynamic aspect is most obvious in an endemic disease. The main difference between the SIRS model and the SIR model is the fact that in the SIR model a member of the population can never join the susceptible group, ($S(t)$), again after leaving, such as the disease measles. This type of disease can not break out again until the number of susceptible people has built back up. While in the SIRS model, you can lose your immunity to the disease. A person can move from one group to another as indicated in the figure 1 below.

The rate of transmission $a$, the rate of recovery $b$ and the rate of immunity lose $c$, helps describing the flow of people moving between the groups. The group is assumed to be mixed homogeneously and the total population remains constant.

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




Figur 1: A flow diagram in which the boxes represent the different compartments and the arrows the transition between the compartments for a SIRS model.

If we assume that the time scale is much smaller than the average person's lifetime, than the effect of the birth and death rate of the population can be ignored. For these assumption we are given a set of coupled differential equations that we use to construct our model.

$$\frac{dS}{dt} = cR - \frac{aSI}{N} \tag{2}$$
$$\frac{dI}{dt} = \frac{aSI}{N} - bI \tag{3}$$
$$\frac{dR}{dt} = bI - cR \tag{4}$$

Though this set does not have a analytic solution, the equilibrium solutions are simple to obtain.

$$\frac{dS}{dT} = c(N - S - I) - \frac{aSI}{N} \tag{5}$$
$$\frac{dI}{dT} = \frac{aSI}{N} - bI \tag{6}$$
$$\tag{7}$$

### 2.1.1 The steady state

The steady state is found by setting both of these equations equal to zero. The fraction of people in each group will then be:

$$s^* = \frac{b}{a} \tag{8}$$
$$i^* = \frac{1 - \frac{b}{a}}{1 + \frac{b}{c}} \tag{9}$$
$$r^* = \frac{b}{c}\frac{1 - \frac{b}{a}}{1 + \frac{b}{c}} \tag{10}$$

Each fraction must be between $(0 - 1)$ and the three factions must sum up to $1$. This also shows that $b < a$ for the disease to establish itself in the population.

## 2.2 Improvements to SIRS

The same principles as for the simple model are utilized to extend the model to include more details about the population and the disease.

### 2.2.1 Vital dynamics

Vital dynamics are added so that the model can describe the spread of the diseases which occur over longer stretches of time. If $e$ is introduced as the brith rate, $d$ and $d_I$ as death rate, and death rate for the infected people due to the disease, then the differential equations, assuming all babies born are susceptible, are given by:

$$\frac{dS}{dT} = cR - \frac{aSI}{N} - dS + eN \tag{11}$$
$$\frac{dI}{dT} = \frac{aSI}{N} - bI - dI - d_I I \tag{12}$$
$$\frac{dR}{dT} = bI - cR - dR \tag{13}$$

### 2.2.2 Seasonal Variation

Seasonal variations can also be added to better represent diseases such as influenza, where the rate of transmission depends largely on the time of the year. During cold months individuals are more likely to spend time in closer proximity to one another, resulting in a higher rate of transmission. We change the transmission rate ($a$), so that it oscillates.

$$a(t) = A\cos(\omega t) + a0 \tag{14}$$

where $a0$ is the average transmission rate, $A$ is the maximum deviation form $a0$, and $\omega$ is the frequency of oscillation.

### 2.2.3 Vaccination

Diseases with available vaccinations allow people to move directly from $S$ to $R$, breaking the cyclic structure.[**zaman2008stability**] Here it is assumed that a susceptible individual's choice to become vaccinated does not depend on how many other susceptible are vaccinated or the anti vaccination movement. We can assume that the rate of vaccination, $f$, depends on time, since this rate may oscillate during the course of a year and/or increase as awareness and medical research increase. The differential equations become:

$$\frac{dS}{dT} = cR - \frac{aSI}{N} - f \tag{15}$$

$$\frac{dI}{dT} = \frac{aSI}{N} - bI \tag{16}$$

$$\frac{dR}{dT} = bI - cR + f \tag{17}$$

# 3 Computational Theory & implementation

The code used for all models of SIRS can be found here: github.

## 3.1 4h order Runge-Kutta

The 4th order Runge-Kutta(RK4) is one of the classic methods for numerical integration of ODE models. For a brief introduction of RK4 refers to Wikipedia.

For this problem; consider the following initial value problem of ODE

$$\frac{dy}{dt} = f(t, y) \tag{18}$$

$$y(t_0) = y_0 \tag{19}$$

where $y(t)$ is the unknown function which we would like to approximate.

The iterative formula of RK4 method for solving ODE 19 is as follows:

$$y_{n+1} = y_n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{20}$$

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + \frac{\Delta t}{2}, y_n + \frac{k_1 \Delta t}{2}) \\ k_3 &= f(t_n + \frac{\Delta t}{2}, y_n + \frac{k_2 \Delta t}{2}) \\ k_4 &= f(t_n + \Delta t, y_n + k_3 \Delta t) \\ t_n + 1 &= t_n + \Delta t \\ n &= 0, 1, 2, 3, ... \end{aligned}$$

The SIRS model is defined as follows:

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

with $a, b, c, S, I, R$ and $N$ as previously defined.2.1

According to the general iterative formula 20, the iterative formulas for $S(t), I(t), R(t)$ of SIRS model can be written out:

$$S_{n+1} = S_n + \frac{\Delta t}{6}(k_1^S + 2k_2^S + 2k_3^S + k_4^S)$$

$$k_1 = f(t_n, S_n, I_n, R_n) = cR - \frac{aSI}{N}$$

$$k_2 = f(t_n + \frac{\Delta t}{2}, S_n + \frac{k_1^S \Delta t}{2}, I_n + \frac{k_1^S \Delta t}{2}, R_n + \frac{k_1^R \Delta t}{2}) = -\frac{a}{N}(S_n + \frac{k_1^S \Delta t}{2})(I_n + \frac{k_1^I \Delta t}{2}) + c(R_n + \frac{k_1^R \Delta t}{2})$$

$$k_3 = f(t_n + \frac{\Delta t}{2}, S_n + \frac{k_2^S \Delta t}{2}, I_n + \frac{k_2^S \Delta t}{2}, R_n + \frac{k_2^R \Delta t}{2}) = -\frac{a}{N}(S_n + \frac{k_2^S \Delta t}{2})(I_n + \frac{k_2^I \Delta t}{2}) + c(R_n + \frac{k_2^R \Delta t}{2})$$

$$k_4 = f(t_n + \Delta t, S_n + k_3^S \Delta t, In + k_3^I \Delta t, R_n + k_3^R \Delta t) = -\frac{a}{N}(S_n + k_3^S \Delta t)(I_n + k_3^I \Delta t) + c(R_n + k_3^R \Delta t)$$

$$
\begin{aligned}
I_{n+1} &= I_n + \frac{\Delta t}{6}(k_1^I + 2k_2^I + 2k_3^I + k_4^I) \\
k_1^I &= \frac{aS_nI_n}{N} - bI_n \\
k_2^I &= \frac{a}{N}(S_n + \frac{k_1^S\Delta t}{2})(I_n + \frac{k_1^I\Delta t}{2}) - b(I_n + \frac{k_1^I\Delta t}{2}) \\
k_3^I &= \frac{a}{N}(S_n + \frac{k_2^S\Delta t}{2})(I_n + \frac{k_2^I\Delta t}{2}) - b(I_n + \frac{k_2^I\Delta t}{2}) \\
k_4^I &= \frac{a}{N}(S_n + k_3^S\Delta t)(I_n + k_3^I\Delta t) - b(I_n + k_3^I\Delta t)
\end{aligned}
$$

$$
\begin{aligned}
R_{n+1} = R_n &= \frac{\Delta t}{6}(k_1^R + 2k_2^R + 2k_3^R + k_4^R) \\
k_1^R &= bI_n - cR_n \\
k_2^R &= b(I_n + \frac{k_1^I\Delta t}{2}) - c(R_n + \frac{k_1^R\Delta t}{2}) \\
k_3^R &= b(I_n + \frac{k_2^I\Delta t}{2}) - c(R_n + \frac{k_2^R\Delta t}{2}) \\
k_4^R &= b(I_n + k_3^I\Delta t) - c(R_n + k_3^R\Delta t)
\end{aligned}
$$

Pew, now note that since the population $N = S(t) + I(t) + R(i)$ is constant, they will have $0 = \frac{dS}{dt} + \frac{dI}{dt} + \frac{dR}{dt}$. Meaning that only two of the three ODEs are independent and are thus sufficient to solve the ODEs. In our code, we ignore this fact, so that we can utilize the full power of our cpu.

## 3.2 Monte Carlo Simulation

This project consists of different cases with a different degree of complexity. For the basis case it is enough to use the Monte Carlo method. This method utilizes the fact that a large number of experiments converges towards the expectation value. When the probability of doing something is added, the system utilizes the Metropolis Algorithm. The system has to do a random choice of acceptance or denial of the case. This random choice is done with the Random Number Generator from NumPy[**NumPy**]. A random number i created between $(0-1)$. If the number is less than the probability term, accept the new state. If the number is greater than the probability for change then discard the change. The procedure is outlined in four steps:

- Chose a transition state randomly.
- Find the probability for the transition.
- A RNG is used to chose a number between $(0-1)$. Now if the probability term is less then the RNG number, reject the transition. If not, accept the tranistion.
- Update the system with the transferred states.

## 3.3 Neural networks

First of all we wanted to make sure that we could train the neural network(NN) to reproduce the RK4 solution. This is certainly a prerequisite toe the idea working. The NN will here output three values, one for each concentration. We also chose to use the Swish function due to this article 6. We later tested for OTHER.

We then trained our network to reproduce the solution, one layer network with 8 nodes to output all three concentrations pretty accurately. This confirmed that the solution could be represented by a neural network. (Se the section for results) REF.

The next major issue is how do we get the relevant derivatives. The solution method developed here relies on using optimization to find a set of weights that produces a NN whose derivatives are consistent with the ODE equation. The NN outputs three concentrations, and we need the time derivatives of them. In Autograd we found three options; grad, elementwise_grad and Jacobian. We cannot use grad because our function is not scalar. We cannot use elementwise_grad because that would give us the wrong shape. This left us with one option. This was a little weird because it gave an output that was 4-dimensional, as expected from the documentation, the first and third output where related to our time steps, then we used some fancy sorting to see if the data was comparable to the derivatives defined by the ODEs. This was inefficient, due to it requiring a lot of calculations to create the jacobian.

Finally we solved the system with our NN, firstly defining a time grid to solve it on, defining our objective function, taking care of initial conditions, running the optimization and getting the solution from the NN.

# 4 Result & Discussion

## 4.1 SIRS modelling

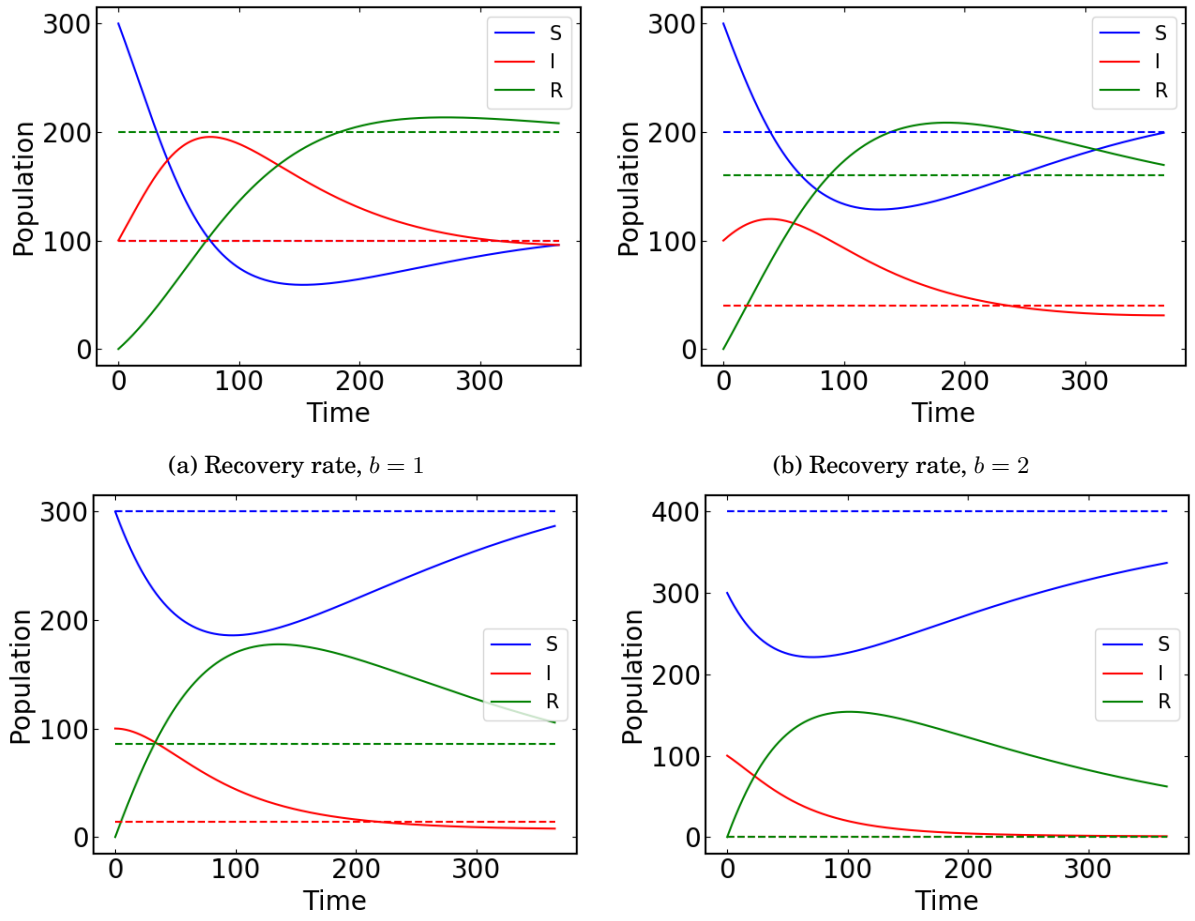
Here we confirmed that the recovery rate, $b$, should be smaller than the rate of transmission, $a$, to sustain a disease. We saw that there is a surprisingly good match between the RK4 and the MC model, but the MC model needs more time, approximately 3 years, to stablize and find a steady - state in the population. Finally we figured that the MC simulation crashes if the number of infected people reaches zero. Details can be read below.

The equilibrium expectation values and corresponding standard deviation, after 10 years of time or 3650 days, were found to be: [**Nobody06**]

| | S(t) | I(t) | R(t) |
|---|---|---|---|
| Mean | 108.81 | 110.11 | 181.06 |
| Stdev | 39.48 | 32.06 | 28.24 |

### 4.1.1 4th order Runge-Kutta on the ODEs from SIRS, testing recovery rate.

The simplest form of SIRS models differential equations solved with 4th order Runge - Kutta, then plottet. All plots in figure: 2, have a black line that represent the change in total population. One can see there is not much change in population, as expected, at this moment. The equilibrium-states, discussed in section: 2.1.1, are plottet with stapled lines. These solutions are only plottet for one year, due to their obvious approach of the equilibrium-states. The difference between these figures are due to different recovery rate. One can see form these figures that after the initial burst of infection most develops an immunity after a short time and the disease reaches a steady-state. We can also confirm that if the transmission rate is larger then the recovery rate ($b < a$) the disease mange to get some kind of foothold, to sustain itself in the population.




(a) Recovery rate, $b = 1$

(b) Recovery rate, $b = 2$

(c) Recovery rate, $b = 3$

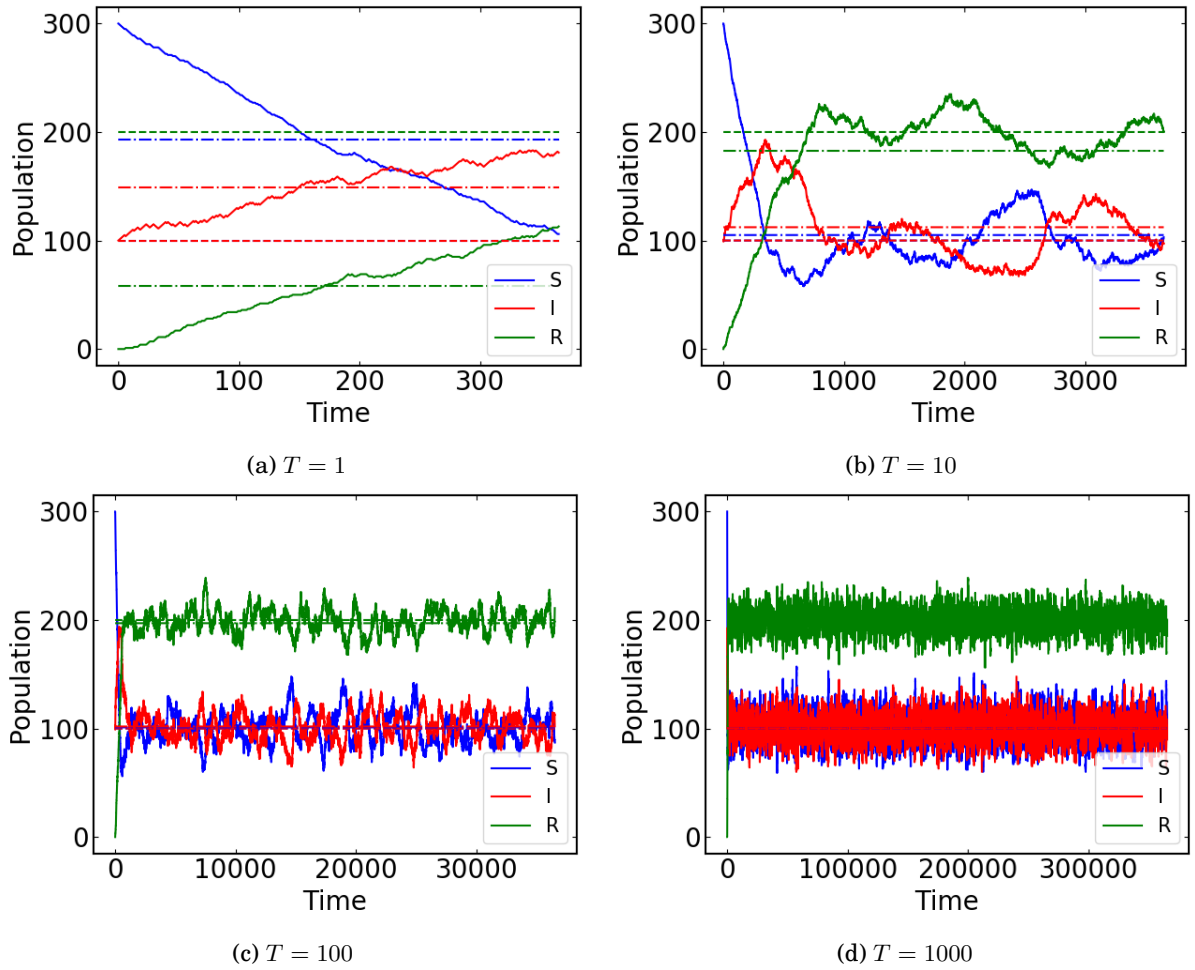
(d) Recovery rate, $b = 4$

Figur 2: 4th order Runge-Kutta on ODEs given for SIRS model, where the only difference is the recovery rate,b, that is equal to $1, 2, 3, 4$ respectivly

#### 4.1.2 Neural networks

#### 4.1.3 Monte Carlo simulation of simple SIRS model, using different time.

Firstly we quickly understood that one year was simply not enough time to get to a proper stady-state, so we look at how the simulation would look if we ran it for $1, 10, 100, 1000$ years. 3 And we quickly figured that $5$ to $10$ years should suffice, depending on what effects we want to observe. Here the both the equilibrium line, same as before, and the mean of the Monte Carlo simulation are plottet, and even after $1000$ years, these lines were still off by $1$ or $2$.




(a) $T = 1$

(b) $T = 10$

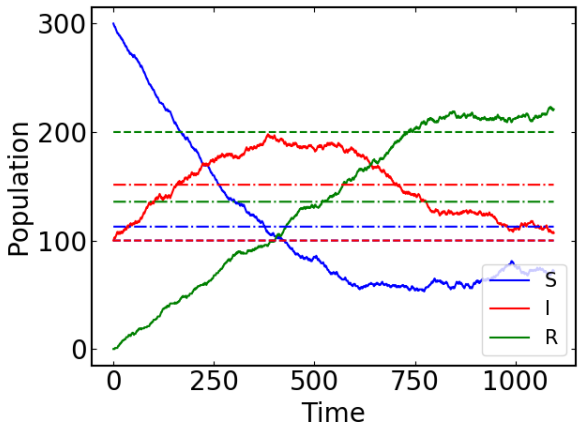
(c) $T = 100$

(d) $T = 1000$

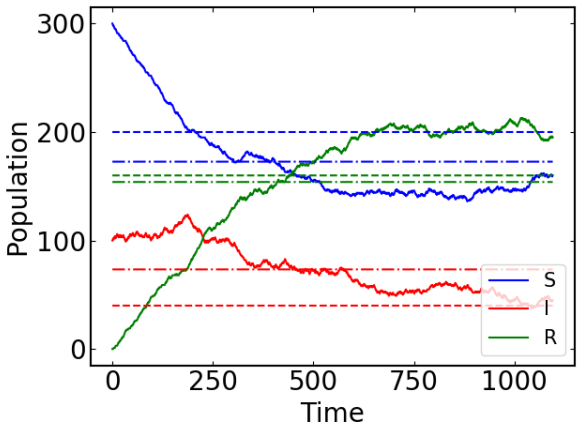
Figur 3: Monte Carlo simulation of the simple SIRS model, the dotted lines are the equilibrium values and the dot-dashed lines are the mean of the functions. As we can see from the plot they vary

#### 4.1.4 Monte Carlo simulation of simple SIRS model, testing recovery rate

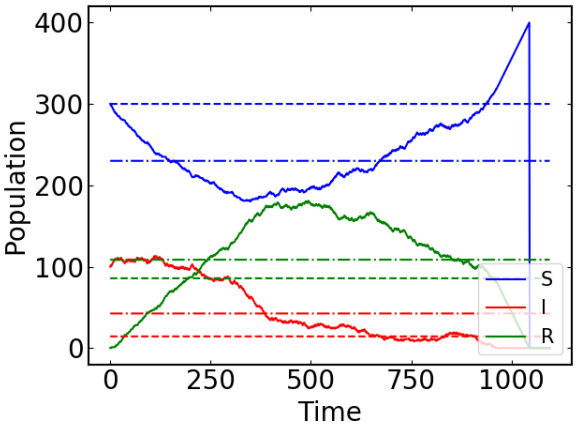
In these figures we notice something spectacular! The program crashes at higher recovery rate! This is due to the fact that there are no more infected people at some point in the program, and therefor no reason to track the infections development, if there is no infection. There are theoretical steady - state for the recovery rate,$b$,- values that are smaller then the transmission rate ,$a$, -values, but due to the randomness of the Monte Carlo Simulations the infection tend to die out by itself, which makes a lot of sense in a population that fluctuate. From here on out we will use $b = 1$ in all figures.



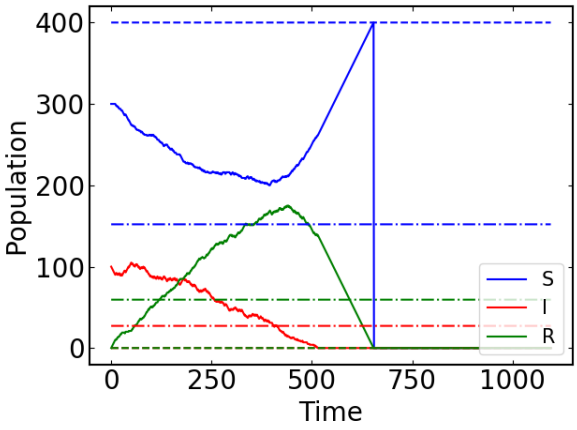

(a) $T = 3$ and $b = 1$



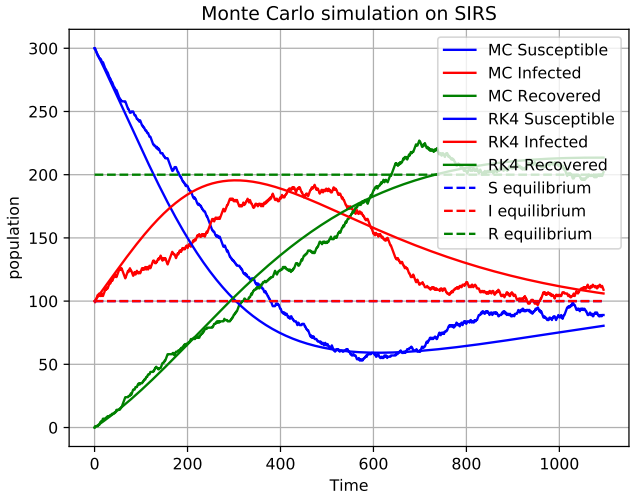

(b) $T = 3$ and $b = 2$




(c) $T = 3$ and $b = 3$




(d) $T = 3$ and $b = 4$




(e) $T = 3$ and $b = 1$, both MC and RK4 plottet on top of each other.

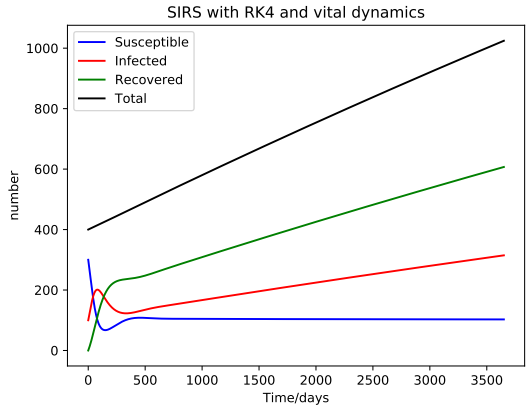
Figur 4

## 4.2 SIRS modelling with Improvements

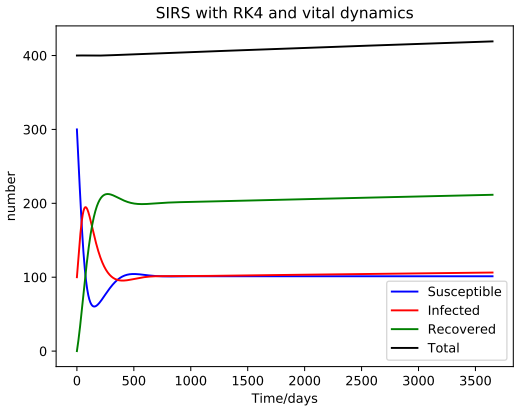
Here we present the results form the different versions of SIRS, Vital dynamics, Seasonal variation and Vaccination, and some of these combined.

### 4.2.1 Vital dynamics

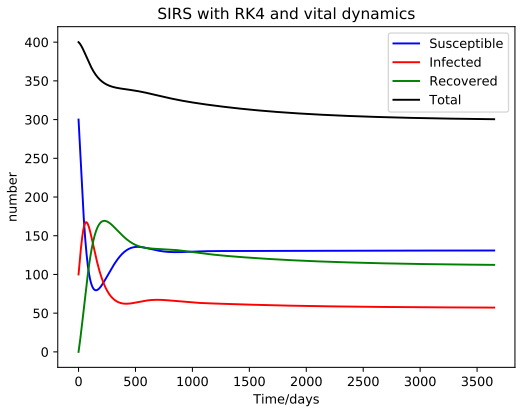
Vital dynamics are here included as presented in the theory section 2.2.1 for both RK4 and MC. There is some difference in the given death/birth - rates between RK4 and MC due to lack of normalization $dt$. It is easy to see that these numbers are vital for both the survival and death of the disease



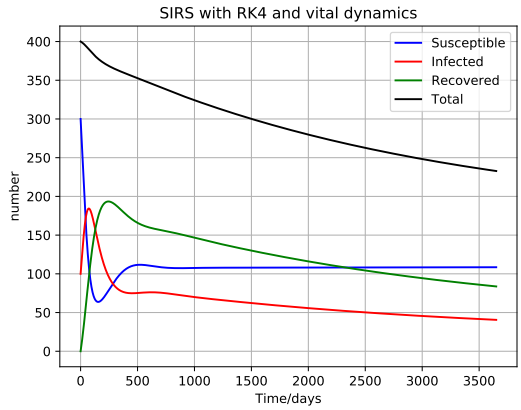

(a) High birth-rate($e = 0.05$), low death-rate, both in general, but also for the infected. ($d = 0.001, d_I = 0.01$)




(b) low birth-rate ($e = 0.005$), low death-rate, both in general, but also for the infected ($d = 0.001, d_I = 0.01$)

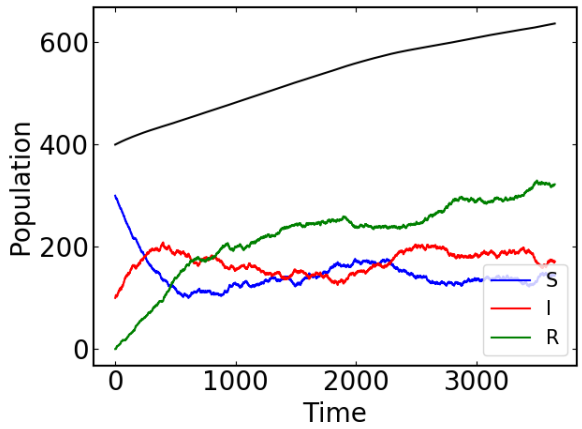



(c) high birth-rate ($e = 0.05$), high general death-rate and very high death-rate for the infected.

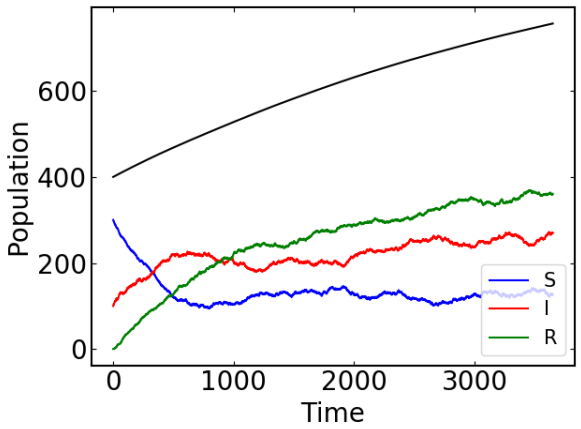



(d) low birth-rate ($e = 0.006$), low general death-rate($0.002$), high death-rate for infected($d_I = 0.10$)

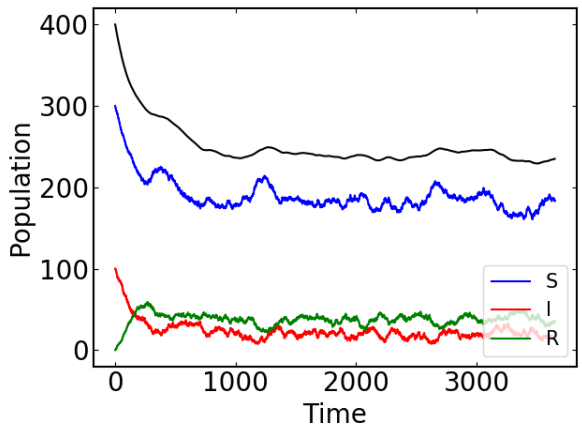
Figur 5: Looking at different birth-, death-, infected death - rates.



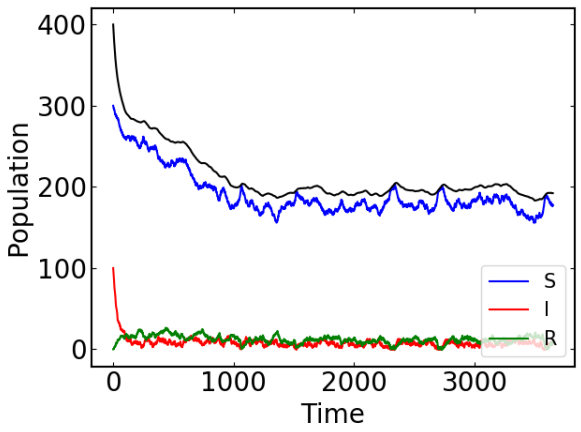

(a) $d = 0.0002, d_I = 0.0004, e = 0.0006$




(b) $d = 0.0002, d_I = 0.0001, e = 0.0006$




(c) $d = 0.0002, d_I = 0.001, e = 0.0006$

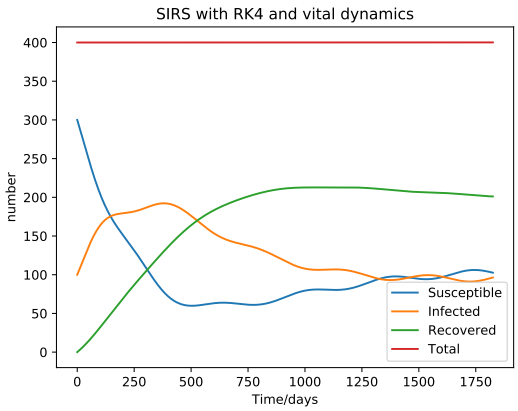



(d) $d = 0.0002, d_I = 0.03, e = 0.0006$

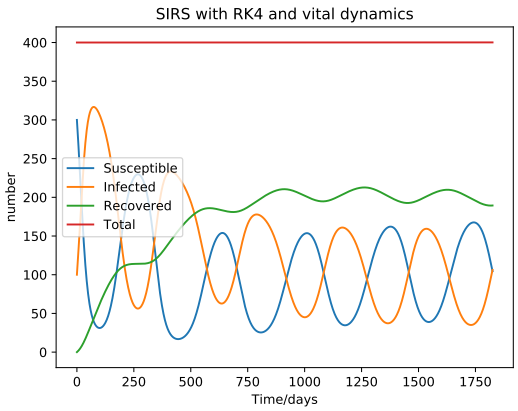
Figur 6: Main difference between the MC method and the RK4 is that $dt$ was not normalized in the cod, so smaller numbers had a much bigger impact.

### 4.2.2 Seasonal Variation

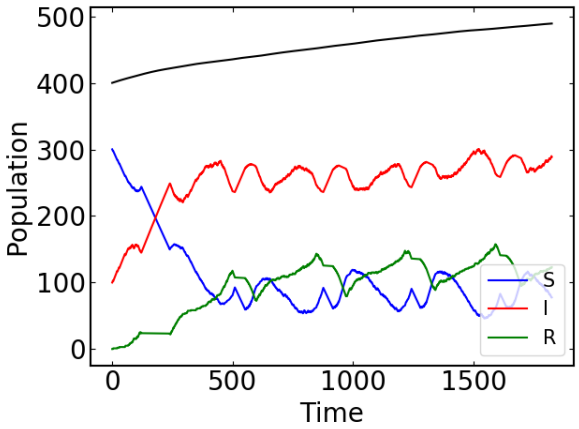
Here a Seasonal Variation is included 2.2.2, The value $A$ represents the maximum deviation from the infection rate $a_0 = 4$.



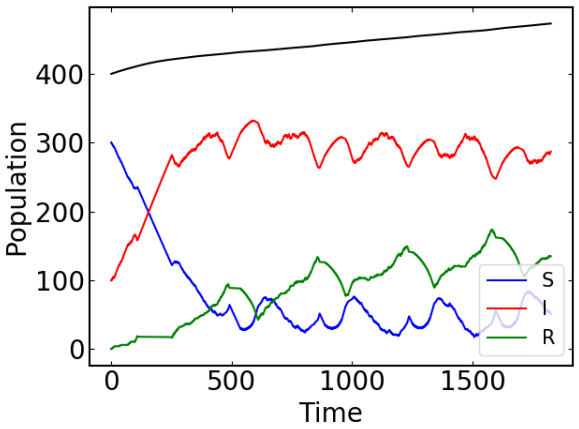

(a) RK4 $A = 1.2$




(b) RK4$A = 2.0$




(c) MC $A = 12$

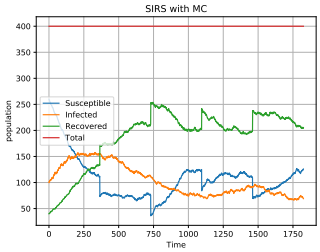



(d) MC $A = 20$

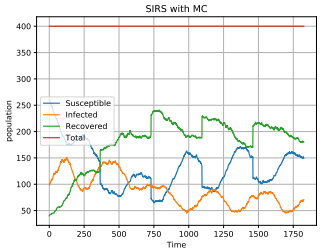
Figur 7: This figure shows the impact of seasonal variations on RK4 and MCS, notice how the seasonal variation impact is delayed in the MC scheme.

### 4.2.3 Vaccination

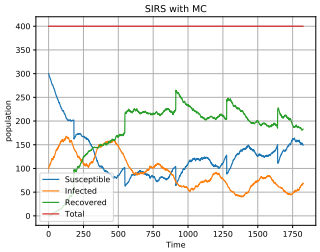
If we assume that vaccines work and the earth is round, than we can see what incredible tool vaccines are for managing diseases. These results shows how efficient vaccine in combination with seasonal variation can be, when you only vaccinate $10\%$ of the population once per year.



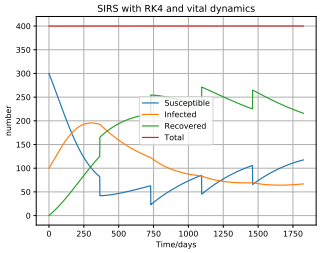

(a) MC vaccine



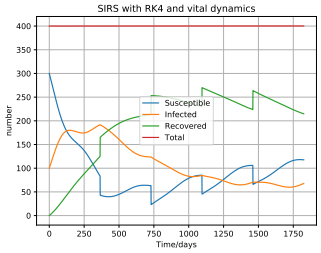

(b) MC vaccine no timing



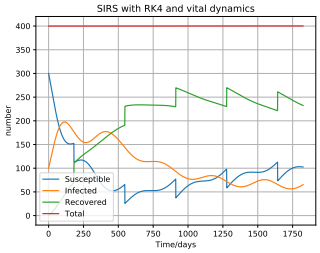

(c) RK4 vaccine seasonal timing




(d) RK4 vaccine




(e) RK4 vaccine no timing




(f) RK4 vaccine seasonal timing

Figur 8: The SIRS model, figure a) and d) are clean SIRS models with the effect of a yearly vaccine, b) and e) include the seasonal variation of the disease, lastly in figure c) and f) we have timed the vaccine to be given right before the winter, where it is assumed to be at its worst. (at least in this model).

# 5 Conclusion

It seems that the SIRS model is a great system dynamics model for representing real diseases, in general a great foundation to build on for predicting disease, in this case, and how they establish themselvs within the population. We have also seen that it is very versatile, with a few improvements tested out, and combined in this report. It was shown that a Monte Carlo simulation were a great way of approaching the differential equations in a more realistic manner, if given a large enough dataset.

# 6 References:

note: bibtex did not want to work with me on this project.


- Yang, Wan, Alicia Karspeck, and Jeffrey Shaman. "Comparison of filtering methods for the modeling and retrospective forecasting of influenza epidemics.PLoS computational biology 10.4 (2014): e1003583.
- Zaman, Gul, Yong Han Kang, and Il Hyo Jung. Stability analysis and optimal vaccination of an SIR epidemic model.BioSystems 93.3 (2008): 240-249.
- Ramachandran, Prajit, Barret Zoph, and Quoc V. Le. Searching for activation functions."(2018).

# 7 Appendix

## 7.1 Supervised Learning, some results:

| func | node | epoch | step | MSE | R2 |
|---|---|---|---|---|---|
| swis | 100 | 100 | 01 | 4301.860 | -1.504686 |
| swis | 100 | 100 | 001 | 4383.970 | -0.062676 |
| swis | 100 | 100 | 0001 | 18649.730 | -1612.388089 |
| swis | 100 | 1000 | 01 | 5405.858 | -0.947049 |
| swis | 100 | 1000 | 001 | 4996.257 | -1.148733 |
| swis | 100 | 1000 | 0001 | 4068.276 | -0.079732 |
| swis | 100 | 10000 | 01 | 5405.352 | -0.946243 |
| swis | 100 | 10000 | 001 | 5405.650 | -0.946284 |
| swis | 100 | 10000 | 0001 | 5395.676 | -0.955657 |
| swis | 1000 | 100 | 01 | 5344.535 | -0.9690825 |
| swis | 1000 | 100 | 001 | 4301.586 | -1.5146678 |
| swis | 1000 | 100 | 0001 | 7687.034 | -7.0514296 |
| swis | 1000 | 1000 | 01 | 5406.056 | -0.9465423 |
| swis | 1000 | 1000 | 001 | 5405.020 | -0.9540634 |
| swis | 1000 | 1000 | 0001 | 4381.460 | -1.5773855 |
| swis | 1000 | 10000 | 01 | 5408.282 | -0.9504758 |
| swis | 1000 | 10000 | 001 | 5405.543 | -0.9462314 |
| swis | 1000 | 10000 | 0001 | 5407.491 | -0.9473403 |
| tan | 100 | 100 | 01 | 4722.055 | -1.4382791 |
| tan | 100 | 100 | 001 | 4541.822 | -63.0650826 |
| tan | 100 | 100 | 0001 | 18327.134 | -2164.3759400 |
| tan | 100 | 1000 | 01 | 5389.155 | -0.9489306 |
| tan | 100 | 1000 | 001 | 4560.229 | -1.4495536 |
| tan | 100 | 1000 | 0001 | 4609.627 | -47.9667788 |
| tan | 100 | 10000 | 01 | 5406.032 | -0.9464465 |
| tan | 100 | 10000 | 001 | 5405.054 | -0.9468466 |
| tan | 100 | 10000 | 0001 | 5136.405 | -1.0999677 |
| tan | 1000 | 100 | 01 | 5301.474 | -1.0137502 |
| tan | 1000 | 100 | 001 | 4726.264 | -1.5468762 |
| tan | 1000 | 100 | 0001 | 6837.188 | -10.2774431 |
| tan | 1000 | 1000 | 01 | 5408.170 | -0.9473984 |
| tan | 1000 | 1000 | 001 | 5401.553 | -0.9657959 |
| tan | 1000 | 1000 | 0001 | 4568.399 | -1.4008756 |
| tan | 1000 | 10000 | 01 | 5530.394 | -0.9048310 |
| tan | 1000 | 10000 | 001 | 5408.057 | -0.9453489 |
| tan | 1000 | 10000 | 0001 | 5408.827 | -0.9476682 |
| arctan | 100 | 100 | 01 | 4695.224 | -1.3245204 |
| arctan | 100 | 100 | 001 | 3061.016 | -6.1246861 |
| arctan | 100 | 100 | 0001 | 18306.907 | -1982.2542040 |
| arctan | 100 | 1000 | 01 | 5353.242 | -0.9334961 |
| arctan | 100 | 1000 | 001 | 4352.663 | -1.4293472 |
| arctan | 100 | 1000 | 0001 | 3198.084 | -5.7849874 |
| arctan | 100 | 10000 | 01 | 5409.115 | -0.9443223 |
| arctan | 100 | 10000 | 001 | 5399.718 | -0.9443793 |
| arctan | 100 | 10000 | 0001 | 5029.071 | -1.0632415 |
| arctan | 1000 | 100 | 01 | 5102.813 | -1.0711406 |
| arctan | 1000 | 100 | 001 | 4579.046 | -1.4685736 |

See github.come/sondrt/