Lab 5: The SEIR Model

Due: Friday, February 28, 2020 at 5 pm

1 Introduction

In this lab, you'll learn how to solve a system of ODEs with more than two dependent variables. As an example, we'll explore the SEIR model, which adds an exposed population into the SIR model. Then, we'll change the system to include quarantine and see how this affects the system. We'll also learn some new techniques for plotting larger data sets, including creating panels of graphs using `subplot`.

Your lab submission should contain a pdf file with your solutions to exercises, including any images. This should be typed in LaTex. A LaTex template is available on Moodle, or you may use your own. You should also submit any **scripts** (.m files) that you use. **Lab # 5 Exercises are due Friday, February 28th at 5 pm and should be submitted on Moodle. Please include your name in your filename.**

2 The SEIR model: Solving Larger Systems

The SEIR model is a slight modification to the SEIR model. In addition to the susceptible, infected, and recovered categories, we now include an exposed population, E. We'll assume that time is measured in weeks. The parameters are:

β = the rate of exposure from interactions between susceptible and infected individuals

η = the rate of exposed individuals who become infected.

ν = the rate of recovery

γ = the rate at which recovered individuals become susceptible again. If recovered individuals stay immune, then $\gamma = 0$.

This model is represented by the system of equations:

$$\begin{aligned}
\frac{dS}{dt} &= -\beta SI + \gamma R \\
\frac{dE}{dt} &= \beta SI - \eta E \\
\frac{dI}{dt} &= \eta E - \nu I \\
\frac{dR}{dt} &= \nu I - \gamma R
\end{aligned}$$

Note that these equations sum to $\frac{d(S+E+I+R)}{dt} = 0$, so there is no change in the total population. (So there are no vital dynamics.)

We've already learned how to solve a system of ODEs with two dependent variables; now, we'll have four dependent variables. The process is exactly the same; we just have more variables to manage. First, let's create a function file. This time, we're going to save the function file separately from the main code. Create a function `SEIRfunc` and save the file as `SEIRfunc.m`. The file name has to match your function name. We'll choose parameters $\beta = .05$, $\eta = .5$, $\nu = .5$, and $\gamma = .1$.

```
function v = SEIRfunc(t,Y)
    b = .05
    eta = .5
    nu = .5
    gamma = .1
    S = Y(1);
    E = Y(2);
    I = Y(3);
    R = Y(4);
    v(1) = -b*S*I+ gamma*R;
    v(2) = b*S*I - eta*E;
    v(3) = eta*E - nu*I;
    v(4) = nu*I - gamma*R;
    v = v';
end
```

This file will not run. That's okay! It's still working. Let's create a separate file called `SEIRscript.m`. In this file, we'll solve the function using `ode45` with initial conditions $S = 99$, $E = 0$, $I = 1$, $R = 0$. So the population is fixed at 100 and there is one infected individual. *There is no need to include the function SEIRfunc in this file - MatLab knows to find the function file!*

```
[t,Z] = ode45(@SEIRfunc, [0 20],[99,0, 1 ,0]);
```

Previously, we plotted a system one variable at a time. For example, we could use

```
plot(t, Z(:,3))
```

to plot the third dependent variable (the infected population) over time. However, there is a shortcut to plot all dependent variables on the same graph! Instead, use

```
plot(t, Z)
```

This will plot all four dependent variables over time! It might be hard to tell which variable is which. Let's add a legend. Simply list the dependent variables in the same order as you entered them in `ode45`.

```
legend('Susceptible','Exposed', 'Infected','Recovered')
```

You can adjust the axes manually:

```
axis([0 20 0 100])
```

And of course, we should always include labels:

```
title('SEIR model')
xlabel('Time (in weeks)')
ylabel('Number of people')
```

Right now, it looks like the disease never dies out; the epidemic stabilizes with a near-constant infected population. This is because $\gamma > 0$, which means that recovered individuals keep returning to the susceptible population. In the next section, we'll explore how different recovery rates ν affect the outcome.

3 Using Subplots

When analyzing a model, it can be helpful to plot different parameter values. One of the interventions we discussed in class was to try to increase the ν parameter. In other words, we can combat the disease by treating the infected population to increase the rate of recovery.

We can go into our function and change the ν value manually. But this would be extremely tedious; we'd have to create a new function for each ν value we want to plot! Instead, let's include ν as a parameter in our function. Modify the first line of your function code:

```
v = SEIRfunc(t,Y,nu)
```

Then erase or comment out the line in your function where ν is defined. You won't need this; ν will be defined based on the input of the function. Unfortunately, MatLab demands that the input to `ode45` is a function with inputs t (a number) and Y (a vector). So we'll still need to create a new function, but we can just do this in-line. To define a function with $\nu = .5$, type

```
tempfunc = @(t,Y) SEIRfunc(t,Y, .5)
```

Now, we can plug this function into `ode45` and plot.

```
[t,Z] = ode45(tempfunc, [0 20],[99,0, 1 ,0]);
plot(t, Z)
```

Now, suppose we want to compare this to a recovery rate of $\nu = .7$. Let's include the code to solve the system and plot for $\nu = .7$.

```
tempfunc = @(t,Y) SEIRfunc(t,Y, .7)
[t,Z] = ode45(tempfunc, [0 20],[99,0, 1 ,0]);
plot(t, Z)
```

The problem is that when we run the code for $\nu = .5$ and $\nu = .7$ together, the second plot overrides the first plot, and we only see the plot for $\nu = .7$. To fix this, we'll view the plots side by side with `subplot`. The command `subplot(m,n,i)` creates an $m \times n$ grid of plots, and places the next plot created in the i^{th} spot (counting across the first row, then the second row, etc.) We want to create a 1×2 grid for these plots.

```
tempfunc = @(t,Y) SEIRfunc(t, Y, .5)
```

```
[t, Z] = ode45(tempfunc, [0 30], [99,0,1,0]);
subplot(1, 2,1)
plot(t,Z)
title('nu = .5')
legend('Susceptible', 'Infected', 'Exposed', 'Recovered')

tempfunc = @(t,Y) SEIRfunc(t, Y, .7)
[t, Z] = ode45(tempfunc, [0 30], [99,0,1,0]);
subplot(1,2,2)
plot(t,Z)
title('nu = .7')
legend('Susceptible', 'Infected', 'Exposed', 'Recovered')
```

This worked well for two different parameter values. However, if we wanted to create a large subplot with many ν values, we'd basically repeat the same code over and over with different ν values. This is a sign that we should use a for loop instead!

4 Changing Parameters with a For Loop

Now, let's look at a side-by-side comparison of many different ν values: $\nu = .5, .6, .7, .8, .8, and 1$. If we use a for loop over $i = 1 : 6$, then we should set $nu = .5 + .1 * (i - 1)$. (Check that this gives the values you want.) We'll re-run the same code that we wrote for $\nu = .5$ and $\nu = .7$.

```
for i = 1:6
nu = .5 + .1*(i-1)
tempfunc = @(t,Y) SEIRfunc(t, Y, nu)
[t, Z] = ode45(tempfunc, [0 30], [99,0,1,0]);
subplot(2, 3,i)
plot(t,Z)
axis([0 20 0 100])
legend('Susceptible', 'Infected', 'Exposed', 'Recovered')
end
```

This will plot size different graphs in a 2×3 grid. It would be really helpful to label each graph with each ν value. The problem is, we can't include `nu` in the title since it is a number, not a string. (This is the fancy programming word for text.) We can use the command `num2str(nu)` to convert this number to a string.

```
title(num2str(nu))
```

If you want to be really fancy, you can use the command `strcat` to concatenate the string `'nu = '` with the value of `nu`.

```
title(strcat('nu = ',num2str(nu)))
```

5 Lab #5 Exercises

In this lab, we saw that changing increasing ν, the rate of recovery, wasn't enough for the disease to die out. So now we're going to explore a stronger intervention - quarantine. We'll start with the SEIR model, but we'll add a variable Q for quarantined individuals. Exposed individuals who become infected have probability q of becoming quarantined. Then quarantined individuals will recover at the same rate, ν, as the infected individuals. The new system is:

$$\begin{aligned}
\frac{dS}{dt} &= -\beta SI + \gamma R \\
\frac{dE}{dt} &= \beta SI - \eta E \\
\frac{dQ}{dt} &= q\eta E - \nu Q \\
\frac{dI}{dt} &= (1-q)\eta E - \nu I \\
\frac{dR}{dt} &= \nu(I+Q) - \gamma R
\end{aligned}$$

We'll use the parameters $\beta = .05$, $\eta = .5$, $\nu = .5$, and $\gamma = .1$. Time is measured in weeks. The probability q of being quarantined will vary. The initial conditions are $S = 99$, $I = 1$, and $E = Q = R = 0$. (So there is one infected individual in a population of 100.)

Problem 1

Create a function, `SEQIRfunc`, for the above system with $q = .5$. Create a script file `SEQIRscript`. Use `ode45` to solve from time $t = 0$ to 20 weeks. Plot the solution, including a legend, axis labels, and a title. How does including a quarantine change the disease progress and outcome?

Problem 2

Modify your function `SEQIRfunc` so that the probability q of being quarantined is a variable. Create a new script, called `SEQIRloop`. Using a for loop, define a temporary function `tempfunc` for the q values $q = .1, .2, ..., 1$. Plot the solutions in a 2×5 grid using subplot. Label each plot with the value of q, and include a legend. You may have to resize your output window after running to improve the display.

Include the output in your writup. Also, include your observations and recommendations to health officials. What value of q would you recommend so that the disease ultimately dies out? (In other words, the infected population I approaches zero.)