LAB 2: FIRST ORDER SYSTEMS AND THE VAN DER POL OSCILLATOR, PART A

1. MATLAB

MATLAB commands we use in this lab are below, and include an add-on function `dirfieldsys2`. Excepting `dirfieldsys2`, these are the same commands that we have used before, but we will be using them now with systems and their (vector) solutions; here we introduce them in this context. The lab assignment follows this section.

1.1. **dirfieldsys2.** This isn't a native *MATLAB* command; download it from the labs page. The command `dirfieldsys2` plots the direction field for a system of two first order equations $x' = f(x, y)$, $y' = g(x, y)$. The command is

```
>> dirfieldsys2(f_handle, g_handle, [xmin, xmax], [ymin, ymax])
```

for example, to plot the direction field for the system $x' = x\,y$, $y' = \sin(x + y)$ on the domain $x \in [-3, 3]$, $y \in [0, 5]$, we would use

```
>> dirfieldsys2(@(x,y) x.*y, @(x,y) sin(x+y), [-3,3], [0,5])
```

We can also define function handles first and then use them in `dirfieldsys2`:

```
>> fxy = @(x,y) x.*y;
>> gxy = @(x,y) sin(x+y);
>> dirfieldsys2( fxy, gxy, [-3, 3], [0, 5] )
```

(**Note**: dirfieldsys2 will interpret *all* operations in the input function handles to be pointwise, that is, will interpret both `x*y` and `x.*y` as `x.*y`. This should be what you want, and you should be able to ignore this comment.)

1.2. **ode45.** We've used `ode45` to solve first-order initial value problems of the form $x' = f(t, x)$, $x(0) = x_0$; the command is `[t,x] = ode45(@f(t,x)..., [t0 t1] x0)`, where the `...` give the definition of the function on the right-hand side. To solve a system of equations, we call `ode45` with a vector `x` and vector `f`. For example, suppose we want to solve the initial value problem $x' = y$, $y' = x\sin(ty)$, with $x(0) = 1$, $y(0) = 0$. In vector form this is $\mathbf{x}' = \begin{pmatrix} x \\ y \end{pmatrix}' = \begin{pmatrix} y \\ x\sin(t\,y) \end{pmatrix} = \mathbf{f}(t, x, y)$, with $\mathbf{x}(0) = \begin{pmatrix} x(0) \\ y(0) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. Note that the first component of the vector $\mathbf{x}$ is the original variable x, and the second component is y. Thus, using `ode45` we will have a vector `x` where `x(1)` is x and `x(2)` is y. Thus, to solve the system we use

```
>> [tsol,xsol] = ode45( @(t,x) [x(2); x(1)*sin(t*x(2))],...
          [0 5], [1; 0]);
```

(the elipses, `...`, are just to break the line here), or, defining the function on the right-hand side first,

```
>> f = @(t,x) [x(2); x(1)*sin(t*x(2))];
```

```
>> [tsol,xsol] = ode45( f, [0 5], [1; 0]);
```

The final thing to note is that `xsol` now has two columns: the first column gives the x values at the t values in `tsol`, and the second gives the y values. Thus

```
>> xsol(3,2)
```

returns the value of y at the third time step.

1.3. **ode15s.** This is another numerical solver for differential equations, and takes exactly the same arguments as `ode45`. We will want to use it, in general, when we're solving problems which have solutions that change rapidly in some portions of the domain on which we're solving. This is discussed below.

1.4. **plot.** We use `plot` exactly as we did before. Note, however, that if we had generated a solution to a system using `ode45` as above, we could then plot the trajectory in the phase plane with

```
>> plot( xsol(:,1), xsol(:,2) );
```

(which should make sense—the `:` just gives all values along that index, so we're just giving a vector of x values, and then a vector of y values).

1.5. **axis.** The `axis` command sets the range for the x and y axes of a plot:

```
>> axis( [ -3 3 -2 2 ] );
```

sets the axis scale for the current plot to $-3 \leq x \leq 3$ and $-2 \leq y \leq 2$.

2. Part A

Exercises in this section are to be completed by pairs. They will allow you to complete Part B of the lab, and will provide information for your lab writeup. At the end of Workday 1, pairs should present their solutions to Exercises 2 and 3 to each other. Note that material from Part A appears in one of your written homework problems and will be relevant for Part B.

3. Background

In this lab we are considering the van der Pol oscillator, a model of an active RLC circuit with a nonlinear resistor that dissipates energy when the amplitude of the current is high, and pumps energy into the system whenever the amplitude of the current is too low. The resulting differential equation we are using is

$$x'' + \mu(x^2 - 1)x' + x = 0, \tag{1}$$

which we are calling "the" van der Pol equation. In Exercise 2 in the prelab you wrote this as a system of equations, and at the end of the prelab we linearized the system to obtain the linearized version of the equation,

$$\begin{aligned} x' &= y \\ y' &= -x + \mu y. \end{aligned} \tag{2}$$

In the following we look at the nonlinear and linear systems, and explore their solutions, similarities and differences.

Before starting work, download the file `dirfieldsys2.m` (described above) from the lab page, and check with your partner to confirm that you have the same system for the van der Pol equation.

4. Reflection

For this lab you are not submitting a formal lab writeup. Instead, you will submit a shorter "reflection" at the end of Part B of the lab. In this you will be considering the three questions

(a) How is the linearized system obtained from the nonlinear van der Pol system? What does this tell you about what it should be able to tell you about the behavior of the nonlinear system?
(b) How is the description of the nonlinear resister in the Prelab and background sections of Parts A and B reflected in the phase portraits that you generated in this lab?
(c) In the van der Pol equation, x is a current in the circuit. What will a graph of x look like as a function of time, t, given your phase portraits for the linear and nonlinear systems?

As you work through the following you may wish to keep these questions in mind.

5. Part A Exercises

All exercises are to be completed with $\mu = 1$.

Exercise 1. Plot the direction field for the nonlinear van der Pol system you obtained in the prelab. Save one copy of the plot with the x- and y-axes restricted to the interval $[-1, 1]$ and a second copy restricted to $[-3, 3]$. On a second graph, plot the direction field for the linearization of the van der Pol system, (2), with the x- and y-axes restricted to the interval $[-3, 3]$. What information does the direction field for the linearized system give you about the nonlinear system? (That is, where are the direction fields for the nonlinear system similar to and different from that for the linearized system? Why does this make sense?)

Exercise 2. Pick an initial condition (x_0, y_0) close to the origin and find a solution to the van der Pol system with `ode45`. Plot this solution trajectory in the phase plane. Note that this tells you the direction of motion along the direction fields that you found in Exercise 1. How is the behavior of the trajectory consistent or inconsistent with the linear and nonlinear direction fields that you obtained in exercise 1? Note that if you extend your solution for large enough t, the positive x-intercepts of this trajectory converge to some value x^*. Estimate this value from your plot.

After you have found x^*, add a solution trajectory from an initial condition far from the origin to your graph. Use a different line type or color so that you can distinguish this from your first trajectory. How does it behave?

If the "possibility" of crossing trajectories isn't obvious, think about the existence and uniqueness theorems, Theorems 3.6.1 and 2.4.2 in [BB, §§3.6,2.4], and Remark 2 that follows Theorem 2.4.2. These say something about whether solutions can cross.

Exercise 3. One way to find the value of x^* would be to zoom in on the point where the trajectories cross the x-axis. Do this on your plot from exercise 2, either by using the zoom tool in the plot window or by using the `axis` command. Your x-range should be something on the order of $x^* \pm 0.01$. Do the trajectories cross? Is that really possible?

What we're seeing here is an issue of the precision of the approximate solution generated by `ode45`. Graph the same trajectories, but generate them with `ode15s` instead of `ode45`—`ode15s` is another numerical solver that generates approximate solutions to differential equations, but it deals better with rapidly changing solutions (in particular, with places like the corners in the nonlinear trajectories that we see in Exercise 2). With this, are the trajectories more properly separated?

References

[BB] Brannan, James R, and William E Boyce. Differential Equations: an Introduction to Modern Methods And Applications. Third edition. Hoboken, NJ: Wiley, 2015.