

## Solving Systems of ODE's with Matlab's ode45

### 0.1 Previous encounters with ode45

Earlier in the semester we saw how to solve a linear, first-order ODE of the form

$$y'(t) = f(t, y). \quad (1)$$

We called the Matlab function `ode45` using the syntax

$$[t \ y]=\text{ode45}(\text{@yprime}, [t0 \ tf], y0).$$

The LHS tells Matlab to store the output from `ode45` in two vectors, `t` and `y`. The arguments on the RHS are:

- `@yprime` : handle for Matlab function returning the value of (1)
- `[t0 tf]` : Beginning and ending time for the desired solution
- `y0` : Initial condition, i.e.  $y(t_0)$

Previously, we were able to run `ode45` and then use the data stored in `t` and `y` to produce a plot of the numerical solution using the `plot` command.

### 0.2 Systems of first-order ODE's

You've been learning in lecture how a linear ODE of order greater than one can be rewritten as a system of first-order linear equations. In general, if you're given any system of first-order ODE's, whether they're linear or not, Matlab's ODE solvers can at least attempt to solve them numerically. Take, for example, the following system:

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= -\frac{1}{5}y_2 - \sin(y_1) \end{aligned} \quad (2)$$

How do we model this system with Matlab? If you stop to think about it, the above system is very similar to (1), except that it has two equations instead of one. We might rewrite it as follows to emphasize the similarity:

$$\begin{aligned} y_1'(t, y_1, y_2) &= f_1(t, y_1, y_2) \\ y_2'(t, y_1, y_2) &= f_2(t, y_1, y_2) \end{aligned} \quad (3)$$

Just as before, we'll have to write a Matlab function that will take in `t` and `y` and return the value of the derivative, except now `y` is going to be a vector containing the two values  $y_1$  and  $y_2$ . Remember that if `y` is a row or column vector, we can access its  $i^{\text{th}}$  element using the syntax `y(i)`.

```
function dydt=yprime(t,y)

dydt(1,1)=y(2);
dydt(2,1)=-0.2*y(2)-sin(y(1));
```

*Question:* What kind of vector does the function `yprime` output?

*Answer:* A *column* vector! This wasn't arbitrary: Matlab's ODE's solvers, including `ode45`, want the function that evaluates the derivative to return a column vector (this *does* appear to be arbitrary!). If you start getting red error messages, this might be one place to check!

**EXAMPLE:** Suppose we want a solution for (2) with  $t \in [0,40]$  and initial conditions  $y_1(0) = 0$  and  $y_2(0) = 3$ . We're going to call `ode45` in the same way as in the 1-dimensional case, except that now our `y0` is a vector of the form `[y1 y2]`. The command will look like:

```
[t y]=ode45(@yprime,[0 40],[0 3]);
```

Now you can plot `t` vs. `y` to see the solutions, or `y1` vs. `y2` to see the phase portrait. Try this with a few different choices of initial conditions.

### 0.3 Solving systems of ODE's with one or more free parameters

In Lab 1, we tried to improve our model of oscillatory data by adding terms to our differential equation. What if we wanted to have our `yprime` function be able to take in parameters such as these, in addition to `t` and `y`? Fortunately, the ODE solvers in Matlab accomodate this with ease. Suppose we took the system (2) and let the coefficient of  $y_2$  in the second equation be a free parameter. Then we can redefine `yprime` to be a function of three variables as follows:

```
function dydt=yprime(t,y,a)
dydt(1,1)=y(2);
dydt(2,1)=a*y(2)-sin(y(1));
```

If the derivative function called by `ode45` has additional parameters, `ode45` is called as follows:

```
[t y]=ode45(@yprime,[t0 tf],[y1 y2],[ ],a)
```

The seemingly random `[]` in the list of arguments for `yprime` is an empty placeholder where certain options can be input to `ode45`. If you're curious about that, you can query `help ode45` to see more information. Otherwise, just remember to include it when passing in additional arguments to `yprime`.

*Note:* There is no limit to how many free parameters `ode45` will pass on to `yprime`. You can simply redefine `yprime` to handle more inputs and add those

inputs to the end of the list of `ode45` arguments to your heart's desire. Try generating solutions for a few different values of `a` to see the effect of the parameter on the system.

## 0.4 Contour plots

*NOTE:* You won't need contour plots for this week's homework. However, you might find it handy when answering the energy equation questions in Lab 2!

The Matlab `contour` function will automatically create contour plots for a 3-dimensional set of data. Recall from calculus that the contour of a multivariable function, say  $f(x, y)$ , is a set of pairs  $(x, y)$  for which  $f(x, y) = c$ , where  $c$  is constant.

Suppose we want to plot contours of the function  $f(x, y) = x^2 + y^2$ . First we need to create a grid of data values  $(x, y)$  using the `meshgrid` function:

```
[x,y]=meshgrid(0:.1:1,0:.1:1);
```

To create a plot of 20 different contours of  $f(x, y)$ , type:

```
contour(x,y,x.^2+y.^2,20)
```

As always, you can type `help contour` to see the documentation.

## 0.5 Homework

Write an m-file that solves the following two second-order, nonlinear ODE's:

1.  $y'' + \sin y = 0$
2.  $y'' + \frac{1}{10}y + \sin y = 2 \cos t$

In order to use the techniques discussed in this worksheet, you'll have to rewrite both equations as a system of first-order ODE's: you may want to ask a neighbor or refer to your textbook if you've forgotten how to do this.

**What your script should do:** Produce two plots per system. One plot should contain a phase portrait ( $y_1$  vs.  $y_2$ ) for several different initial conditions. The other should show solutions ( $y(t)$  vs.  $t$ ) corresponding to different initial conditions. They should be clearly labeled, more or less like the ones shown below (but don't worry about having captions).

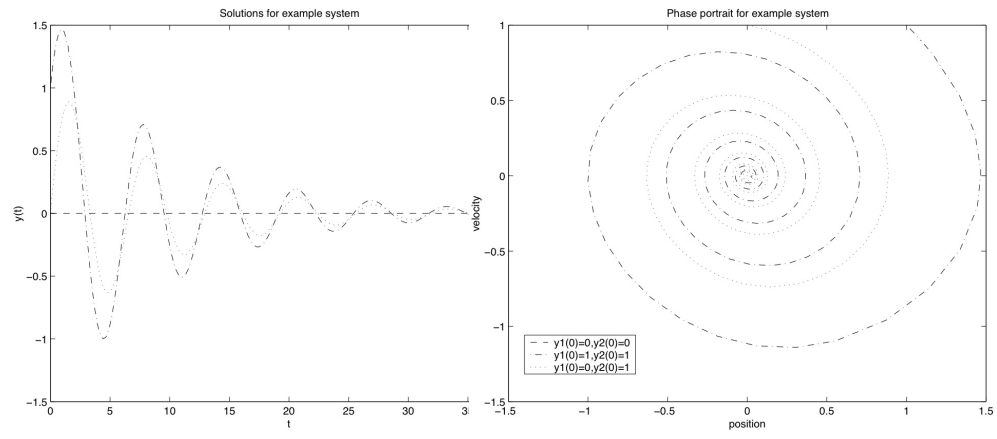


Figure 1: Some solution trajectories for the sample system.

Figure 2: A phase portrait for several initial conditions.