APPM 2460
SOLVING SYSTEMS OF EQUATIONS

1. Introduction

Today, we will be implementing a "pumped up" version of Euler's method–Matlab's built in ode solve `ode45`. We can use `ode45` in either of these cases; we just need to learn how to adapt ourselves. First let's review the relationship between second order ODEs and systems.

2. Solving a Second-Order ODE with `ode45`

We will now go over how to solve higher order differential equations using Matlab. Let's consider the initial value problem for an undamped oscillator

$$x'' + x = 0, \quad x(0) = 1, \quad x'(0) = 1. \tag{1}$$

Some calculations show that the solution to the IVP is

$$x(t) = \sin(t) + \cos(t).$$

In order to solve this system in Matlab, we must first convert the equation to a system of 1st order equations by the transformation

$$x_1 = x,$$
$$x_2 = x',$$

so that the equation (1) is equivalent to the first order system

$$x_1' = x_2,$$
$$x_2' = -x_1.$$

We would like to solve this forward in time. To do this, we must first create a function M-File that holds the differential equation. It works exactly how the function M-file works for solving a first-order differential equation, except we must treat our variables (except time) as vectors instead of scalars as we did before. The function M-File for this differential equation should be saved as `system_ex.m` and looks like

```
function xprime = system_ex(t,x)
xprime = zeros(2,1);
xprime(1) = x(2);
xprime(2) = -x(1);
end
```

See how `x` is a vector, where `x(1)` is associated with x_1 and `x(2)` is associated with x_2? The same is true of `xprime`, where `xprime(1)` is associated with x_1 and `xprime(2)` is associated with x_2. That's all there is to it!

Now we'd like to solve the differential equation with initial conditions $x_1(0) = 1$ and $x_2(0) = 1$ forward in time, lets say $t \in [0, 20]$. The command is just the same as we have used before, except we need to give it a vector of initial conditions instead of just a scalar. In a new script, type:

```
[t,x] = ode45(@system_ex,[0,20],[1,1])
```

The system has been numerically solved. Looking in the workspace, you see we now have two variables. t holds all the time steps while y is a matrix with 2 columns. The first column of the matrix is all the x_1 values and the second column is all the x_2 values. You can plot these against time to see the solution of each variable, or plot them against each other to generate solutions in the phase plane:

```
plot(t,x(:,1))
plot(t,x(:,2))
plot(x(:,1),x(:,2))
```

which will result in the plots shown in Figure 2 (more code will need to be written to include axis labels)

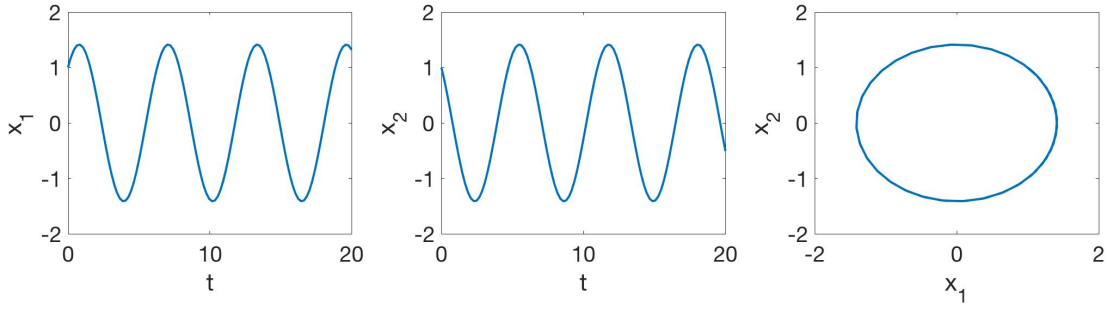

FIGURE 1. Plots for the harmonic oscillator example equation (1).

3. A nonlinear example

Consider the so-called van der Pol equation,

$$x''(t) - x'(t)(1 - x^2(t)) + x(t) = 0, \quad x(0) = 2, \quad x'(0) = 0. \tag{2}$$

This is a second-order ODE, but we will now recast it as a *system* of first-order ODEs. We'll use x_1 to denote x and x_2 to denote $x'(t)$. Then, we can write (2) as

$$\begin{aligned} x_1'(t) &= x_2(t) \\ x_2'(t) &= x_2(t)(1 - x^2(t)) - x(t) \end{aligned}$$

where $x(0) = 2$ and $x_2(0) = 0$. We thus see that second-order ODEs (and indeed, any order ODE) can be rewritten as a system of first-order ODEs. This first-order form is what we need in order to use `ode45`. Note that we could treat the two elements of our system (x_1 and x_2) as two elements of a vector, that is to say our vector x would be

$$x(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} \tag{3}$$

So in this case, $x = x_1$ and $v = x_2$. Note that $x'(t) = (x_1'(t), x_2'(t))$. We could then write our ODE as a **vector-valued** function, i.e.

$$x'(t) = f(t, x) \quad \text{where} \quad f(t, x) = \begin{pmatrix} x_2 \\ x_2(1 - x_1^2) - x_1 \end{pmatrix} \tag{4}$$

To summarize: we have rewritten the ODE (2) in the form $x'(t) = f(t, x)$, but now x is a vector, with $x = (x_1, x_2)$. With our ODE in this form, we can put it into MATLAB.

3.1. Inputting the ODE into MATLAB.

Now, we must figure out how to input this into MATLAB. Our first task is to write a function `f(t,x)` that takes in a time t (i.e. a number) and a vector y and outputs another vector $f(t, x)$. Well, we can do this by modifying the code for the previous example:

```
function xprime = odefun(t,x)
xprime = zeros(2,1); % initialize xprime as a column vector of length 2
xprime(1) = x(2);
xprime(2) = x(2)*(1-x(1)^2) - x(1);
```

The above function generates an output `xprime` that is a 2×1 column vector. We can now call ode45 like so:

```
x0 = [2 0]; % since y = (x,v) and x(0)=2, x_2(0)=0
tspan = [0 20]; % time runs from 0 to 20
[t,x] = ode45(@odefun,tspan,x0);
plot(t,x(:,1),t,x(:,2)); % plot the two columns of the result
```

We have plotted both $x_1(t) = x_1(t)$ and $x_2(t) = x'(t)$ against time. Are there other ways to visualize solutions to systems of ODEs? Next we will look at another way of plotting the result.

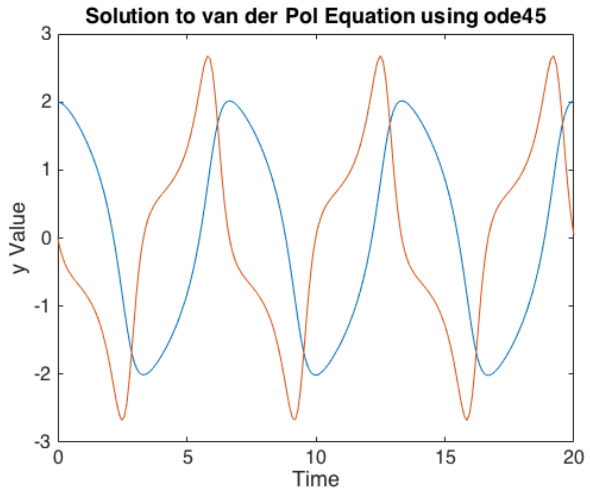

FIGURE 2. Solution to the van der Pol equation, using supplied code.

4. Working in Phase Space

Physicists often like to visualize systems in what they call **phase space**. Recall that we have a vector that depends on time, $x(t) = (x_1(t), x_2(t))$. To plot in phase space is simply to plot the parametric curve given by $(x(t), x_2(t))$. We can modify the above code to plot in phase space as follows:

```
x0 = [2 0]; % since x = (x1,x2) and x1(0)=2, x2(0)=0
tspan = [0 20]; % time runs from 0 to 20
[t,x] = ode45(@odefun,tspan,x0);
plot(x(:,1),x(:,2)); % plot the two columns of the result AGAINST ONE ANOTHER
```

The resulting figure is shown in Figure 3.

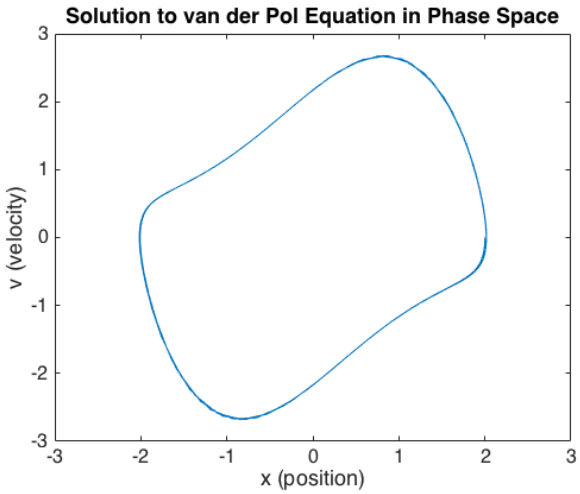

FIGURE 3. Phase space visualization of van der Pol solution.

5. HOMEWORK

Write a script that finds a numerical solution to the second-order IVP

$$x(t)'' - \mu(1 - x^2(t))x'(t) + x(t) - A\sin(\omega t) = 0$$

using parameters $\mu = 8.53$, $A = 1.2$, and $\omega = 2\pi/10$. Use the initial conditions $x(0) = 2$ and $x'(0) = 0$. (Note that the above equation reduces to the van der Pol equation we solved above if we set $\mu = 1$ and $A = 0$. Adding in nonzero A adds **forcing** to the system, i.e. makes it non-homogeneous.) Plot for times $t = 0$ to $t = 100$. Include both a plot of x and v against time, as well as a phase space plot of your solutions. Plots should be appropriately labelled and titled.

If you solve the system correctly, you will see some very strange behavior! In particular, there will be a pattern that almost repeats itself, but not exactly. This is an example of **chaotic** dynamics, which we will explore more thoroughly in a few weeks.