MTH 698: Final Project

Jennefer Maldonado

December 15, 2021

1 Introduction

In the past two years we have been faced with the COVID-19 pandemic. It has taken away millions of lives worldwide and impacted the health of many as time has passed. Differential equations are used to find the rate of change among populations. The idea of using differential equations to determine the spread of the COVID-19 pandemic is important to aid in the safety of others but also determine scenarios of public health. We could see a scenario where a large percentage of the population is getting sick, in turn raising death or recovery rates. We could also see a scenario where a large percentage of the population is not becoming infected, thus lowering the percentage of those in the hospital and lowering death rates. The best way to do this is by simulation. Writing numerical methods allows for these numbers to be updated to model what we are seeing in the real world. This allows epidemiologists and others to let the general public know what the near future could look like.

2 Model

The model we wish to study looks at multiple populations of people. The S variable represents the susceptible portion of the community. These are the group of individuals who are not infected. The infected but not tested part of the population is represented by I_0. This population can now be moved into one of two groups, the first is the infected who test positive for the virus or I_+. The second, I_-, is the infected who test negative and are false negatives. Finally, R represents the recovered part of the infected community.

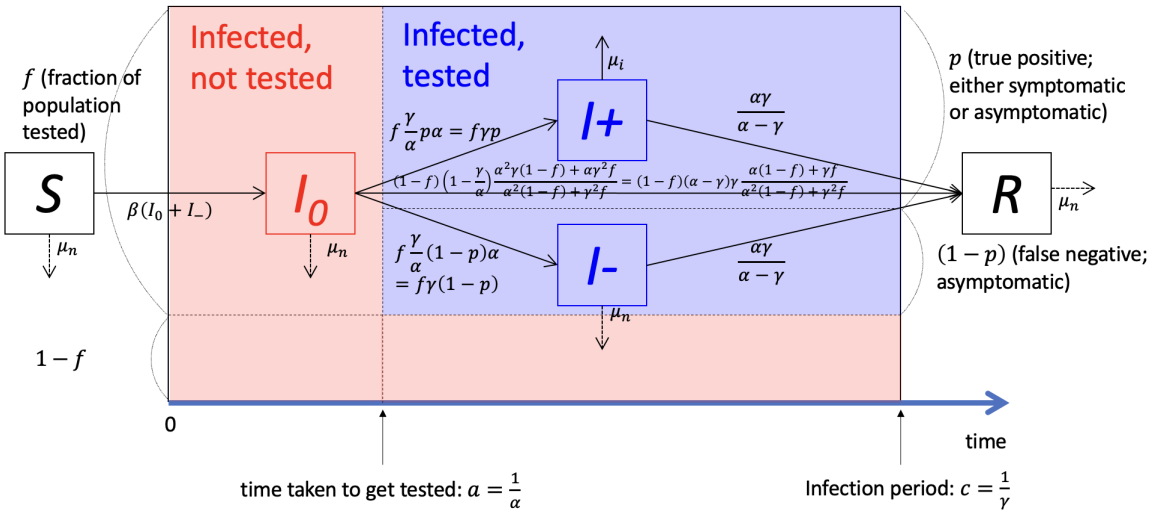

Figure 1: Model of Community Interaction

Next we can examine the portion of the population in the hospital.

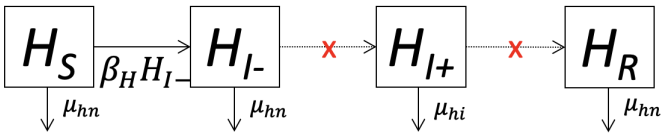

Figure 2: Model of Hospital Interaction

In this simulated world there are four kinds of people who are in the hospital. The first is the susceptible community H_S who are in the hospital for reasons unrelated to COVID-19. Similarly the portion of the population who test negative H_{I_-} and the population who are discharged H_R are unrelated to COVID-19

and are unaffected by the virus. The red x's represent that there is no flow and the different variables do not influence each other. There are two assumptions we must follow in the model. The first is that the only population infected by the disease is H_{I+} since they have tested positive. The second assumption is infected individuals who tested negative will remain negative on test results.

The two models can now be combined to see the flow between the population outside and inside the hospital.

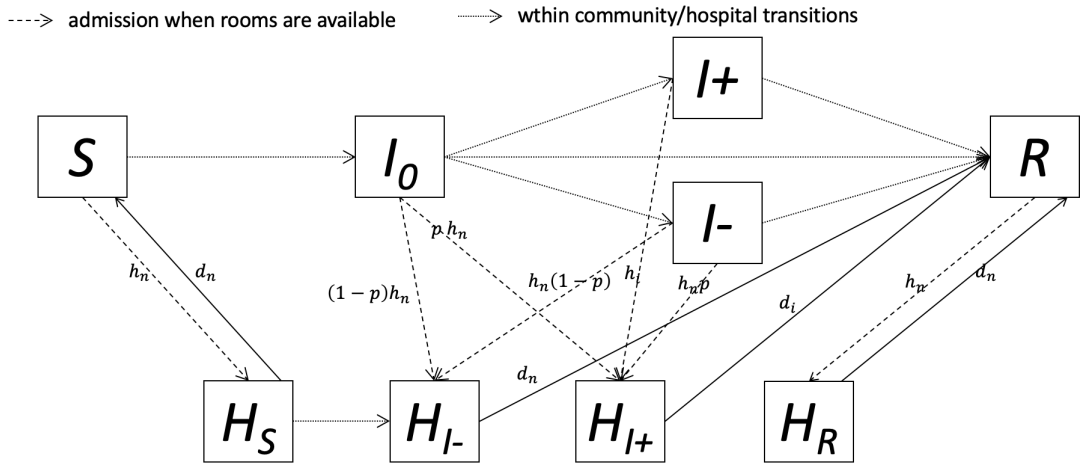

Figure 3: Model of Interactions between Community and Hospital

This model gives us a clear image of how the spread of COVID-19 leads infected individuals into the hospital and also shows how they recover and return to the susceptible population once again. This diagram does not take into account the vaccine found which has prevented hundreds of thousands of people from becoming infected.

This model can now be translated into a series of differential equations. This allows us to find the change in each variable showing the effects of infection among the community and hospital.

$$S' = -\mu S - \beta(I_0 + I_{\text{-}}) - Sh_n + H_s d_n \tag{1}$$

$$I_0' = S\beta I_{\text{-}} + I_0[-\mu_n + S\beta - (1-f)(\alpha-\gamma)\gamma\frac{\alpha(1-f)+\gamma f}{\alpha^2(1-f)+\gamma^2 f} + f\gamma(2p-1) - h_n] \tag{2}$$

$$I_+' = -I_+\mu_i + I_0 f\gamma p - I_+(h_i + \frac{\alpha\gamma}{\alpha-\gamma}) \tag{3}$$

$$I_{\text{-}}' = -I_{\text{-}}(\mu_n + h_n + \frac{\alpha\gamma}{\alpha-\gamma}) + I_0 f\gamma(1-p) \tag{4}$$

$$R' = -R(\mu_n+h_n)+I_{\text{-}}(\frac{\alpha\gamma}{\alpha-\gamma})+I_+(\frac{\alpha\gamma}{\alpha-\gamma})+I_0[(1-f)(\alpha-\gamma)\gamma\frac{\alpha(1-f)+\gamma f}{\alpha^2(1-f)+\gamma^2 f}]+H_{I_{\text{-}}}d_n+H_{I_+}d_i+H_R d_n \tag{5}$$

$$H_s' = -H_s(d_n + \mu_{h_n} + \beta_H H_{I_{\text{-}}}) + Sh_n \tag{6}$$

$$H'I_{\text{-}} = I_0(1-p)h_n - H_{I_{\text{-}}}[h_n(1-p) + d_n + \mu_{h_n}] + H_s\beta_H H_{I_{\text{-}}} \tag{7}$$

$$H_{I_+}' = -H_{I_+}(d_i + \mu_{h_i} + \epsilon) + I_+h_i + I_{\text{-}}h_n p + I_0 p h_n \tag{8}$$

$$H_R' = -H_R(d_n + \mu h_n) + Rh_n \tag{9}$$

The equations have different positive and negative coefficients. It is important to note a negative coefficient means the population is flowing out of that current state. Whereas a positive coefficient represents the population flowing into that current state. Each differential equation was created by following the direction of flows into and out of the state and paying close attention to the coefficient values that represented the size of the population flow.

2.1 Constant Values

There are also a few variables that will stay constant throughout the entire simulation. These variables were computed with real world data found online to try and find the most accurate simulation results.

Variable	Constant Value
μ_i	0.0001
μ_n	0
μ_{h_n}	0.025
μ_{h_i}	$0.025 + \epsilon$
ϵ	0.0001
f	0.3
p	0.4
γ	0.05
α	0.25
β	$0.2, 0.5, 0.7, 1$
β_H	$0.2, 0.5, 0.7, 1$
h_n	0.0001
h_i	0.01
d_n	0.2
d_i	0.05

Table 1: Table of Constant Values from Real World COVID-19 Data

2.2 Initial Conditions

The initial values of the differential equations can vary over many values depending on the impact of the disease at any current moment. I created four scenarios of what initial conditions could possibly look like.

Variable	Scenario A	Scenario B	Scenario C	Scenario D
S	$1 - 0.0010001$	$1 - 0.0010001$	$1 - 0.0010001$	$1 - 0.0010001$
I_0	0.0000001	0.000001	0.0001	0.01
I_-	0	0	0	0
I_+	0	0.0000001	0.00001	0.001
H_S	0.0001	0.0001	0.0001	0.0001
R	0	0	0	0
H_R	0.1	0.1	0.1	0.1
H_{I_-}	0	0.001	0.003	0.002
H_{I_+}	0	0.00001	0.0001	0.001

Table 2: Table of Initial Values for Simulation

Scenario A represents a time we do not know much about the population. We know the total susceptible population in all scenarios. Here we have the lowest infected rate that have not been tested. We are unsure about how many people are going to test negative or positive and there is a steady population of people in the hospital as there was before the pandemic. The second scenario, Scenario B, now adds an idea of how many people we expect to test positive. It is a small number of the overall infected population but it is an increase from 0. Also now there is an assumption that people are already in the hospital for COVID-19 related symptoms as well as those who test negative are already admitted as well. The the

3 Methods

3.1 Euler's Method

Euler's method is a first order numerical method for solving ordinary differential equations. It also assumes the equations are explicit.

$$y_{n+1} = y_n + hf(t_n, y_n) \tag{10}$$

The method is defined as follows where $y' = f(t, y(t))$. For this project each equation has Euler's method applied to it and the solutions are stored in a vector.

4 Analysis

In figures 4,5,6, and 7 are the results of 5 out of the 9 equations of our model. I chose to display 5 of them for a clear picture of what is happening solely with those diagnosed with COVID-19. Scenario A is the top left graph, Scenario B is the top right, Scenario C is the bottom left and Scenario D is the bottom right.

The first set of graphs below in Figure 4 represent the four scenario conditions with $\beta = 0.2$. Since zero is the initial value for the populations of infected and tested individuals, the change in infected populations is not present. This may mean β is the issue or the initial conditions are not representative of the system. For the later two graphs with the updated initial conditions the infected populations grow and describe the infected community against the recovered community.

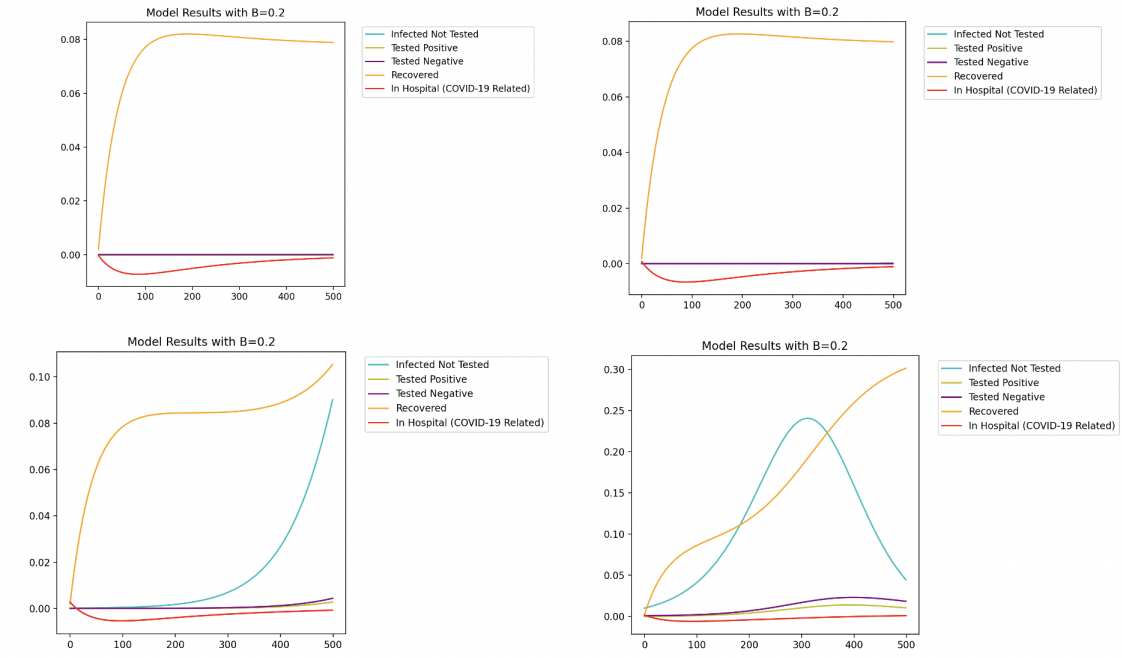

Figure 4: Simulations with four different initial conditions and $\beta = 0.2$

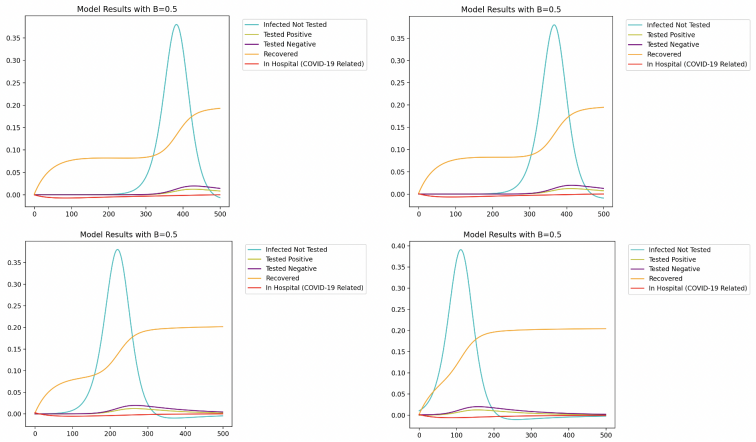

Figure 5: Simulations with four different initial conditions and $\beta = 0.5$

Now we can update the value of β to be equal to 0.5. Here the initial peak of those infected is prominent throughout time. We can note that the recovery rate drops as peak of infection occurs. This makes sense because more individuals are getting sick than are recovering. A similar pattern occurs for the same four initial conditions but with $\beta = 0.7, 1$. The hospital rates stay steady throughout each graph which means there could be an issue with the model or people are being admitted as quickly as they are being released. The number of individuals in the community testing negative and positive are around the same for each initial condition set. This could mean that there needs to be more tests amongst the infected community.

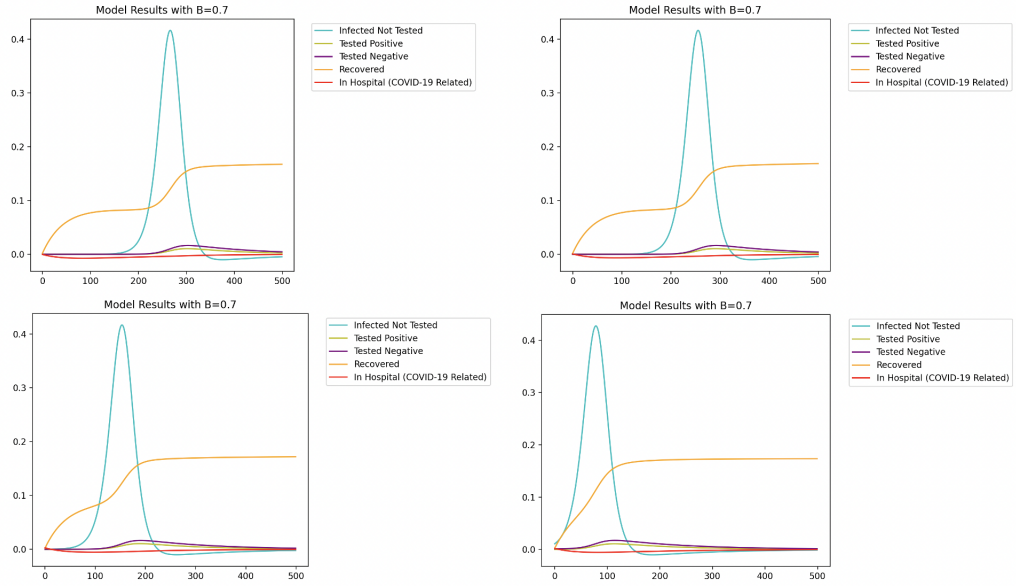

Figure 6: Simulations with four different initial conditions and $\beta = 0.7$

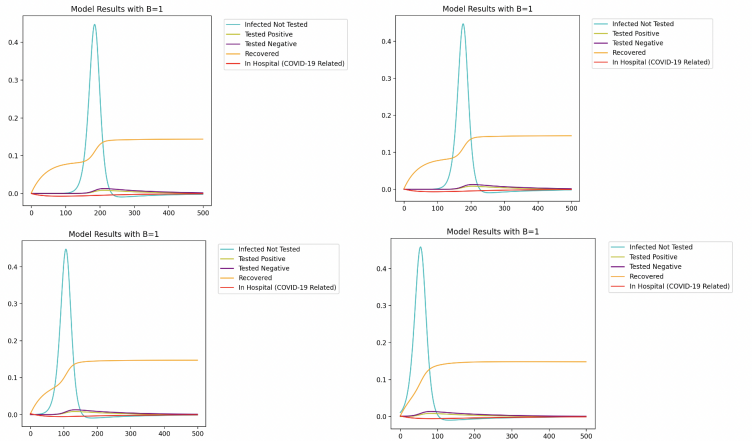

Figure 7: Simulations with four different initial conditions and $\beta = 1$

5 Conclusion

There have been many factors that have affected the spread of COVID-19. Differential equations can offer insight into the changes of populations over time. This allows scientists and epidemiologists to prepare for changing conditions and prepare the community with the best ways to protect themselves. The model I have created shows just a few scenarios of initial community data but it can be modified to include multiple sets of initial conditions to decide what the best prediction for the future might be. It is important to keep in mind the changes in communities and to learn how modeling and simulation is important for keeping ourselves and others around us safe.

6 Appendix

```
import numpy as np
import matplotlib.pyplot as plt

def seir_model(current_values):
    alpha = 0.25
    beta = 0.2 #?
    beta_H = 0.2 #?
    gamma = 0.05
    epsilon = 0.00001
    f = 0.3
    p = 0.4
    mu_n = 0
    mu_i = 0.001
    mu_hn = 0.025
    mu_hi = 0.025
    h_n = 0.0001
    h_i = 0.01
    d_n = 0.2
    d_i = 0.05

    S_prev, I0_prev, Ip_prev, Im_prev, H_s, R_prev, H_im, H_ip, H_R = current_values

    S_next = -(mu_n*S_prev)-beta*(I0_prev + Im_prev) - (S_prev*h_n) + (H_s*d_n)
    I0_next = (S_prev*beta*Im_prev) + I0_prev*(-mu_n + S_prev*beta -(1-f)*(alpha -
        gamma)*gamma*((alpha*(1-f)+gamma*f)/((gamma**2)*(1-f)+(alpha**2)*f)) + f*gamma*(2*p-1)-h_n)
```

```
    Ip_next = - (Ip_prev*mu_i) + (I0_prev*f*gamma*p) - Ip_prev*(h_i + (alpha*gamma)/(alpha-gamma))
    Im_next = - Im_prev*(mu_n + h_n + (alpha*gamma)/(alpha-gamma)) + I0_prev*f*gamma*(1-p)
    R_next = -
        R_prev*(mu_n+h_n)+Im_prev*((alpha*gamma)/(alpha-gamma))+I0_prev*((1-f)*(alpha-gamma)*gamma*((alpha*(1-f)+ga
        + (H_im * d_n) + (H_ip*d_i) + (H_R*d_n)
    Hs_next = - H_s*d_n + S_prev*h_n - H_s* mu_hn - H_s * beta_H * H_im
    Him_next = I0_prev*(1-p)*h_n - H_im*h_n*(1-p) - H_im*d_n + H_s*beta_H*H_im - H_im*mu_hn
    Hip_next = - H_ip*d_i + Im_prev*h_n*p + Ip_prev*h_i + I0_prev*p*h_n - H_im*(mu_hi + epsilon)
    HR_next = - (H_R*d_n) + (R_prev*h_n) - (H_R* mu_hn)
    return [S_next, I0_next, Ip_next, Im_next, Hs_next, R_next, Him_next, Hip_next, HR_next]

def euler(current_values, h):
    numerical_values = []
    estimated_values = seir_model(current_values)
    for i in range(0, len(current_values)):
        numerical_values.append(current_values[i] + h*estimated_values[i])
    return numerical_values

# set as initial values first
numerical_solutions = []
approx_values = [[1-0.0010001, 0.0000001, 0, 0, 0.0001, 0, 0.1, 0, 0],
       [1-0.0010001, 0.0000001, 0, 0.0000001, 0.0001, 0, 0.1, 0.001, 0.00001],
       [1-0.0010001, 0.0001, 0, 0.00001, 0.0001, 0, 0.1, 0.003, 0.0001],
       [1-0.0010001, 0.01, 0, 0.001, 0.0001, 0, 0.1, 0.002, 0.001]]
h = 0.1
for initial_conds in approx_values:
    current_values = initial_conds
    current_numerical = []
    for i in range(0,500):
        updated_values = euler(current_values,h)
        current_numerical.append(updated_values)
        current_values = updated_values
    numerical_solutions.append(current_numerical)


for sol_list in numerical_solutions:
    s_list = []
    i0_list = []
    ip_list = []
    im_list = []
    h_list = []
    r_list = []
    him_list = []
    hip_list = []
    hr_list = []
    for sol in sol_list:
        for i in range(0, len(sol)):
            if i == 0:
                s_list.append(sol[i])
            elif i == 1:
                i0_list.append(sol[i])
            elif i ==2:
                ip_list.append(sol[i])
            elif i == 3:
                im_list.append(sol[i])
            elif i == 4:
                h_list.append(sol[i])
            elif i == 5:
                r_list.append(sol[i])
```

```
        elif i == 6:
            him_list.append(sol[i])
        elif i == 7:
            hip_list.append(sol[i])
        else:
            hr_list.append(sol[i])

#plt.plot(s_list, c = 'b', label = 'Susceptible Population')
plt.plot(i0_list, c ='c', label = 'Infected Not Tested')
plt.plot(ip_list, c = 'y', label = 'Tested Positive')
plt.plot(im_list, c = 'purple', label = 'Tested Negative')
#plt.plot(h_list, c = 'c', label = 'In Hospital (not related)')
plt.plot(r_list, c = 'orange', label = 'Recovered')
#plt.plot(him_list, c = 'y', label = 'In Hospital (not related)')
plt.plot(hip_list, c = 'r', label = 'In Hospital (COVID-19 Related)')
#plt.plot(hr_list, c = 'orange', label = 'Recovered (not related)')
plt.title('Model Results with B=0.2')
plt.legend(bbox_to_anchor=(1.05, 1.0), loc='upper left')
plt.tight_layout()
plt.show()
plt.clf()
```