LAB 0: A MATLAB (RE-)INTRODUCTION

1. Model and Objectives

1.1. **Model.** Consider a runner in a short race. Their velocity v (in meters/s) is a function of time t (in seconds), and we propose that it may be modeled by the initial value problem

$$\frac{dv}{dt} = A - \frac{1}{\tau} v, \quad \text{with} \quad v(0) = 0.$$

Here A is a constant giving the sprinter's initial (maximum) acceleration, and τ is a constant that measures the rate at which the sprinter tires. With a little work, we can find A and τ to approximate Usain Bolt's (2008) world-record 100 meter run (9.69 sec) with the equation

(1) $$\frac{dv}{dt} = 8.730 - 0.721v, \quad v(0) = 0.$$

If you don't remember, or aren't yet comfortable with differential equations, don't worry! The three things to know about differential equations are: (1) they are equations, (2) they involve a dependent (v) and independent (t) variable—and derivatives of the dependent variable ($\frac{dv}{dt}$), and (3) our goal is to find a function $v(t)$ that satisfies the equation.

1.2. **Objectives.** In this lab our goals are to see how to use *MATLAB* to gain insight on what differential equations and their solutions mean, and conceptually what they may be telling us. In particular, today we want to learn

- how *MATLAB* thinks about the world (in particular, that everything is an array, or vector),
- how to save and edit *MATLAB* scripts and run them in *MATLAB*,
- how to plot functions in *MATLAB*, and
- how to solve differential equations numerically using *MATLAB*.

What do these have to do with differential equations? Recall that when we think about mathematical expressions and concepts we often try looking at them from multiple viewpoints: as symbols, as graphs, by looking at numerical values, or by describing their behavior or what that tells us about something that they model. What we do in this (and subsequent) labs focuses in these ideas, especially on the graphical representation and modeling. We will find that *MATLAB* allows us to see what solutions to differential equations are doing (often without having to solve them), and to consider meaningful models of the real world, which is both a core focus of this course and a characteristic of differential equations generally.

2. Lab Instructions

This lab is different from the others you will complete this semester, in that it has no pre-lab, and you will work in pairs to complete the lab in one class. Furthermore, *the primary goal of this lab is for you to gain (a little) familiarity with MATLAB*. The only way this will happen is if you are actually working with *MATLAB*. To do this, work in the following manner:

(1) Work with a partner, at one of the two computers at the table.
(2) As you work, collaborate with the other pair at the table to be sure that you all understand what you're getting and how to make *MATLAB* do what you want.
(3) If you are typing at the keyboard, *make sure you are checking with your partner to be sure that you both understand what you are typing and why*. A key goal of this lab is for you both to then be comfortable working with *MATLAB*, which will be accomplished best if you are both thinking about what you are doing.
(4) If you are not typing at the keyboard, *check with the person who is typing to be sure that you both understand what is being typed and why*.
(5) *Change who is typing at the computer every 15 minutes.*

For this lab, when you have completed it you will submit a "published" version of the *MATLAB* command file that you used to solve the lab exercises.

3. *MATLAB*'s world-view

If you haven't already, start *MATLAB* on the computers at your table. The fail-safe way to do this on a Mac is to use Searchlight: search for "matlab," and run the first option (alternately, there may be a link in the Dock, or you can also look under "Applications" in the Finder. If you're on a PC, look under "all applications" in the Windows menu. If you're on a Linux computer, type "matlab" at a terminal prompt.

Example 1*:* Notice the four windows that are open in the *MATLAB* frame. The most obvious of these is the *Command Window*. Type the following command at the >> prompt:

```
>> 5*sin(3)
```

What changes in the *Workspace* window? Then type each of

```
>> v = [1; 3; 0]
>> A = [1, 2; 0, 5]
>> 5 sin(3)
```

What happens when you type `5 sin(3)`? Why?

Example 2*:* Note that the formatting of *MATLAB*'s output for `v` and `A` suggests that these are vectors and matrices. To test this, type the following:

```
>> ans(1)
>> v(3)
>> A(1,2)
```

Are the results what you expect? Why or why not?

These first two examples illustrate four very important observations about how *MATLAB* views the world.

Important MATLAB information

(1) *MATLAB* requires operators: `*` for multiplication, etc.
(2) In *MATLAB*, every variable is a vector (array), or matrix.
(3) You can get components of one of these vectors by indexing them with an integer index that indexes from one: `v(3)`, or `A(1,2)`.
(4) *MATLAB* provides a list of all defined variables in the current session in the *Workspace* window. The variable `ans` is used for any unnamed calculation. Note that it gets overwritten with the next unnamed calculation.

4. Plotting in *MATLAB*

To work the remainder of the lab, we will edit a *MATLAB* script. Click "New Script" (at the top left of the *MATLAB* frame). This will open a script editor. A few notes about this before we go on:

(1) The percent sign (%) is a comment character in *MATLAB* (it will ignore everything to the right of a %). Insert comments or discussion in the file by putting a percent and then typing your comment. Putting a double comment, %% is useful for formatting the file; *MATLAB* interprets that as the start of a new section. The comment after the double percent is the title of the section, and any comments immediately below the section title will be typese. Create a section title comment at the top of the file (e.g., "Lab 0 by Your Names").
(2) Save your work. Click "Save as" and save the file with an appropriate name like "`lab0_last1_last2.m`," where `last1,2` are your last names.

As you work the following exercises, indicate which problem you are working with a comment that puts it into a new section, e.g.,

```
%% Exercise 1
```

And answer the questions in the exercise by putting comments in the file as you go. If you put the comments immediately after the section title these will be typeset nicely. For lab 0, you will submit a published version of this script file at the end of the lab as your lab report.

Remember that the person at the keyboard should be getting feedback from their partner, not just typing.

In Exercise 1 we plot the solution to the differential equation model we're considering here, which turns out to be

$$v(t) = 12.110(1 - e^{-0.721t}).$$

(We'll see how to find this, later, in lecture.) Because *MATLAB* fixates on vectors you might expect that the appropriate command would be something like `plot( tvector, vvector)`. It turns out that you would be right.

Exercise 1: Define a vector of t values; recalling that Bolt's run was just shy of 10 sec, define
`t = [ 0 1 2 3 4 5 6 7 8 9 10 ];`
(what is the effect of the appended semicolon here?) and then try an appropriate plot command:
`plot( t, 12.11*(1 - exp(-0.721*t)) )`
save your file and click the "Run" button at the top of the editor window. In your script file, answer the following two questions in comments:
(a) What does the model predict for Bolt's maximum speed? In what units? Convert this to miles or kilometers per hour (by multiplying by 2.236 or 3.6,[1] respectively). How crazy fast is this?
(b) Why is the plot slightly jagged? *(Hint: To answer this, try running the command* `plot(t, 12.11*(1-exp(-0.721*t)), 'o')`*)*

In *MATLAB*, a colon, `:` is an operator that generates a vector of values: the definition of the vector `t` we used above is equivalent to any of the following: `t = [ 0:1:10 ]`, or, `t = 0:1:10`, or, more simply, `t = 0:10`. The middle argument is the stepsize, which defaults to one, and the `:` operator returns something *MATLAB* thinks is a vector whether you enclose it in brackets or not.

Exercise 2: Plot the solution to the differential equation with a t vector of 1000 points. Instead of the final argument `'o'` that you used at the end of Exercise 1, try each of `'-k'`, `'--r'`, `'-.g'` and `':c'`. Label the x and y axes of your plot and add a title:

```
xlabel('time');
ylabel('velocity');
title('Usain Bolt''s modeled velocity, 2008');
```

Note what `''` does: it's an escaped apostrophe, so that you can include an apostrophe in labeling text.

Remember to label this exercise with a comment, and also in comments indicate what the plot arguments (`'--r'`, etc.) you used do.

Exercise 3: Finally, let's add the graph of the asymptote (Bolt's top speed) to your graph. *Note: by default the last plot command overwrites the existing axes and all labeling!* To avoid this, you need to turn on "hold" to keep adding to the graph:
`hold on`
Note, however, that once you have hold on for the existing axes you to

[1] Why 2.236 or 3.6? You might want to check that these numbers make sense.

accumulate all graphs there. To turn this off later, use the command
`hold off`
To add the asymptote, figure out Bolt's top speed v_m, e.g., by evaluating the velocity function at $t = 10$; you can then just plot the line from $(0, v_m)$ to $(10, v_m)$ with something like look at the *Workspace* window to see how large your t vector is; then, if `k` is Bolt's top speed and `L` the length of the t vector, add a plot of
```
plot( [0 10], [v_m, v_m], '--b' )
```
(where `v_m` is the value you determined). Another useful labeling feature is "legend"; try adding a legend with
```
legend( 'velocity', 'maximum', 'Location', 'southeast' );
```
Play with your script until you have (only) a labeled v vs. t graph of Bolt's velocity and his maximum velocity, with a legend.

Notice that when you click "Run" in your script, *MATLAB* is running it by inserting the name of the lab file at the command prompt (>>). You can run any script file you see in the *Current Folder* window the *MATLAB* window frame in this way.

5. Differential Equations and *MATLAB*

Our last goal of this lab is to use *MATLAB*'s differential equation solver, `ode45`, to generate an approximate—numerical—solution to the model that we're looking at.

Exercise 4: Suppose we want to solve the initial value problem given in (1), $v' = 8.730 - 0.721v$, $v(0) = 0$. Type
```
[ t1, v1 ] = ode45( @(t,v) 8.730 - 0.721*v, [ 0, 10 ], 0 );
```
Plot the `v1` as a function of `t1` with the plotstyle argument `'ok'`. What do you notice about the spacing of the elements of the vectors `t1` and `v1`? Add a legend and label your axes.

More Important MATLAB information

(5) *MATLAB*'s `ode45` command generates vectors of values for the *independent* (t; saved in `tx` in the command below) and (a numerical approximation for the) *dependent* (v; saved in `vx` in the command below) variables in a differential equation. The command is, for an equation $\frac{dv}{dt} = \text{RHS}$,
```
    >> [tx, vx] = ode45( @(t,v) RHS, [t0,t1], v0 );
```
where `RHS` is the right-hand side of the differential equation, `t0` and `t1` are the minimum and maximum values of the independent variable t, and `v0` is the initial condition for the dependent variable v.

The `@(t,v) 8.730 - 0.721*v` expression is actually generating a *function handle*: a pointer to a(n otherwise anonymous) function. We can think of the `@(t,v)` as telling *MATLAB* that we're defining a function of two variables, t and v, and subsequent expression tells *MATLAB* how to evaluate the function.

Exercise 5: Define the function handle you used in the previous example explicitly, by typing
`rhs = @(t,v) 8.730 - 0.721*v`
What output do you get when you type
`rhs(0,0)` or `rhs(9.69,12.11)`? What about `rhs(1.12,6.69)`? What do these tell you about v? *(Hint: what is the function calculating?)*

Note that the `@` operator allows us to define a new function in *MATLAB*. When we did this in the `ode45` command the function was anonymous (we never called it anything); in this last exercise we named the function we created. We will use this ability to create function handles many times this semester!

6. Finishing up

That's it! When you're done, **be sure that the names and UM ID numbers of you and your partner are at the top of the script file**. Review the list of important *MATLAB* rules and tricks above, and the additional three items below. Then click the "publish" tab in the script editor. Click the arrow below the "Publish" button and "Edit publishing options." For "Output file format," select "pdf," and check that you know what the output file folder is (because that's where the published lab report will go). Then click "publish." Submit your published lab report in Canvas.

More Important MATLAB information

(6) Consistent with *MATLAB*'s world-view, functions and operators are vectorized: thus, if `t` is a vector of values, `sin(t)` returns a vector of the sine of those values. Similarly, operators such as `*` and `^` will perform *matrix multiplication and exponentiation*. To apply these element-by-element, which is what we will want to do most of the time in this course, prepend a period: thus `t.^2` is a vector with each element of the vector `t` squared.

(7) To pass a function to another function, use a function handle, for example, `@sin`. We can define an anonymous function as `@(t) t^2` (a function that squares every element of the input), or, if it is a function of two variables, `@(t,v) 8.730 - 0.721*v` (or similar).

(8) We can run any *MATLAB* script file by referring to it by name in our scriptfile, or by typing it at the command prompt (>>). We will do this in many labs to avoid having to retype commonly used functions many times.

7. References and resources

(1) Keller, J.B. A Theory of Competitive Running. *Physics Today*, Sep 1973, p.43.

(2) Alexandrov, I. and P. Lucht. Physics of Sprinting. *American J. Physics* **49**, 1977.

(3) Pritchard, W.G. Mathematical Models of Running, *SIAM Review* **35**:359-379, Sep 1992.

(4) Thompson, W.J. *Computing in Applied Science*. New Jersey:Wiley, 1994, pp.107-109.

(5) Dunbar, S. The ODE of World Class Sprints. *CODEE Newsletter* Spring 1994, pp.7-9. On-line at `http://www.codee.org/library/newsletters/Spring%201994`. Retrieved on: 9 Jan, 2013.

(6) Lee, J. (August 22, 2008) Usain Bolt 100m 10 meter Splits and Speed Endurance. *SpeedEndurance.com*. Retrieved on: 9 Jan 2013.