Caitlyn Jones

Prof. Yangyang Wang

MATH 40A: Intro to Applied Math

6 November 2024

Lab 4

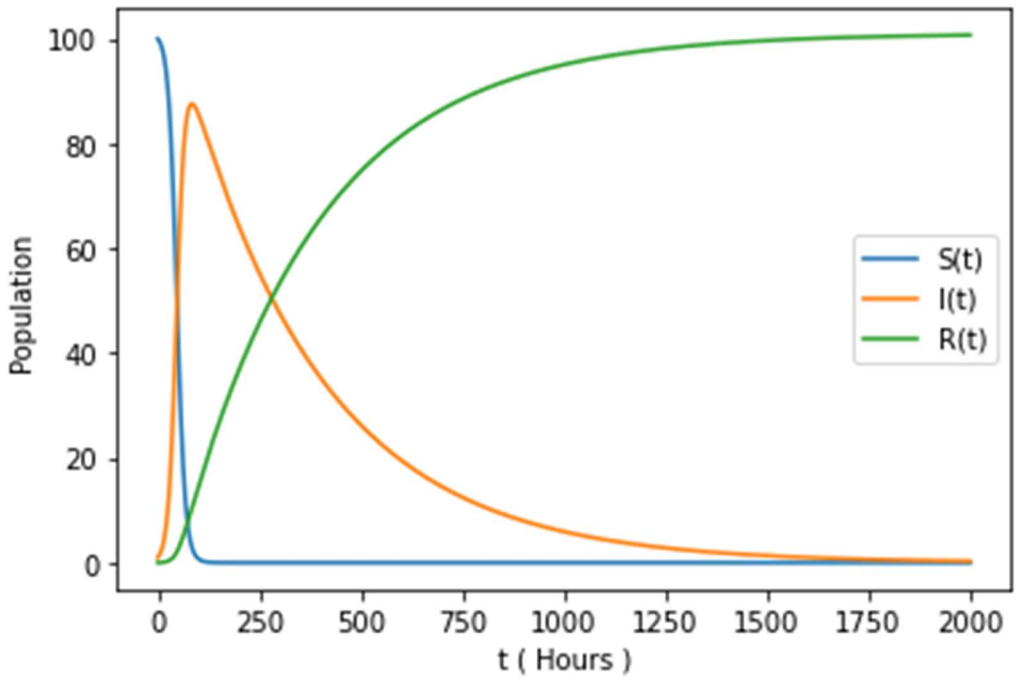
1.

```
beta = 1.0/1000 #Infection rate
gamma = 1.0/(24*14) #Recovery rate


'''
The function below defines the differential equations
for the SIR model
'''
def deriv (x,t):
    ifc = beta *x [0]* x [1] #Rate of new infections based in the population of susceptible and infected people
    rec = gamma * x [1] #Rate of recoveries based on the population of infected people
    return np.array ([-ifc, ifc - rec, rec]) #Returns the rate of change for S, I, and R


time = np.linspace (0, 2000, 1000) #Variable to track time, 2000 hours
xinit = np.array ([100 ,1 ,0]) #Initilizing variables for S, I, and R
x = odeint ( deriv , xinit , time ) #Integrates the equations for S, I, R over time

plt.figure () #Creates a new figure
p0, = plt.plot (time, x[: ,0]) #Defines and plots the values S(t)
p1, = plt.plot (time, x[: ,1]) #Defines and plots the values I(t)
p2, = plt.plot (time, x[: ,2]) #Defines and plots the values R(t)
plt.legend ([ p0 , p1 , p2 ] ,["S(t)","I(t)","R(t)"]) #Labels the graphs S(t), I(t), R(t)
plt.xlabel ("t ( Hours )") #Labels the x-axis in hours
plt.ylabel ("Population") #Labels the y-axis as population count
plt.show () #Displays the plot
```



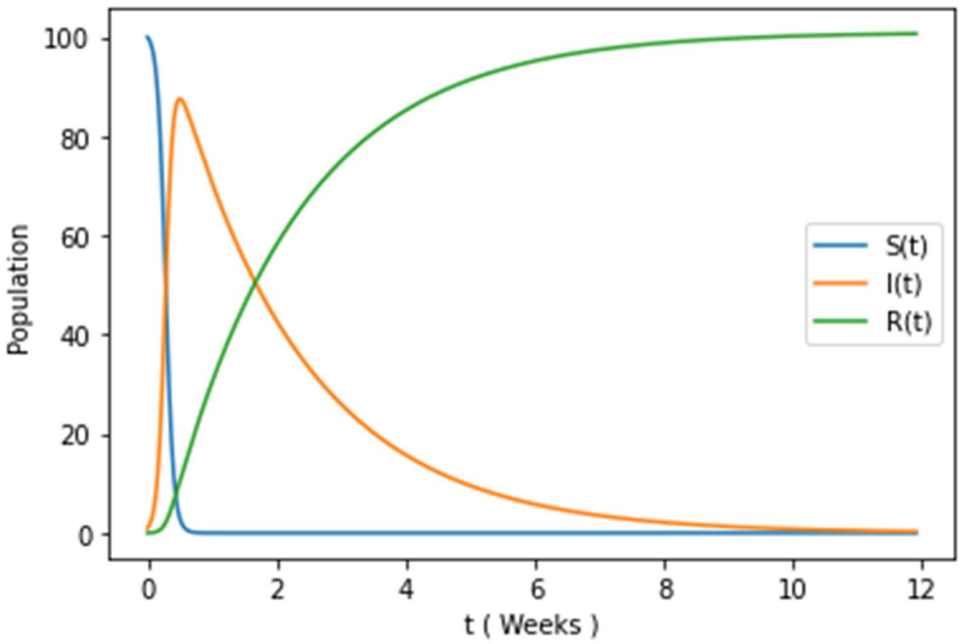
2. S(0) for this model is 100, as you can see from the graph and by looking at the first value in x[:, 0]. S(0) of the model from class was 999. We can round this number up to 1000, which will allow us to assume that the model from class is directly proportional to the model from this homework.
3. To modify the function so that time is measured in weeks instead of hours, I first changed the rate of infection and rate of recovery from per hour to per week.

```
beta = (1.0/1000)*(168)
gamma = 1.0/2
```

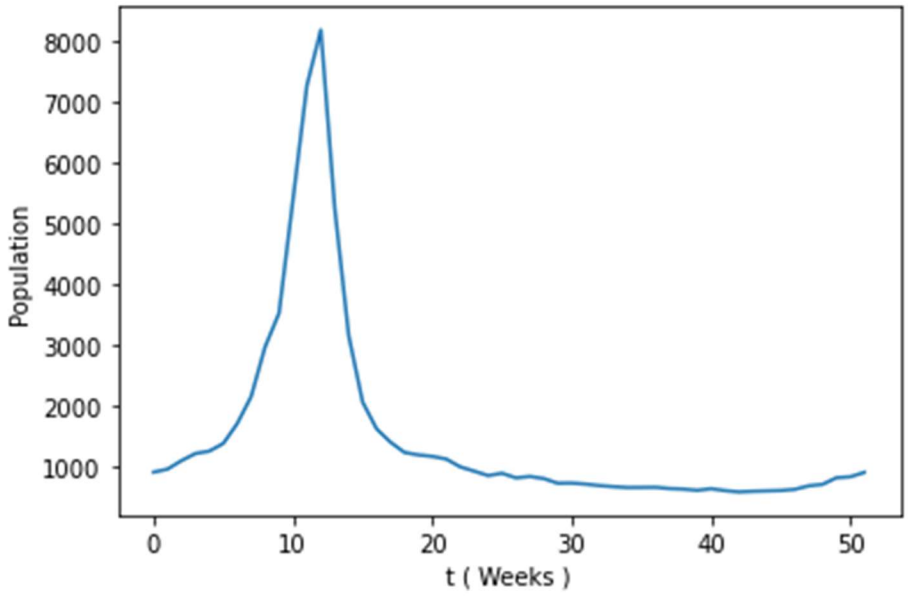
Then, I updated the value of time, so that the function is graphed over a period of weeks not hours.

```
time = np.linspace (0, 2000/168, 1000)
```

Here are the results. The graph looks the same as before due to the scaling of the numbers. Because we are changing both rates and the period of time, we will receive the same progression on both graphs.



4. Below is the plot of the first year of data.



When trying to fit the data, I had a little trouble with the changing the parameters. The closest I got was when I changed the parameters for time, initial values, and gamma.

```
beta = (1.0/1000)*168
gamma = 0.2


def deriv (x,t):
    ifc = beta * x[0] * x[1]
    rec = gamma * x[1]
    return np.array ([-ifc, ifc - rec, rec])


time = np.linspace (0,52, 52)
xinit = np.array ([8000,400,0])
x = odeint ( deriv , xinit , time )
```

Below are the results.