Lab 2: Disease Dynamics and the SIR Model

1. The SIR Model

During the COVID-19 pandemic, there has been intense interest in mathematical models that could predict the course the pandemic would take and give us insights into policies that should be enacted to minimize its damage. One of the most basic such models — which falls under the fancy-sounding title of *quantitative epidemiology* — is the *SIR model*, which simulates disease progression in a population.

Figure 1: The SIR model captures the evolution of a disease among three segments of the population. Note that we will use *b* and *g* in place of *β* and *γ*, respectively.

"SIR" is an acronym, referring to the three categories of people into which we choose to divide the population (see Figure 1). There are two important assumptions/approximations that go along with this flow:

1. The disease being modeled is nonlethal, and more generally the population doesn't change in size. The algebraic equation that expresses this statement is

   $$N = S + I + R = \text{constant}.$$

   (S stands for the number of susceptible people at a given time.)

2. Once people recover, they have immunity from the disease. Hence, the "recovered" category and the "susceptible" category have no overlap.

As we saw in class, the basic SIR model has three differential equations:

$$\frac{dS}{dt} = -bIS$$

$$\frac{dI}{dt} = bIS - gI$$

$$\frac{dR}{dt} = gI$$

Here, the variables S, I, and R represent the fraction of the population (values between 0 and 1) that are in each of the three categories. From the model, we also defined a key metric R_0 that

describes the reproductive potential of an infectious pathogen.

$$R_0 = \frac{b}{g}$$

2. Understanding the SIR Model

Notice that if $I = 0$, then the rate of change of S with respect to time is zero. Does this make sense? Why?

Yes. If there are no infections, then the disease cannot be transmitted, and no susceptible people can become infected.

If you add up the right-hand sides of all three equations, what do you get? Why does this make sense?

Zero. This makes sense because the sum of these derivatives will be the derivative of the total population size, and the total population size is not changing.

3. Getting familiar with the basic model

First, we will get familiar with the basic model in MATLAB. You will need the m-file "SIR_with_social_distancing.m" and mat-file "CA_COVID_data_2021-01-12.mat" which you can download from the Lab 2 assignment page in Canvas. You can open the m-file in MATLAB using the "Open" button near the upper left of the MATLAB window.

This file is a "script" containing code for exploring variants of the SIR model. A script is file with a list of commands, just like those you would enter at the command line. However, organizing the commands in a script has several advantages. It allows us to save and share our work. It also allows us to easily repeat a set of commands after making small changes. Finally, it allows us to organize our commands into "blocks" that we can execute in a stepwise fashion (as described in the lecture). We can execute all the commands in a block at once by clicking the "Run Section" button or the "Run and Advance" button in the "Editor" tab at the top of the MATLAB window. We will use this script to explore the SIR model, but you will need to fill in missing items and modify it to complete the lab.

The first block of code contains only comments, which will be useful for interpreting the vector format of our differential equation model. In the second block, we define the two rate constants of our model. What is R0 for this version of the model? Is that a reasonable value for a model of SARS-CoV-2?

2.56 That is about right for SARS-CoV-2

Execute Block 2 and move on to Block 3. This block does two important things. First, it defines our timeframe for simulating the model. We will use real data for the progression of

the epidemic in California in this lab. With that in mind, we will measure time in days since the approximate beginning of the outbreak in California. To handle dates, we will use two built-in MATLAB functions "`datenum`" and "`datestr`". Try the following commands at the command line to learn what the functions do:

```
>> date1 = datenum(now)
>> datestr(now)
>> datestr(now+1)
>> datestr(now+0.25)
```

The `datenum` function converts a date into a number, and the `datestr` function converts these numbers back into human friendly dates. What does a difference or change of 1 in the date numbers represent (in terms of hours or days)?

It represents a difference of 1 day

The last piece of Block 3 defines the differential equations for the model. I described how we do this in lecture, but it still may look unfamiliar. In this command, we are defining `dy` as a "function handle". This is a tool in MATLAB that allows us to write equations for functions that depend on dependent variables. MATLAB can then interpret these functions as differential equations when we use ode45. The @ character tells MATLAB that we are defining a function handle. The `(t,y)` tells MATLAB that the function depends on two variables called `t` and `y`.

The square brackets define the output of our function as a (column) vector. Each row corresponds to one output. If you look carefully at the first row "`-b*y(1)*y(2)`" you should see that it looks strikingly similar to our differential equation for dS/dt. This is indeed our representation in MATLAB of that differential equation. Similarly, for the following two lines with the next two differential equations. Notice that we have replaced S with `y(1)` and I with `y(2)`. What does `y(3)` represent?

R (the recovered population)

In Block 4, we define the initial conditions and solve the differential equations to simulate the model. To make it easy to change our initial conditions sensibly, we first define a variable "`frac0`" to represent the fraction of the population that is infected at time zero. We will also assume that at time zero, there are zero people who have already recovered from infection. We will set the total population size to be 1, so that a value of 0 represents 0% of the population and a value of 1 represents 100%.

Edit line 24 to define the **three-element** initial condition vector `y0` (interpreted as [S I R]) at time zero, using the variable `frac0`. (Hint: What should the sum of the three values be?) Enter the completed line here:

```
y0 = [(1-frac0) frac0 0]; % [S I R]
```

(the parentheses around 1-frac0 are not necessary, but can help for clarity)

Execute Block 4 to simulate the model. The output of the model will now be stored in `y`. Looking in the workspace window, what are the dimensions of `y`?

323 x 3

Based on these dimensions, what do you think the columns of `y` represent?

The first column is S, the second is I, and the third is R

What does each row represent?

Each row represents one time point.

Execute Block 5 to plot the results of our simulation. What happened to each population during the simulation?

The susceptible population starts at about 1, declines sharply, and then levels out around 0.1. The Recovered population starts around zero, rises sharply, and levels out around 0.9. The Infected population starts at zero, rises to a peak of about 0.25, then drops back to about zero.

Since we organized our code in blocks, we can easily re-run our simulation using different parameters to explore the model. Try adjusting the rate constants `b` and `g` and re-running the model. (You will need to execute Blocks 2, 3, 4, and 5 again). If you double both rate constants, how does the model output change? Does the total fraction of the population that gets the virus change (this will be the final value of R at the right side of the plot)?

Everything happens faster, but the fraction of the population that gets the virus stays the same.

What about if you double `g`, but leave `b` at the original value of 0.4275?

In this case, the model output is dramatically changed. The outbreak occurs much later, and fewer people get the virus.

4. Simulating Social Distancing

In lecture and above, we simulated the basic SIR model to understand where the metric R_0 comes from, and how it determines whether a pathogen is capable of triggering outbreaks. However, you should have noticed that the simulated timecourse of infections looked pretty different from the numbers we've actually observed during the COVID-19 pandemic. One major reason for this is because the behavior of individuals changed during the pandemic. We will now adapt the SIR model to make a slightly more complicated version to model these changes, and appreciate the impact that social distancing, masks, and other measures can have. We will use real data from California during the COVID-19 pandemic to help us explore.

In the SIR model, which parameter (or rate constant) should be affected by social distancing and mask wearing?

b

In Block 6, we will load data for California from a saved .mat file. The .mat format is a MATLAB format for saving MATLAB variables for use in future MATLAB sessions. I created this .mat file using data downloaded from the rt.live website. Execute Block 6 to load the data. (If this does not work, it is probably because MATLAB cannot find the file. You can get around this by dragging the file into the workspace.)

At the command line, type `CA_data` and press enter to see what was retrieved from the file. You will see that the data is stored in a new variable type "struct" or structure. A structure is handy way to organize data with multiple components together in MATLAB. In this case, the fields of the structure contain different measures of the pandemic in California.

Use Block 7 to plot the estimated new infections over time in California. These values are based on reported cases, but correcting for changes in the testing rate over time, to try to estimate the actual timecourse of new infections (but it is just an estimate!). This curve should look more like the ones you're used to seeing on news websites. Try plotting other parameters stored in the structure. If you overlay plots (remember from last week the command that allows you to overlay the plots on the same graph) of `R_eff` and `b_over_g_ratio` on the same plot, how do they compare?

They look very similar, but b_over_g ratio gets to be higher than R_eff

5. Short aside: Defining custom functions in MATLAB

We have used the @ symbol today to define our differential equations. More generally, the @ symbol can be used in MATLAB to define a custom function that you can use at the command line any time you like. For example, try the following:

```
>> exampleFunction = @(x) 3*x;
>> exampleFunction(2)
```

@(x) indicates that you can plug anything in for *x* (the "argument" of the function), and the function will be evaluated at that value. At the command line, use the `dy` function you created in Block 3 above to compute the derivatives *dS/dt*, *dI/dt*, and *dR/dt* at `t=0` and `y = [0.99 0.01 0 0]`. Evaluate it again at `t=10` and the same values for `y`. Did you get the same answer both times? Why?

Yes. Because dy does not have any terms that actually depend on t

6. Back to our social distancing model

In Block 8, we define a g parameter for this model as `gCA` as a constant as we did for the model above. For the other rate b, however, we will now define it as a function (instead of just a number) to allow it to vary over time. I've included details of how that was done in the box below, but for the purposes of this lab the main point is that `bCA` is now a function that will return a value for any value of `t`.

(Optional) Extra Detail: Making a function `bCA` to model how the transmission rate changes over time.

Our California data has estimated R_{eff} values at each time point, representing the effective reproduction rate of the virus. Based on the fraction of the population that was still susceptible to infection, those values were used to compute estimates of the ratio of the b and g parameters at each time.

To make a function that can be used in our differential equations, we need something that will return a value for any time value (not just the values in our data table). To make such a function, we use MATLAB's interpolation function (`interp1`) to fill in values not present in the table by interpolating between the nearest values that were there. If that line is a bit hard to understand at the moment, that is fine (it's more advanced). The last line in this block defines our function for the b values (called `bCA`). This function multiplies our estimated b to g ratios by our value for g to get the estimates for b.

Block 9 sets up our differential equations for the new model. Here, we make two versions, one that includes the changes in b empirically observed in California due to social distancing and other measures. The other model keeps b constant, as we did in the original model above. What is the numerical value for b that we are using in the second model? (Your answer should be a number like 0.547)

0.4279

Execute this block. Just as you did for `dy` above, now use the `dySocialDist` function at the command line to compute the derivatives *dS/dt*, *dI/dt*, and *dR/dt* at `t=0` and `y = [0.99 0.01 0 0]`, and evaluate it again at `t=10` and the same values for `y`. Now do you get the same answer both times? Why not?

No. Because for dySocialDist, the first two derivatives do depend on t (because bCA depends on t)

Block 10 defines our initial conditions and solves the equations for the two models. Here, we define the fraction of the population at time zero infected with the virus as a small number in scientific notation. Using an estimate of the total population of California from the `CA_data` structure, what does this initial condition correspond to in terms of total people?

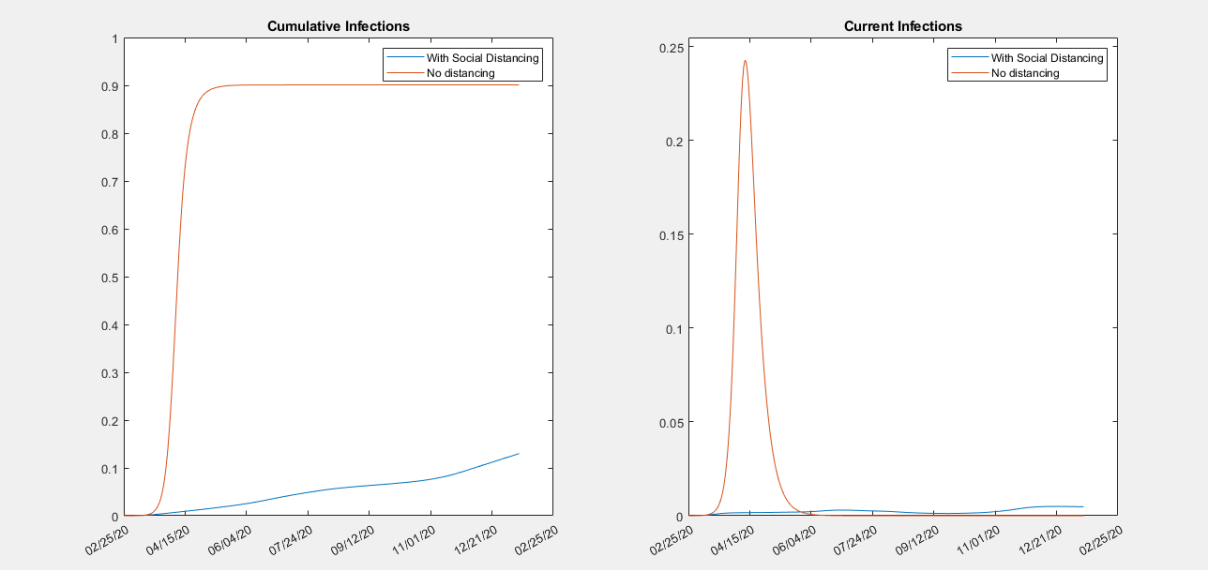
355.5 people

Execute blocks 10 and 11 to simulate the models and plot the results. You should see two plots, each with two curves (one for each model). For the plot on the left, we plotted the sum of the second and third columns of the model output (e.g., `ySocialDist(:,2) +`

`ySocialDist(:,3)`), but the plot is titled "Cumulative Infections". Why is that an appropriate title for the plot?

The total cumulative number of people who have been infected is equal to the recovered population plus the currently infected population. That is what the sum of columns 2 and 3 gives you.

Paste in a screenshot of your plots:



If you have time:

Edit Block 12 to create a new plot indicating the total number (cumulative) of infections prevented by social distancing and other measures as a function of time. Copy and paste your code for Block 12 below:

```
cumulativeNoDist= yNoDist(:,2) + yNoDist(:,3);
cumulativeSocialDist= ySocialDist(:,2) + ySocialDist(:,3);
plot(t, cumulativeNoDist - cumulativeSocialDist);
```

Paste a screenshot of your plot below: