Tutorial 6: Working with ODE's

Getting Started

1. This tutorial will emphasize working with ODE's using the Lorenz equations discussed in class.

2. Download LorenzEq.m from the "Lectures" website. Make sure this file is saved in your Matlab path.

3. Start Matlab.

Storing Data in Cells

4. Most studies using computer simulations follow a simple formula. First, you write down a model defined by a set of parameters. Then, you explore what happens as you change parameters. You'll get to do this for the Lorenz equations. In class, I made two runs (the diary file with the commands I entered is online). The first run used the default parameters, and the output was chaotic. The second run used a new set of parameters with the second parameter, r, equal to 10 (default was 28), and the system settled down to a stable equilibrium. We'll design a simple experiment to determine where the transition from stability to chaos occurs.

5. First, define the initial conditions

   ```
   >>x0=[1 1 1];
   ```

   Now, let's solve the equations from the times 0 to 50:

   ```
   >>[t, XYZ]=ode23(@LorenzEq, [0 50], x0);
   ```

   To see the data, type

   ```
   >>plot(t, XYZ)
   ```

6. For our experiment, we'll need to be able to specify the parameters. To run with r=15 and the other parameters set to their defaults, type

   ```
   >>[t2, XYZ2]=ode23(@LorenzEq, [0 50], x0,[], [10, 15, 8/3] );
   ```

   Be sure to give the outputs unique names.

7. Now, let's take a closer look at the output. Type "whos" and compare the sizes of t and t2. Are they the same? Matlab picks times so that a plot of

the output is smooth. It does this by having the times closely spaced when the function is changing rapidly. To see this, type

```
>>plot(diff(t2),'x');
```

For the second run, the function oscillates at the beginning and settles down later on. The plot shows the length of time separating successive values in XYZ; small at the beginning, larger near the end.

8. The only problem with Matlab choosing the times is that it makes it hard to compare two ODE solutions. We can force Matlab to give us answers at a specific set of times. Create a vector of times:

```
>>t0=linspace(0,50, 500);
```

Then pass these to ode23:

```
>>[t3, XYZ2]=ode23(@LorenzEq, t0, x0,[], [10, 15, 8/3] )
```

t3 should be exactly the same size as t0.

9. We're almost ready for our experiment. We still need a way to tell chaotic from "well-behaved" solutions. There are a number of formal mathematical techniques, but we'll just use our eyes. The hallmark of chaos is "sensitive dependence on initial conditions." This means that if the model is chaotic, then two simulations starting from slightly different initial conditions will diverge. To see this, repeat the first experiment, using the t0

```
>> [t, XYZ]= ode23(@LorenzEq, t0, x0);
```

Then, repeat, but add 0.001 to x0:

```
>>[t2, XYZ2]= ode23(@LorenzEq, t0, x0+0.001);
```

Now, plot the difference between the two simulations:

```
>>plot(t, XYZ2-XYZ);
```

Repeat this procedure for the r=15 case.

10. Now, we're ready for the experiment. We know that r=15 is well-behaved but that r=28 is chaotic. It follows, that for some r in this range, the model shifts behavior. To try to find this, we'll create a vector of r's and try each of them. Set r=[15 20 25 27 28]. We'll then loop over the r's and store each into a layer of a 3D array. Before running the experiment, preallocate the storage for the arrays:

```
>>XYZ=zeros(length(t0),3,length(r));XYZ2=XYZ;
```

We'll store the standard solutions in XYZ and the perturbed solutions in XYZ2. The run becomes

```
>>for j=1:5;
   [t,XYZ(:,:,j)]=ode23(@LorenzEq, t0, x0,[],[10 r(j),8/3]);
   [t,XYZ(:,:,j)]=ode23(@LorenzEq, t0, x0+0.001,[],[10 r(j),8/3]);
   end
```

11. To see the results, put the plot statement in a loop and use the pause statement:

```
>>for j=1:5;plot(t,XYZ(:,:,j)-XYZ2(:,:,j));disp(r(j));pause;end
```

When Matlab encounters a pause command, it stops. You must hit a key to force Matlab to continue.

12. You should see that at around r=25, the differences between the two solutions seem to oscillate. According to our qualitative analysis, 25 seems like a good place to declare the transition. You could improve this analysis by increasing the length of the simulation and by trying additional r's between 20 and 27.