Modeling the Spread of Infectious Diseases: A practical exercise with MATLAB for COVID-19

Jesse Gabriel
jessegabriel11@gmail.com

16 June 2021

Introduction

The spread of infectious diseases such as flu have been modeled mathematically. This has been true for COVID-19 since its declaration as a pandemic by WHO in late 2019. Various methods of analysis and mathematical models that describe the spread of COVID-19 have been developed by different researchers. Here we consider an example exercise with MATLAB. The raw data can be obtained from WHO or other sites (e.g. here)

Derivation of the Mathematical Model

To create a suitable model, we divide the population into several groups: (1) susceptible, S, (2) infected, I, (3) recovered, R, and (4) deceased, D. The idea is that those in the group S get infected and move on into group I, then further move into either group R or D. The population in each of the groups are changing (with respect to time) as subjects are moving from one group to the other. This change is represented by a differential equation. Thus, for the four groups, we have a set of first-order ordinary differential equations as given below:

$$\left.\begin{aligned} \frac{dS}{dt} &= -f(t,S) \\ \frac{dI}{dt} &= f(t,S) - g(t,I) - h(t,I) \\ \frac{dR}{dt} &= g(t,I) \\ \frac{dD}{dt} &= h(t,I) \end{aligned}\right\} \tag{1}$$

To simplify our problem, we can assume that the rate of change in each of the groups is linear or steady. Thus, we have four (4) rate constants represented by C, and our set of equations is now:

$$\left.\begin{aligned} \frac{dS}{dt} &= -C_1 S \\ \frac{dI}{dt} &= C_1 S - C_2 I - C_3 I \\ \frac{dR}{dt} &= C_2 I \\ \frac{dD}{dt} &= C_3 I \end{aligned}\right\} \tag{2}$$

where $C_1 = infection\ rate, C_2 = recovery\ rate, C_3 = death\ rate.$

Solution

Runge-Kutta 4^{th} Order

Various numerical methods are available to solve such initial value problems including Euler's and Runge-Kutta's methods. Here we use the Runge-Kutta 4^{th} order as it provides relatively more reliable results. For the differential equation $y = f(t, y)$ where $y(t_0) = y_0$ the Runge-Kutta of fourth-order method (RK4) method is defined using the following recursion formula:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)\,h \tag{3}$$

where:

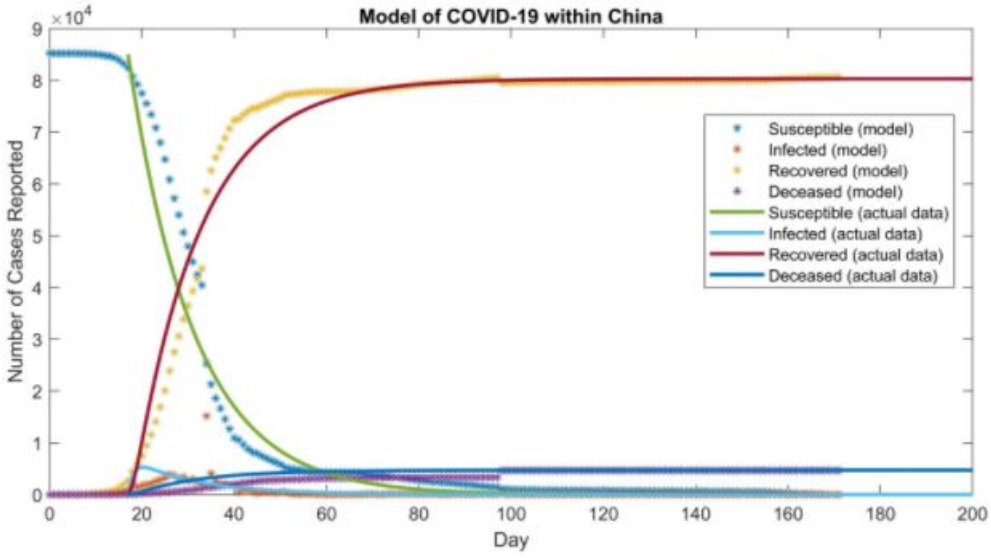
$$\left.\begin{aligned} &\frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) = \text{weighted average of slope} \\ &k_1 = f(t_n, y_n) \\ &k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \\ &k_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2) \\ &k_4 = f(t_n + \frac{h}{2}, y_n + hk_3) \\ &h = \text{step size} \end{aligned}\right\} \tag{4}$$

Initial Condition

Let's look at China as an example. Towards the end of October last year (2020), the number of people infected was around 85000. In order to make our model fit that data, our initial condition (day zero) would be $S(0) = 85000, I(0) = 0; R(0) = 0; D(0) = 0$. This will be the same approach for other countries.

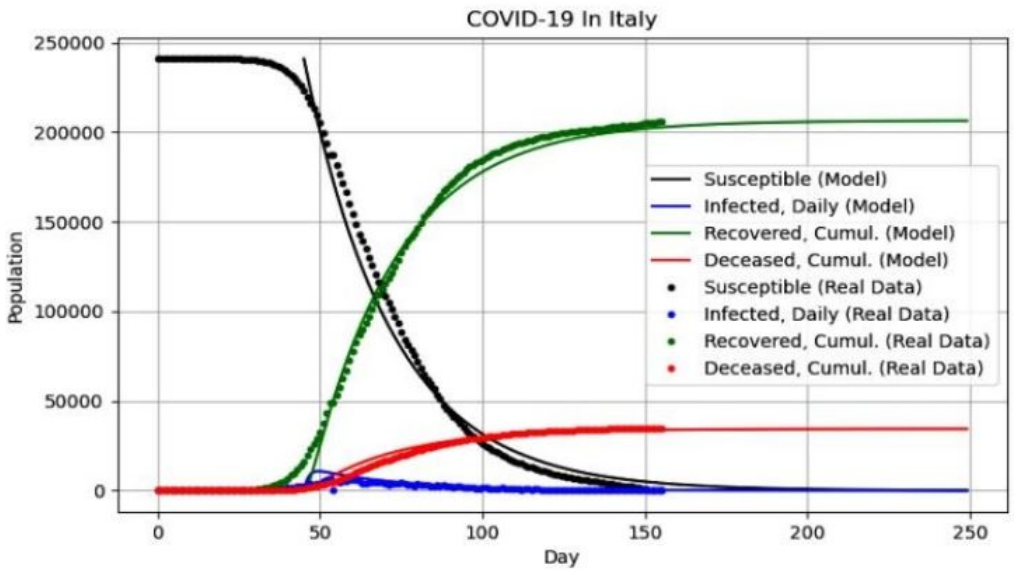
Results

1. China:



 (originally posted on LinkedIn: https://www.linkedin.com/pulse/modeling-spread-infectious-disease-covid-19-jesse-gabriel

2. Italy (Python code).



Discussion & Conclusion

The results from the model (predictions) satisfactorily fits the actual data. It is quite difficult, but with rigorous research, it is possible to mathematically model the spread of infectious diseases such as COVID-19. The key issue is in determining the susceptible population and the rate constants. The main way to determine these would be to investigate factors such as health and hygiene conditions, people mobility, the nature of the virus to spread in a particular country or climatic conditions, and many other factors that health professionals could provide. These would help in deducing the right values for the susceptible population and the rate constants, which would then be used to simulate the spread of the virus in a certain country.

Appendix

A software package and source code.

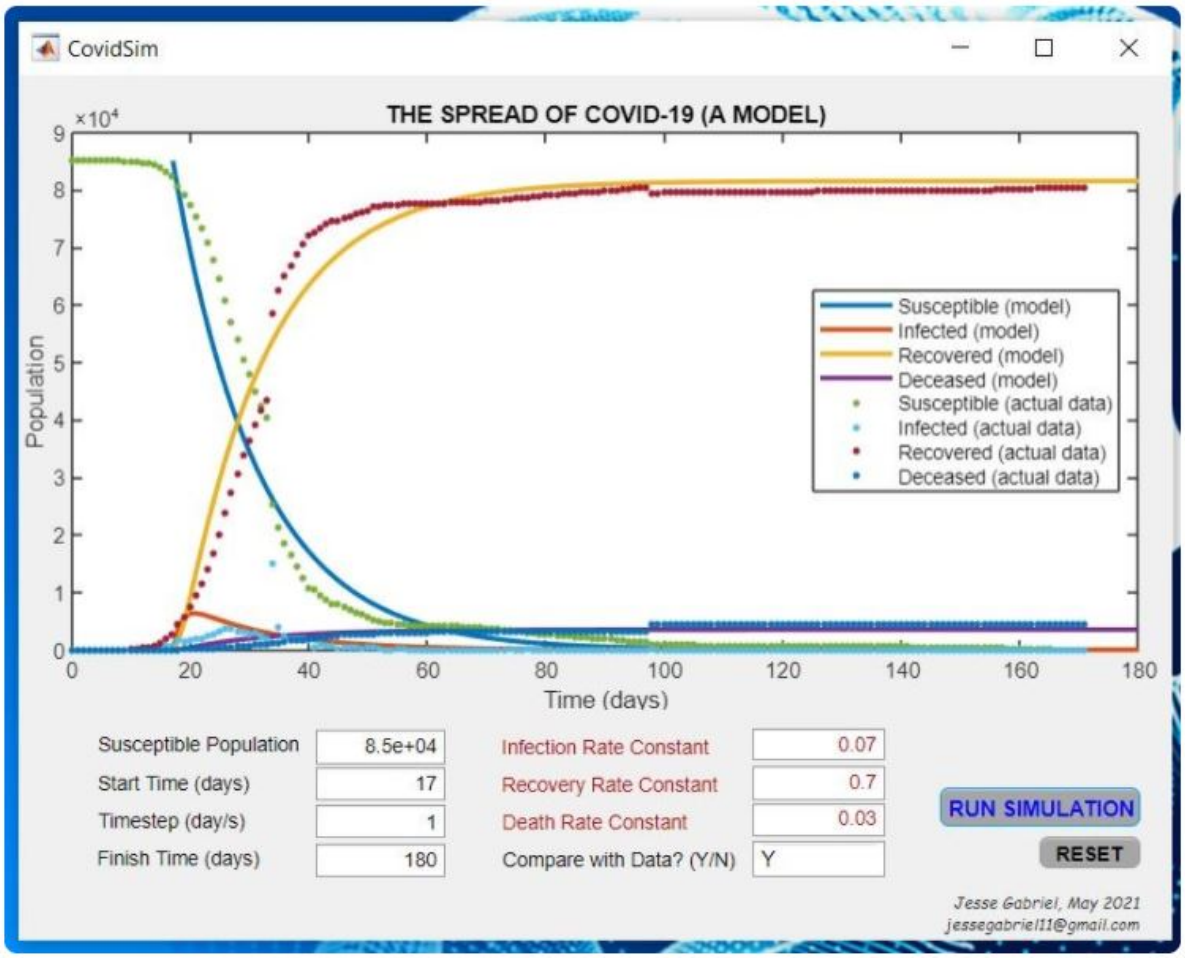


Figure 1. Graphical user interface (GUI) of the software developed from the described model. (*NB: Source code will be provided freely upon request at jessegabriel11@gmail.com*).

MATLAB CODE FOR COVID-19

Main Script

```
close all; clear; clc;
%% Define Problem
%const.c0 = 0.05;
const.c1 = 0.07;
const.c2 = 0.85;
const.c3 = 0.05;
country = input(' ENTER COUNTRY:  ');
susceptible = input(' ENTER SUSCEPTIBLE POPULATION:  ');
t0 = input(' ENTER START TIME (DAY):  ');
dt = input(' ENTER TIMESTEP (DAYS):  ');
tf = input(' ENTER FINAL TIME (DAYS):  ');
f = @(t,V) funCOVID(V,const);
V0 = [susceptible; 0; 0; 0];
%% Solve Differential Equation using Runge-Kutta 4th Order
% t0 = 17;
% tf = 200;
% dt = 1;
[V, t] = RungeKutta4(f, V0, t0, tf, dt);
YN1 = [' Would you like to compare your model with ',
num2str(country),'''s ', 'actual data? Y / N?   ']; YN2 = input(YN1);
if YN2 == 'Y'
    %% Plot Actual Data
    CN = input(' ENTER ACTUAL DATA (EXCEL FILE):  '); %
xlsread('WHO_COVID19.xlsx','China','G2:K173');
    day = CN(:,1); Susceptible = CN(:,2); Infected = CN(:,3);
Recovered = CN(:,4); Deceased = CN(:,5);
    plot(CN(:,1),(CN(:,2:5)),'*','MarkerSize',5),  hold on
   %% Plot Simulation Results
    titl = ['Model of COVID-19 within ', num2str(country)];
    plot(t,V,'LineWidth',2), title(titl)
    xlabel('Day'), ylabel('Number of Cases Reported')
    legend('Susceptible (model)','Infected (model)','Recovered
(model)','Deceased (model)', 'Susceptible (actual data)','Infected
(actual data)','Recovered (actual data)','Deceased (actual data)')
elseif YN2 == 'N'
   %% Plot Simulation Results
    titl = ['Model of COVID-19 within ', num2str(country)];
    plot(t,V,'LineWidth',2), title(titl)
    xlabel('Day'), ylabel('Number of Cases Reported')
    legend('Susceptible (model)','Infected (model)','Recovered
(model)','Deceased (model)')
else
    disp(' ENTER ONLY Y / N !');
end
```

Function 1: Model Equations (funCOVID)

```
function all = funCOVID(V, const)
c1 = const.c1;
c2 = const.c2;
c3 = const.c3;
all = [-c1*V(1)                          % susceptible
            c1*V(1)-c2*V(2)-c3*V(2)      % infected
            c2*V(2)                      % recovered
            c3*V(2)];                    % deceased
end
```

Function 2: Numerical Solution (RungeKutta4)

```
function [V,t] = RungeKutta4(f, V0, t0, tf, dt)
        t = t0:dt:tf;
        nt = numel(t);
        nV = numel(V0);
        V= nan(nV, nt);
        V(:,1) = V0;
        % Solving Using RK-4 Method
        for k = 1:nt-1
                k1 = dt*f(t(k), V(:,k));
                k2 = dt*f(t(k)+dt/2, V(:,k)+k1/2);
                k3 = dt*f(t(k)+dt/2, V(:,k)+k2/2);
                k4 = dt*f(t(k)+dt, V(:,k)+k3);
                dV = (k1+2*k2+2*k3+k4)/6;
                V(:,k+1) = V(:,k)+dV;
        end
end
```