The SIR Model for Disease Transmission with Euler's and Fourth-Order Runge-Kutta Numerical Solutions

Aidan Sirbu, Katie Brown
AM 2814G: Lab 4B

Western University — March 25, 2021

Abstract

The objective of this investigation was to explore and compare several methods of numerically solving systems of ordinary differential equations in the context of modelling disease transmission. To achieve this, two methods within the Runge-Kutta family of ODE solver's were used, Euler's method and Runge-Kutta method of order 4 (RK4). The way in which each method was coded allowed for both to produce equivalently accurate solutions, the difference being the time needed to execute. Euler's method was found to execute 630 times slower than RK4 to achieve the same accuracy in its solution.

The SIR model for the spread of disease is a simple epidemiological framework for modelling outbreaks using a system of coupled differential equations representing the members of a population that are susceptible to, infected by, or recovered/have died from a disease. The behaviour of this model is largely governed by the *contact ratio q* and basic reproductive ratio R_0, which represent the average number of close contacts, and secondary infections caused by a single infected individual, respectively. The RK4 solutions to the SIR model were used to explore how varying these parameters would impact whether the outbreak would grow into an epidemic, as well as the maximum number of simultaneous infections. It was found that minimizing the values of q and R_0 would slow the spread of the outbreak and reduce I_{max}, thereby lessening the harm done. This demonstrated the importance of these numerical methods of solving ODE's by illustrating the important insights that can be drawn from them.

1 Introduction and Background

1.1 SIR Model

The Susceptible-Infected-Recovered (SIR) model is a simple and well-known deterministic framework for predicting the spread of infectious diseases. It involves constructing a system of coupled differential equations representing the changes in the members of a population who can be susceptible to, infected by, or recovered from the disease. The susceptibles (*S*) are the members of the population that could become infected. The infected (*I*) are those who currently have the disease. When they recover or die they are classified as removed (*R*), and are assumed to no longer be susceptible to the disease.

As individuals become infected, the number of susceptibles in a population decreases at a rate of

$$\frac{dS}{dt} = -rIS \tag{1}$$

where r is the growth rate, determined by how contagious the disease is and the rate of contact between the infected and susceptibles. Those who are infected are removed at a rate given by

$$\frac{dR}{dt} = Ia \tag{2}$$

where a is the combined rate of death and recovery known as the removal rate. From these two equations, it then naturally follows that the number of individuals infected at any time is given by

$$\frac{dI}{dt} = -\frac{dS}{dt} - \frac{dR}{dt} = rIS - aI \tag{3}$$

We set the initial conditions for these values to be S_0, I_0, and R_i, and assume that there is no natural immunity in the population, so $R_i = 0$. This model is based on the assumption that the pandemic occurs over a short enough period that the population remains constant. That is,

$$S + I + R = I_0 + S_0 \tag{4}$$

One of the most important questions to ask about an infectious disease is *will the epidemic grow?* In other words, will members of the population become infected faster than they recover? From equation (3), we see that this is the case when $rIS_0 > aI$, or $\frac{rS_0}{a} > 1$. The basic reproductive ratio, $R_0 = \frac{rS_0}{a}$, is defined as the average number of infections caused by one primary infection [1]. It is then evident that the disease will spread if $R_0 > 1$.

During large disease outbreaks, much of the harm comes from exceeding the capacity of the health care system. It is thus important to predict the maximum number of infected individuals at one time, to be able to prepare for the surge in hospital admissions. We can combine equations (1) and (3) to produce

$$\frac{dI}{dS} = \frac{rIS - aI}{-rIS} = \frac{a}{rS} - 1 \tag{5}$$

It is now helpful to introduce a new parameter $q = \frac{r}{a}$, known as the *contact ratio,* which describes the fraction of the susceptible population that comes in contact with an infected [1]. Knowing that the curves of I open downwards, by Fermat's Theorem the maximum value of I will occur when $\frac{dI}{dS} = 0$, and it is simple to observe that this will occur when $S = \frac{1}{q}$ [2]. Equation (5) can then be integrated to yield I(S), the exact number of infected at a corresponding number of susceptible:

$$\begin{aligned} I &= \int \frac{1}{qS} - 1 dS \\ &= \frac{1}{q}\ln(S) - S + C \\ &= I_0 + S_0 - S + \frac{1}{q}\ln(S) - \frac{1}{q}\ln(S_0) \end{aligned} \tag{6}$$

To find the maximum number of infected, substitute $S = \frac{1}{q}$ to yield:

$$I_{max} = I_0 + S_0 - \frac{1}{q}(1 + \ln(qS_0)) \tag{7}$$

The final factor that is important to predict is how many total people will catch the disease. In other words, what will R_{end} be when $I = 0$? Transforming equations (6) and (4) to be in terms of the parameters at the *end* of the outbreak, then combining the two equations and substituting $q = \frac{R_0}{S_0}$ produces an implicit equation for R_{end}, the total number of cases over the course of the entire outbreak, in terms of just the reproductive ratio and initial conditions:

$$R_{end} = e^{(\frac{R_0}{S_0}(R_{end} - I_0 - S_0) + \ln(S_0))} + I_0 + S_0 \tag{8}$$

These three questions - will the outbreak grow, what will be the maximum number of infected, and how many people will catch the disease - are some of the most important factors in predicting the real impact of an outbreak. They will be further investigated in the context of the numerical solutions to equations (1), (2), and (3) in Section 3, using the techniques outlined in Section 2.

1.2 Euler's Method for Numerically Solving Systems of Ordinary Differential Equations

Often times, differential equations have no explicit solution formula. Euler's method provides a numerical method by which one can solve an initial value problem by approximation. Firstly, consider the initial value problem (IVP):

$$\begin{cases} y' = f(t, y) \\ y(0) = y_0 \\ t \text{ in } [a, b] \end{cases} \tag{9}$$

Take the first of equations (9) which represents a first-order, non-autonomous ordinary differential equation. The left hand side is the derivative of the function y, while the right hand side contains a function in terms of t and y. The fact that it contains the term t makes the equation non-autonomous. One can imagine a differential equation as a field of slopes. Each point in this field is assigned a vector with some direction and magnitude. Since we know that *y'* is the slope of the equation *y*, the right-hand side of the first of equations (9) is therefore representative of the slope at point (t,y).

Euler's method takes advantage of the fact that the right hand side of the equation is the slope. The iterative formula for Euler's method is

$$\begin{aligned} \omega_0 &= y_0 \\ \omega_{i+1} &= \omega_i + hf(t_i, \omega_i) \end{aligned} \tag{10}$$

Euler's method takes its initial value ω_0 as the initial condition of the IVP y_0. One then starts at that value and follows the slope (or vector) belonging to that point (t_0, ω_0) for a distance *h*, also known as the *step size* to obtain the next value

ω_1. From ω_1, the process repeats itself over and over until a suitable termination criteria has been reached. This criteria is usually determined by the user choosing some range in which to conduct the numerical solution.

The error in Euler's method can be determined by following the absolute error

$$e_i = |y_i - \omega_i|$$

where y_i are the values of the exact analytical solution and ω_i are the values approximated by Euler's method. It is also known that decreasing the step size h in half results in cutting the error at each step approximately in half [3]. Furthermore, Euler's method is an order 1 ODE solver. While delving into the workings and reasoning behind order is beyond the scope of this investigation, one must understand that the higher the order of the ODE solver, the smaller the upper bound is on its global truncation error [3]. Essentially, the greater the order of the ODE solver, the more accurate it is. Euler's method is one of the least accurate ODE solvers. The order of ODE solvers can take on virtually any integer value using the Taylor methods, however, these can become quite cumbersome when working with complicated systems of equation such as in the SIR model. Instead we will turn our focus to a more popular and accurate model for solving these complicated systems known as the Runge-Kutta Method of order four.

1.3 Runge-Kutta Method of order four for Numerically Solving Systems of Ordinary Differential Equations

The Runge-Kutta family of ODE solver's are a group of well known iterative methods which actually include Euler's method. However, the most well known member of this family aside from Euler's is the *Runge-Kutta Method of order four (RK4)*. This method's popularity stems from the fact that it can approximate higher order ODE's with great accuracy without needing to include any higher order terms. The enhanced accuracy of this method is due to the way in which it determines the slope at each time step. Euler's method merely takes the slope by determining what the right-hand side of equation (9) yields after plugging in the point of interest. RK4 instead determines the slope at the beginning, end, and two in the middle of the time step h [4]. This more properly account for fast changing slope fields. The equation for RK4 is

$$\omega_{i+1} = \omega_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{11}$$

where the k_i are the estimations of the slope at various points in the time step and can be found as

$$\begin{aligned} k_1 &= f(t_i, \omega_i) \\ k_2 &= f\left(t_i + \frac{h}{2}, \omega_i + \frac{h}{2}k_1\right) \\ k_3 &= f\left(t_i + \frac{h}{2}, \omega_i + \frac{h}{2}k_2\right) \\ k_4 &= f(t_i + h, \omega_i + hk_3) \end{aligned} \tag{12}$$

Overall, the RK4 method is of order 4 which means its accuracy is significantly greater than that of Euler's method.

2 Implementing Matlab Code

To continue the investigation, both Euler's method and RK4 method were used to implement and numerically solve the SIR model in MATLAB. The purpose of these functions are to return three different vectors, *Svec1, Rvec1,* and *Ivec1* which contain the solution curves of each of the differential equations (1-3), respectively as well as plot them against one another to model virus transmission. The respective function headers are:

```
function [Svec1,Ivec1,Rvec1,timeStep,timeElapsed]=Euler_SIR(S0,I0,R0,a,r,days,h,tol)
function [Svec1,Ivec1,Rvec1,timeStep,timeElapsed]=RK4_SIR(S0,I0,R0,a,r,days,h,tol)
```

The input parameters for each function are identical. *S0, I0* and *R0* are the initial conditions for the susceptible, infective, and removed populations, respectively. The parameter *a* is the removal rate of the virus while *r* is its growth rate. Both functions operate over time steps measured in days, therefore the *days* parameter allows the user to input the amount of days they want to model the virus transmission for. *h* is the time step the user wants to begin with, *tol* is a tolerance level which determines whether or not a suitable termination criteria has been reached and *timeElapsed* is the time in seconds the function took to evaluate the solution. More on these three parameters and the fourth return variable

timeStep in section 2.3. Both methods will be used to solve the initial value problem:

$$\begin{cases} S' = -rIS \\ I' = rIS - aI \\ R' = aI \\ S(0) = S_0 \\ I(0) = I_0 \\ R(0) = R_0 \end{cases} \tag{13}$$

Note that in this case, R_0 refers to the initial removed population, not the basic reproductive ratio.

2.1 Euler's Method

In order to use Euler's method to approximate a solution of the SIR model, the differential equations (1-3) must be fitted inside of Euler's formula. This is a fairly simple process since all of them are first-order ODE's and already arranged to match the form needed shown in the first of equations (9). The SIR ODE's were fit to Euler's formula which yielded the equations

$$\begin{aligned} S_{i+1} &= S_i + h(-rI_iS_i) \\ I_{i+1} &= I_i + h(rI_iS_i - aI_i) \\ R_{i+1} &= R_i + h(aI_i) \end{aligned} \tag{14}$$

Firstly, the code begins by adding the initial conditions entered by the user into the output vectors

```
Svec1=[S0];
Ivec1=[I0];
Rvec1=[R0];
```

To begin translating equations (14) into MATLAB code, a *for* loop was created to loop from 1 to days-1, for a total of days-1 iterations. This is due to the fact the function defines the values of day 1 as the initial conditions. The loop then executes as

```
Snew = S-h*r*I*S;
Inew = I+h*(r*I*S-a*I);
Rnew = R+h*(a*I);
Svec1 = [Svec1; Snew];
Ivec1 = [Ivec1; Inew];
Rvec1 = [Rvec1; Rnew];
S = Snew;
I = Inew;
R = Rnew;
```

The first three lines are merely equations (14) translated into code. The following three lines update the output vectors with the new values for S, I, and R after every time step. The resulting vectors *Svec1*, *Ivec1*, and *Rvec1* contain the numerical solutions to the ODE's.

2.2 4th Order Runge-Kutta Method

To begin designing a Runge-Kutta solution to the SIR model, the same *for* loop was created as used in section 2.1. The declarations of the resulting vectors and updating of S, I, and R values within the for loop was also identical to that described in section 2.1 for Euler's method. Firstly, the k-values shown in equation (12) needed to be declared. The SIR equations are autonomous since they do not depend on time. A complication arose due to the fact that the SIR model is a system of 3 differential equations. It was already known that the k-values are interdependent when dealing with a single differential equation. These interdependencies needed to be translated for a system of 3 differential equations. It was found that, when dealing with a system of ODE's, each dependent variable S, I, and R, must be multiplied by the k-values preceding it specific to its value. Therefore, for 3 dependent variables, one needs 3 sets of k-values 1 through 4 for a total of 12 k-values. Since, for example, k_1 is called in k_2, the k_1's of each variable were declared first, followed by the k_2's and so on [5]. The k-values code for the susceptible population is shown below as an example, for the full code refer to Listing 2 in the Code Appendix.

```
k1S = -r*S*I;
k2S = -r*(S+k1S*h/2)*(I+k1I*h/2);
k3S = -r*(S+k2S*h/2)*(I+k2I*h/2);
k4S = -r*(S+k3S*h)*(I+k3I*h);
```

Next, equation (11) was implemented once for each dependent variable.

```
Snew = S+(h/6)*(k1S+2*k2S+2*k3S+k4S);
Inew = I+(h/6)*(k1I+2*k2I+2*k3I+k4I);
Rnew = R+(h/6)*(k1R+2*k2R+2*k3R+k4R);
```

With this loop, one can obtain the numerical solution to the IVP in equation (13).

2.3 Termination Criteria

The remaining problem is the determination of some termination criteria. It was decided that both the Euler and RK4 iterations should terminate when the error in the solution has fallen below a certain threshold. The code needed to engineer this termination criteria was identical for both *Euler_SIR* and *RK4_SIR*. Since there are no exact analytical solutions for the SIR equations, there are no exact solutions in which to compare those approximated. Therefore the exact error can not be determined. Instead an approximation of error needed to be found. To do this, the idea was to create a secondary *for* loop.

To begin, the first for loop, which is shown in section 2.1 for Euler's and 2.2 for RK4, would use parameter h as the time step. The second for loop would then perform the same analysis and derive a similar solution for a time step of $h/2$. Each loop would produce 3 solution vectors. The final points of these solution vectors would be compared. If they each agree within an error of a set tolerance, the code terminates and the values from the first loop are returned to the user. Should the errors be too large, the code loops once more, this time the first *for* loop uses a time step of $h/2$ and the second loop a time step of $h/3$. This continues until the difference between the final value of a vector and the final value of a vector with a smaller time step is less than the set tolerance.

To translate this into code, firstly a time step counter was set to 1. Next, the errors in S, I, and R are set to infinity so that the following while loop will execute to matter the tolerance level set by the user.

```
i = 1;
sError = inf;
iError = inf;
rError = inf;
while (sError>tol) || (iError>tol) || (rError>tol)
...
end
```

The while loop shown above contains the two *for* loops each ranging from 1 to (days-1)*i. Since i determines how many times the initial time step is divided. Furthermore, the time step counter i is incremented by one between the two *for* loops. Within each *for* loop, every instance of the time step h was divided by the counter i. Following the two *for* loops, before the end of the while loop, the following code determines the error between each for loop's result.

```
sError = abs(Svec2(length(Svec2),1)-Svec1(length(Svec1),1));
iError = abs(Ivec2(length(Ivec2),1)-Ivec1(length(Ivec1),1));
rError = abs(Rvec2(length(Rvec2),1)-Rvec1(length(Rvec1),1));
```

To continue, the tolerance's effect on execution time was also observed. In order to measure the execution time, the following code was implemented which enclosed the loops in each function.

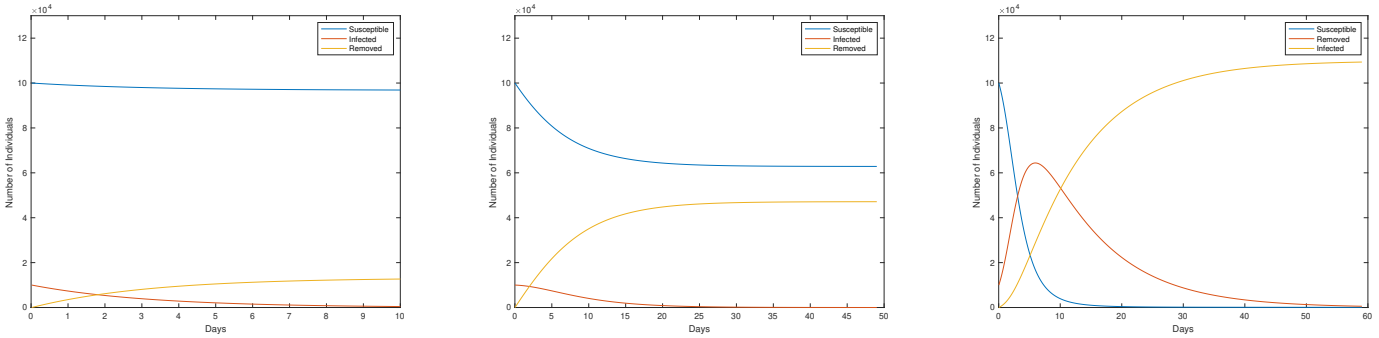
```
tic;
...
timeElapsed = toc;
```

For same input parameters, it was found that the RK4 implementation was approximately 630 times faster than Euler's at obtaining a solution within the same tolerance level. Furthermore, a decrease in tolerance level in RK4 by a factor of 10 approximately doubled the execution time. Interestingly, a decrease in tolerance level in Euler's method by a factor of 10 increases the execution time by approximately a factor of 10. Since the *while* loop and *for* loops of each method are identical, the culprit behind this vastly different execution time can not be the time complexity of the code, but the convergence of the two methods. This perfectly represents the superiority of RK4 over Euler's method. Convergence is defined as the rate at which the error of the approximation goes to zero as step size h goes to zero [3]. Since RK4 is a fourth order solver, every time the step size is halved, the error drops by approximately 2^4 while the error in Euler's would only drop by 2^1. This is also evident by the *timeStep* parameters returned by each function. For example, a tolerance level in which RK4 needs a *timeStep* division of 2 to achieve, Euler's needs 43.

3 Implementing SIR Model

3.1 Predicting Whether Disease Will Spread

As was explained in section 1.1, an outbreak will grow - that is, the rate of infection will be faster than the rate of recovery - if $R_0 > 1$. To visualize this, take a hypothetical contagious disease with a 50% recovery rate in a population of 100,000, with 10,000 individuals initially infected. These values are exaggerated compared to usual outbreaks in order to demonstrate this phenomenon. Using the RK4 Method, models were produced for R_0 values of 0.25, 1, and 7.5, corresponding to growth rates of $r = 1.25 \cdot 10^{-6}$, $r = 5 \cdot 10^{-6}$, and $r = 7.5 \cdot 10^{-6}$, respectively. The resulting numerical solutions representing the susceptible, infected and recovered populations are plotted against time in figure 1.



(a) SIR model with $R_0 = 0.25$, over 10 days (b) SIR model with $R_0 = 1$, over 50 days (c) SIR model with $R_0 = 7.5$, over 60 days

Figure 1: Various RK4 models of the susceptible, infected and recovered populations with varying basic reproductive ratios. Note that the full-sized plots can be found in the Plot Appendix.

In Figure 1a, as $R_0 < 1$, the number of infected individuals immediately decreases from I_0, falling to zero after only 10 days. Similarly, Figure 1b demonstrates the steady-state equilibrium for $R_0 = 1$. In this case, the number of infected decreases much more gradually, falling to zero after 25 days. Finally, in the case where $R_0 = 1.5$, the number of infected quickly peaks, and does not fall to zero until there are no more susceptible individuals to become infected. It is thus clear that the lower the basic reproductive ratio, the lower the impact of the outbreak will be.

Comparing the plots in figure 1 can also produce insights into the question of how many people will catch the disease. For any $R_0 > 1$, this model guarantees that every member of the population will catch the disease over the course of the outbreak. This makes sense; it was determined above that in this case, the susceptible population will fall to zero *before* the infected population. In the case that $R_0 \leq 1$, the final removed population declines as R_0 is reduced. This is evident by comparing the final removed populations of 12688 and 47138 for figure 1a and 1b, respectively. Equation (8) also demonstrates this as well; R_{end} will decrease as R_0 decreases for $R_0 \leq 1$. This re-enforces the previous statement that minimizing the basic reproductive ratio can help to lower the damage done by a disease outbreak.

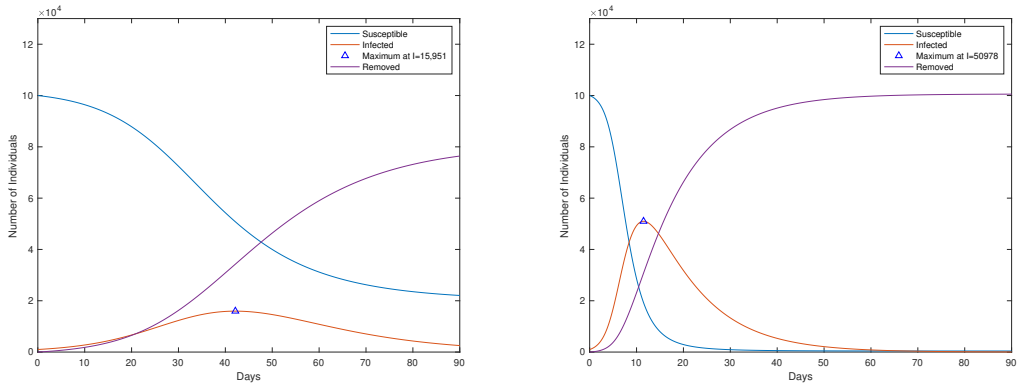
3.2 Predicting Maximum Number of Infections

In order to prepare health care systems to combat an outbreak, it is necessary predict the increase in capacity needed at the peak. This is done by predicting what will be the maximum number of infections at one time. As stated in equation (7), $I_{max} = I_0 + S_0 - \frac{1}{q}(1 + \ln(qS_0))$, where $q = \frac{r}{a}$ is the *contact ratio*. From the results of section 3.1, it is clear that this question is only relevant when $R_0 > 1$ as otherwise, $I_{max} = I_0$. To observe the effects of a varying contact ratio, once again consider a disease in an initial population of 100,000 with 1000 individuals initially infected. Compare the cases where $q = 2 \cdot 10^{-5}$ (corresponding to $R_0 = 2$) and $q = 6 \cdot 10^{-5}$ (corresponding to $R_0 = 6$). The exact values for the predicted maximum number of simultaneous cases would then be:

$$I_{max} = 1000 + 100,000 - \frac{1 + \ln(2 \cdot 10^{-5} \cdot 100,000)}{2 \cdot 10^{-5}} = 16,343$$

$$I_{max} = 1000 + 100,000 - \frac{1 + \ln(6 \cdot 10^{-5} \cdot 100,000)}{6 \cdot 10^{-5}} = 54,470$$

To visualize this, the RK4 Method was used to plot the approximated solutions for these two contact ratios:



(a) SIR model with $q = 2 \cdot 10^{-5}$

(b) SIR model with $q = 6 \cdot 10^{-5}$

Figure 2: Various RK4 models of the susceptible, infected and recovered populations with varying contact ratios, illustrating the changes in maximum simultaneous infectives. Note that the full-sized plots can be found in the Plot Appendix.

Figure 2a, representing $q = 2 \cdot 10^{-5}$, displayed an $I_{max} = 15,951$. This has a relative error of 2.4% when compared to the exact solution of 16,343. Figure 2b illustrated an $I_{max} = 50,978$ for $q = 6 \cdot 10^{-5}$, producing a relative error of 6.4% compared to the exact solution of 54,470. These demonstrate that in order to lower the maximum number of simultaneous cases and therefore reduce the burden on the health care system, it is necessary to minimize the contact ratio.

3.3 Comparing Euler's and RK4 using the 2014 Ebola Outbreak

In order to compare Euler's and RK4 ODE solvers, the 2014 Ebola outbreak was modelled using both and the results were compared against one another. The initial conditions used were $S_0 = 4293265$, $I_0 = 391$, $R_0 = 344$ with a growth rate of $r = 1.63 \times 10^{-7}$ and a removal rate of $a = 0.09$ as found for the Ebola outbreak [5]. The tolerance was set to 0.01 and the outbreak was modelled for 100 days. As one can observe in figure 3 the results from RK4 and Euler's method are almost identical. This is due to the manner in which the termination criteria of the methods was implemented. As previously explained, due to the tolerance parameter, for a given tolerance, both methods will

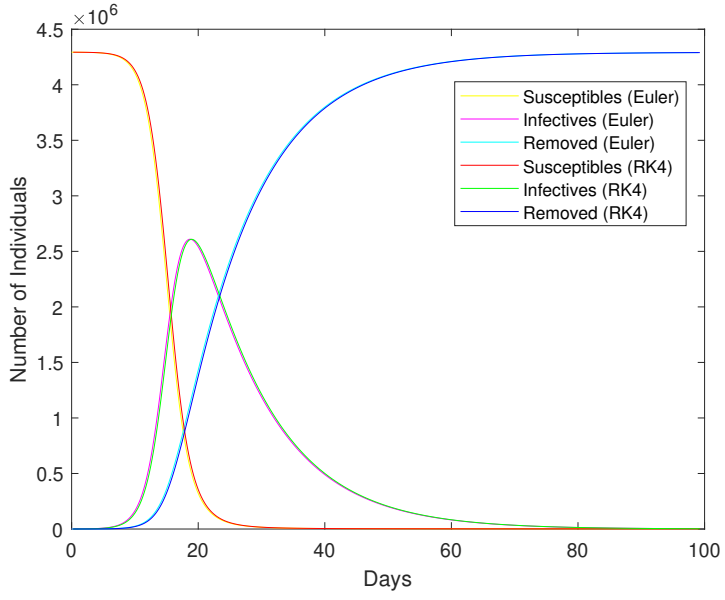


Figure 3: Euler's and RK4 ODE solver methods compared using 2014 Ebola Outbreak data.

produce an output of approximately the same accuracy. However, the *timeStep* parameter is indicative of Euler's poor accuracy. To produce the results in figure 3, Euler's method needed to use a step size of 1/431, while RK4 needed a step size of 1/4. Due to the way in which the code was engineered, the user may determine the accuracy to which the methods should adhere to. However, the user must take into account that due to Euler's poor convergence, for the same tolerance level as RK4, it takes approximately 630 times longer to execute.

4 Conclusions

Despite its simplicity, the SIR Model is a valuable tool for predicting and drawing insights into infectious disease outbreaks. By constructing a system of coupled differential equations representing the members of a closed population that are susceptible to, infected by, or recovered/have died from a disease, simple models of the outbreak can be produced. The behaviour of this system is largely governed by the rates of transmission and recovery, both specific to the disease and the population. These can combine to form two additional parameters: the *contact ratio* q and *basic reproductive ratio* R_0, which represent the average number of close contacts of and secondary infections caused by a single infected individual, respectively.

Firstly, to be able to draw any sort of insight from the SIR equation, a numerical solution needed to be engineered due to their lack of analytical solutions. Both Euler's method and RK4 were used to provide approximate solutions. Due to the termination criteria of the two methods, both yield solutions with the same accuracy if the same tolerance level is input. The areas in which the varying convergences were observed were the execution time and step size needed for each program to produce a solution of equivalent precision. All in all, it was found that RK4 was approximately 630 times faster at execution than Euler's method which was as expected since RK4 has order 4 convergence while Euler's has merely order 1. Both models produced solutions of any specified precision. Once engineered, the programs allowed for the manipulation of various virus parameters in order to observe the mechanics of virus transmission.

There are several key insights that can be drawn from the solutions to an SIR Model that are essential to predicting the impacts of and preparing for a disease outbreak. Firstly, whether or not the outbreak will grow from the initial infected population determines whether there is risk of an epidemic. It was found that the infected population would grow for $R_0 > 1$. It was also evident that for $R_0 \leq 1$, a lower R_0 value caused the outbreak to fall to zero sooner, leading to a lower fraction of the population catching the disease. Finally, it is critical to predict and minimize the maximum number of simultaneous infections in order to ensure the capacity of the health-care system meets the increased demand. It was found that the lower the value of q, the less infections would occur at any one time.

These conclusions are particularly interesting to discuss in the context of the current global Covid-19 Pandemic. Current estimates of the global average basic reproductive ratio for Covid-19 is $R_0 = 4.5$ when no deliberate interventions are taken [6]. This means that every infected individual will subsequently infect, on average, 4.5 others. At this rate, the virus would immediately spread, quickly infecting the entire population. It is thus critical to take interventions to lower the R_0. Additionally, there have been widespread efforts to "*flatten the curve*", which specifically references decreasing the maximum number of infections at any given time in order to allow the health-care system to manage. Section 3.3 demonstrated that I_{max} increases as does q. These explain why measures such as social distancing, hand-washing, face-masks, and surface disinfecting are recommended; these decrease the rate of transmission, thereby decreasing R_0 and q and slowing the spread of the virus. This also relates to the value of the vaccines that are currently being distributed; vaccines essentially transfer individuals from the susceptible populations to the recovered populations without requiring them to become infected. As $R_0 \propto S_0$, this causes R_0 to decline, thereby slowing the spread of the virus as well.

This investigation has highlighted the real-world importance of numerical ODE solvers such as Euler's Method and RK4. In the case of infectious diseases, these allow us to calculate approximate solutions to the SIR system of equations, thereby producing highly valuable outbreak models. While actual frameworks used to model real diseases are generally more complicated than the simple SIR model described in this report - such as using a varying R_0 to account for changes in human behaviour or government restrictions - the underlying principles are the same. These models, that could only be constructed using numerical ODE solvers, allow individuals, health-care systems and governments to better understand and prepare for outbreaks, ultimately preventing countless deaths.

References

[1] Maths, T. R. (Director). (2020, March 18). Oxford mathematician EXPLAINS SIR disease model FOR Covid-19 (coronavirus) [Video file]. Retrieved March 20, 2021, from https://www.youtube.com/watch?v=NKMHhm2Zbkw

[2] Cooper, I., Mondal, A., Antonopoulos, C. G. (2020). A SIR model assumption for the spread of COVID-19 in different communities. Chaos, solitons, and fractals, 139, 110057. https://doi.org/10.1016/j.chaos.2020.110057

[3] Sauer, T. (2019). Numerical analysis. In Numerical analysis (3rd ed., pp. 294-332). Hoboken, New Jersey: Pearson.

[4] Cheever, E. (n.d.). Fourth order Runge-Kutta. Retrieved March 20, 2021, from https://lpsa.swarthmore.edu/NumInt/NumIntFourth.html

[5] HOSSAIN, T., MIAH, M., amp; HOSSAIN, J. (2017). A numerical study to predict the evolution of yellow Fever diseases using fourth order RUNGE-KUTTA METHOD. International Journal of Scientific amp; Engineering Research, 7(10), 1040-1046. doi:10.14299/ijser.2017.10.004

[6] Katul, G. G., Mrad, A., Bonetti, S., Manoli, G., amp; Parolari, A. J. (2020). Global convergence of COVID-19 basic reproduction number and estimation From EARLY-TIME sir dynamics. PLOS ONE, 15(9). doi:10.1371/journal.pone.0239800

5 Code Appendix

Listing 1: SIR Virus Transmission Model implemented using 1st Order ODE solver known as Euler's method. The comments are removed for the sake of a concise code appendix.

```
function [Svec1,Ivec1,Rvec1,timeStep,timeElapsed]=Euler_SIR(S0,I0,R0,a,r,days,h,tol)
    tic;
    i = 1;
    sError = inf;
    iError = inf;
    rError = inf;
    while (sError>tol) || (iError>tol) || (rError>tol)
        Svec1=[S0];
        Ivec1=[I0];
        Rvec1=[R0];
        Svec2=[S0];
        Ivec2=[I0];
        Rvec2=[R0];
        S = S0;
        I = I0;
        R = R0;
        for j=1:(days-1)*i
            Snew = S-(h/i)*r*I*S;
            Inew = I+(h/i)*(r*I*S-a*I);
            Rnew = R+(h/i)*(a*I);
            Svec1 = [Svec1; Snew];
            Ivec1 = [Ivec1; Inew];
            Rvec1 = [Rvec1; Rnew];
            S = Snew;
            I = Inew;
            R = Rnew;
        end
        i = i+1;
        S = S0;
        I = I0;
        R = R0;
        for k=1:(days-1)*i
           Snew = S-(h/i)*r*I*S;
           Inew = I+(h/i)*(r*I*S-a*I);
           Rnew = R+(h/i)*(a*I);
           Svec2 = [Svec2; Snew];
           Ivec2 = [Ivec2; Inew];
           Rvec2 = [Rvec2; Rnew];
           S = Snew;
           I = Inew;
           R = Rnew;
        end
        sError = abs(Svec2(length(Svec2),1)-Svec1(length(Svec1),1));
        iError = abs(Ivec2(length(Ivec2),1)-Ivec1(length(Ivec1),1));
        rError = abs(Rvec2(length(Rvec2),1)-Rvec1(length(Rvec1),1));

    end
    timeElapsed = toc;
    timeStep = i-1;
    x=linspace(1,length(Svec1),length(Svec1))/timeStep;
    plot(x,Svec1);
    hold on
    plot(x,Ivec1);
```

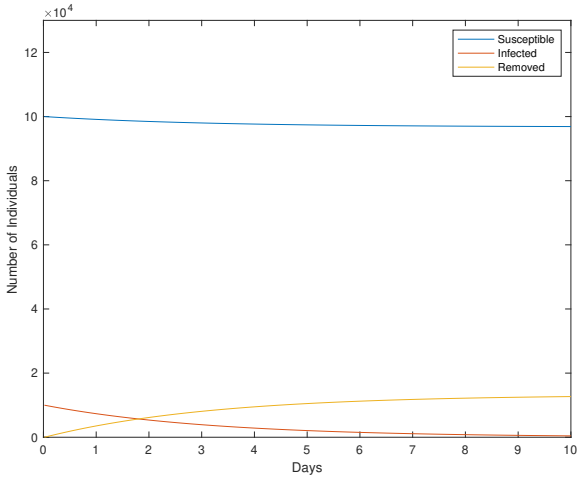
```
    plot(x,Rvec1);
    hold off
end
```

Listing 2: Fourth Order Runge-Kutta Method implementation of SIR virus transmission model. The comments are removed for the sake of a concise code appendix.

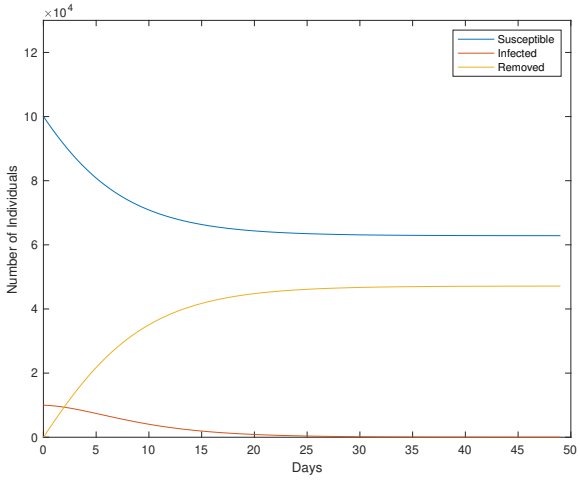
```
function [Svec1,Ivec1,Rvec1,timeStep,timeElapsed]=RK4_SIR(S0,I0,R0,a,r,days,h,tol)
    tic;
    i = 1;
    sError = inf;
    iError = inf;
    rError = inf;
    while (sError>tol) || (iError>tol) || (rError>tol)
        Svec1=[S0];
        Ivec1=[I0];
        Rvec1=[R0];
        Svec2=[S0];
      Ivec2=[I0];
        Rvec2=[R0];
        S = S0;
        I = I0;
        R = R0;
        for j=1:(days-1)*i
            k1S = -r*S*I;
            k1I = r*S*I-a*I;
            k1R = a*I;
            k2S = -r*(S+k1S*h/(2*i))*(I+k1I*h/(2*i));
            k2I = r*(S+k1S*h/(2*i))*(I+k1I*h/(2*i))-a*(I+k1I*h/(2*i));
            k2R = a*(I+k1I*h/(2*i));
            k3S = -r*(S+k2S*h/(2*i))*(I+k2I*h/(2*i));
            k3I = r*(S+k2S*h/(2*i))*(I+k2I*h/(2*i))-a*(I+k2I*h/(2*i));
            k3R = a*(I+k2I*h/(2*i));
            k4S = -r*(S+k3S*h/i)*(I+k3I*h/i);
            k4I = r*(S+k3S*h/i)*(I+k3I*h/i)-a*(I+k3I*h/i);
            k4R = a*(I+k3I*h/i);
            Snew = S+((h/i)/6)*(k1S+2*k2S+2*k3S+k4S);
            Inew = I+((h/i)/6)*(k1I+2*k2I+2*k3I+k4I);
            Rnew = R+((h/i)/6)*(k1R+2*k2R+2*k3R+k4R);
            Svec1 = [Svec1; Snew];
            Ivec1 = [Ivec1; Inew];
            Rvec1 = [Rvec1; Rnew];
            S = Snew;
            I = Inew;
            R = Rnew;
        end
        i = i+1;
        S = S0;
        I = I0;
        R = R0;
        for k=1:(days-1)*i
           k1S = -r*S*I;
           k1I = r*S*I-a*I;
           k1R = a*I;
           k2S = -r*(S+k1S*h/(2*i))*(I+k1I*h/(2*i));
           k2I = r*(S+k1S*h/(2*i))*(I+k1I*h/(2*i))-a*(I+k1I*h/(2*i));
           k2R = a*(I+k1I*h/(2*i));
           k3S = -r*(S+k2S*h/(2*i))*(I+k2I*h/(2*i));
           k3I = r*(S+k2S*h/(2*i))*(I+k2I*h/(2*i))-a*(I+k2I*h/(2*i));
           k3R = a*(I+k2I*h/(2*i));
           k4S = -r*(S+k3S*h/i)*(I+k3I*h/i);
           k4I = r*(S+k3S*h/i)*(I+k3I*h/i)-a*(I+k3I*h/i);
           k4R = a*(I+k3I*h/i);
           Snew = S+((h/i)/6)*(k1S+2*k2S+2*k3S+k4S);
           Inew = I+((h/i)/6)*(k1I+2*k2I+2*k3I+k4I);
           Rnew = R+((h/i)/6)*(k1R+2*k2R+2*k3R+k4R);
           Svec2 = [Svec2; Snew];
```

```
            Ivec2 = [Ivec2; Inew];
            Rvec2 = [Rvec2; Rnew];
            S = Snew;
            I = Inew;
            R = Rnew;
        end
        sError = abs(Svec2(length(Svec2),1)-Svec1(length(Svec1),1));
        iError = abs(Ivec2(length(Ivec2),1)-Ivec1(length(Ivec1),1));
        rError = abs(Rvec2(length(Rvec2),1)-Rvec1(length(Rvec1),1));
    end
    timeElapsed = toc;
    timeStep = i-1;
    x=linspace(1,length(Svec1),length(Svec1))/timeStep;
    plot(x,Svec1);
    hold on
    plot(x,Ivec1);
    plot(x,Rvec1);
    hold off
end
```

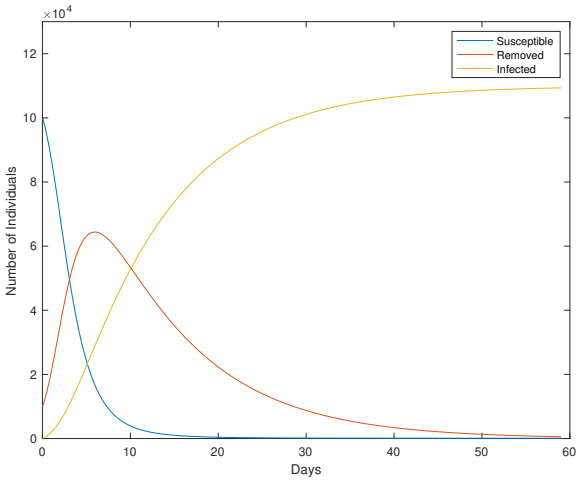
6 Plot Appendix



(a) SIR model with $R_0 = 0.25$, over 10 days

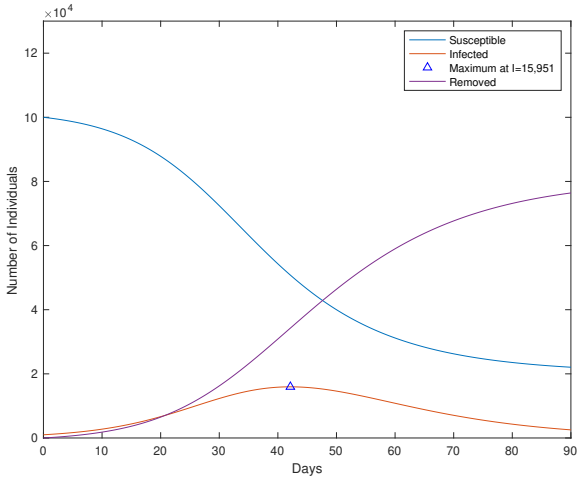


(b) SIR model with $R_0 = 1$, over 50 days

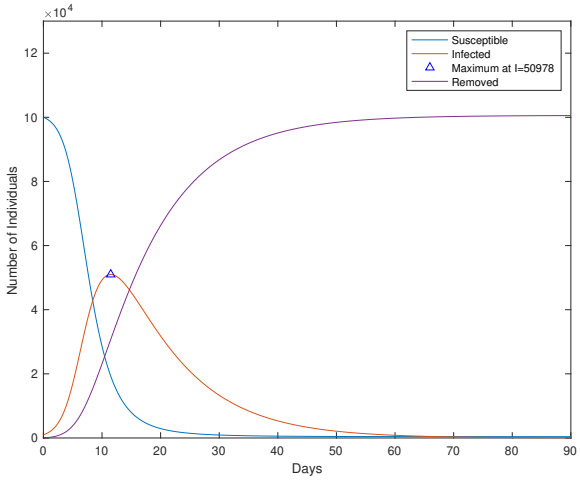


(c) SIR model with $R_0 = 7.5$, over 60 days

Figure 1: Various RK4 models of the susceptible, infected and recovered populations with varying basic reproductive ratios.



(a) SIR model with $q = 2 \cdot 10^{-5}$



(b) SIR model with $q = 6 \cdot 10^{-5}$

Figure 2: Various RK4 models of the susceptible, infected and recovered populations with varying contact ratios, illustrating the changes in maximum simultaneous infectives.

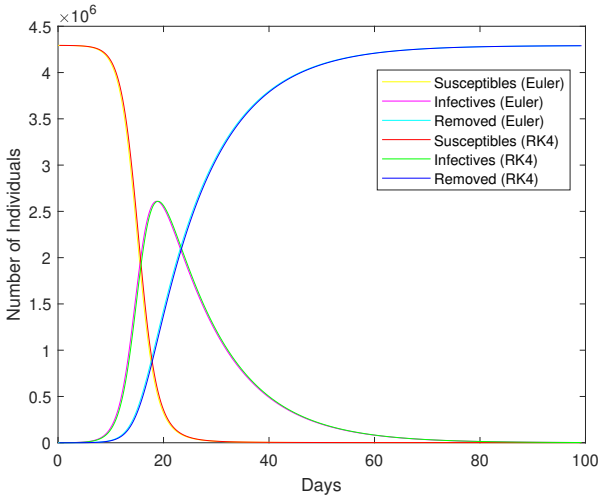


Figure 3: Euler's and RK4 ODE solver methods compared using 2014 Ebola Outbreak data.