

# LAB 0 — MATLAB INTRODUCTION

## Before lab:

It is recommended that you do the second problem on the homework before coming to lab, although that won't be required or collected in lab. It is also recommended that you read the *Introduction to Using MATLAB for Systems Calculations*.

## Objectives:

- 1) Gain proficiency with MATLAB.
- 2) Become comfortable with the two things that will cause confusion repeatedly throughout the course unless you take the time to understand them thoroughly now:
  - a) The differences between different kinds of matrix and vector multiplication in MATLAB.
  - b) The use of functions, function arguments and function names.

## Write-up:

None, but all students should check with the TA that their functions operate satisfactorily before they leave the lab.

## Time allocation (an approximate guideline):

Introduction to the lab, including accessing MATLAB for this lab – 15 minutes

Part 1. Increase familiarity with MATLAB – 1 hour

Part 2. Using functions – 1 hour

## Supplementary materials:

*Using MATLAB for Systems Calculations*. This provides further background on MATLAB beyond that in this handout.

*MATLAB Quick Reference*. A list of the most important commands you'll need for 22, with very brief descriptions.

*Introduction to Unix for EngS 22*. This provides information needed to access MATLAB via the UNIX operating system, which is optional for the purpose of ENGS 22.

Web resources, linked from the *Information on Computers and MATLAB* section of the information section of the ENGS 22 site on <http://blackboard.dartmouth.edu>. (A quick link to the 22 site is <http://tinyurl.com/dse2>).

## 1. Increasing Familiarity with MATLAB

Notes: This handout is not a comprehensive tutorial—you will need to read the handout “Using MATLAB for Systems Calculations” as well, and use the MATLAB help function or on-line documentation.

### 0. Accessing MATLAB.

Starting MATLAB—on a Windows Workstation in Lab 222. Step-by-step instructions are on the next page

- **Details here may change as they are presently updating the computers in 222:**  
Log in, using the Thayer username and password assigned to you, NOT your blitz/DND name and password. Set the domain to “Log on to: Cummings”.
  - If you haven’t yet, you’ll want to change your password. The easy way to do that is on the Web at <http://thayer.dartmouth.edu/other/admin/change.shtml>
  - Click the MATLAB Icon on the desktop or Click ‘Start’ on the taskbar then move the cursor over ‘Programs’ on the Start Menu then ‘Matlab6.5’ on the Programs Menu then ‘Matlab6.5’ application in the Matlab 6.5 Menu.
1. Practice using MATLAB as a calculator. Use a problem of your choice, or calculate the number of hours in a year. Try using the up arrow key to change your calculation.
  2. Create these vectors,

$$\mathbf{a} = [1, 2, -1], \quad \mathbf{b} = [3, -1, 1], \quad \mathbf{c} = [0, 1, 1]$$

a) Type  $\mathbf{a}$  and  $\mathbf{a}'$  to see the vector  $\mathbf{a}$  and its transpose. If you forget whether a vector is a row or a column, this is a good quick way to check.

b) Try asking MATLAB to perform some calculations on those vectors with the following:

$$\begin{array}{lll} \mathbf{a} + \mathbf{b}, & \mathbf{a} * \mathbf{b}', & \mathbf{b}' * \mathbf{a} \\ \mathbf{b} .* \mathbf{a} & \mathbf{b} * \mathbf{a} & \end{array}$$

Make sure the results make sense to you—in particular, understand the difference between the different multiplications. See the appendix on visualizing matrix multiplication (last page of this document) if you are rusty on how matrix multiplication works. Taking the time to understand this now is well worth it—confusion about these issues will cause you trouble repeatedly in this course. You’ll tear out your hair again and again. Get out a pencil and paper and try the calculations yourself to make sure you are right about what MATLAB is doing. **If it doesn’t make sense, don’t go on—stop and ask for help.**

3. Because MATLAB is vector-oriented, most loops can be avoided.

(a) Plot  $y = x^2$  vs  $x$  for 7 points between  $x = -1$  and  $1$ , inclusive. This can be accomplished with the following commands:

```
x = linspace(-1, 1, 7)
y = x.^2
plot(x, y)
```

Note that  $\wedge$  raises each element of  $x$  to a power. ( $x^2$  would try to multiply the vector  $x$  by itself, but the dimensions don’t work out if you do that—it would need to be  $x'*x$  or  $x*x'$ .)

It is worth pausing a moment to recall how this might be done in C++:

```
const int npts = 7;
int n;
real x[npts], y[npts], dx;

dx = 2.0/(npts-1);
x[0] = -1;
for(n=0; n<npts; n++) {
    x[n] = x[0] + n*dx;
    y[n] = pow(x[n], 2);
}
/* And now you can insert your plotting commands */
```

The MATLAB version is considerably simpler!

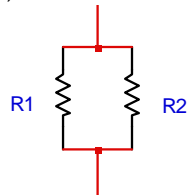
(b) Repeat the plot, but with better resolution—use 100 points instead of six. You can use the up arrow to avoid a lot of retyping. Also, you don't really want to see all 100 numbers on your screen. You can suppress the screen display of the results of any line by putting a semicolon after it. So your commands look like this:

```
x = linspace(-1, 1, 100);
y = x.^2;
plot(x, y);
```

4. Plot the function  $y = t e^{-2t}$  vs.  $t$  ( $t = 0$ ). (Hint: type 'help exp'). You can use much the same approach as was used above. Use as many points as you need to produce a satisfactory graph, and select a maximum value for  $t$  which is sufficient to display the most interesting portion of the graph. Add axis labels (xlabel, ylabel) and a descriptive title (title), which includes your name. (Use MATLAB help to find out more about any of these commands.) Make a hardcopy of your graph.
5. Calculate 100 samples each of the functions  $y_1 = \cos(2\pi ft)$  and  $y_2 = \sin(2\pi ft)$ , where  $f = 1$  Hz and  $\Delta t = .01$  sec.
  - (a) Plot these functions vs.  $t$ , on the same graph.
  - (b) Do it again, but put the two functions on separate graphs in the same figure window (use subplot—see the other handouts, or type help subplot).
  - (c) Plot  $y_2$  vs  $y_1$ . Use the command 'axis square' to make a square plotting box. What do you observe?

Now you'll learn how to create and use an M-file.

6. Something you need to calculate often for electrical systems is the resistance of a parallel combination of resistors. As you'll see later in the class, this same calculation is useful for mechanical, thermal, and fluid systems as well. For two resistors in parallel, like this:



the net resistance is

$$R_P = \frac{R_1 R_2}{R_1 + R_2}$$

You could use MATLAB to calculate this as follows:

```
% SET UP THE RESISTANCES
R1 = 1000;           % Resistance in ohms
R2 = 1200;
% CALCULATE THE PARALLEL COMBINATION
Rp = R1*R2/(R1+R2);
% DISPLAY THE RESULT
Rp
```

Type each of these commands in the command window, just to see what happens (You can omit the comments, for simplicity).

7. Open the MATLAB editor (by clicking on the blank document icon), and create a document containing the commands above. Name it something like “rpar.m”.

**Details here may change as they are presently updating the computers in 222:**

Save it to a folder on the “T” drive. The “T” drive is actually the Thayer network file system. Saving your work on the “T” drive rather than on any of the local drives is important for several reasons:

- You can then access your work from any Thayer machine.
- The local drives are cleaned periodically. If you leave your work on a local drive it will get wiped out with no warning, probably at an inconvenient time.
- If you leave coursework on a public machine, you might get inadvertently mixed up in an Honor Principle hearing. If someone uses your work, it might not be clear to the COS who copied from whom.

- . Now, if you type “rpar” in the MATLAB command window, you perform this simple calculation. However, you have to make sure the MATLAB knows to look on the T drive for your work—if it doesn’t, see step 10 on the last page, or ask a TA for help.

You can change the values of  $R1$  and  $R2$  in the file, save it again, and run it again to calculate different values. But that’s a clumsy way to do it: better is to write a function.

8. I always used to wish my calculator had a “||” button on it to calculate the parallel combinations of resistors. Editing the file is one way to do that, we can make MATLAB much more like that imaginary calculator, by creating a parallel resistor function. To make an M-file into a function, it needs a line like this:

```
function output = foo(input1, input2)
```

The italicized words are ones that you replace with your own names. `foo` is the name of the function and should be the same as the name of the file that has the function in it (`foo.m`). There can be any number of inputs (separated by commas), and `output` is anything you want to pass back to the calling program (or the command window). See the “MATLAB Programming” section of the “Using MATLAB for Systems Calculations” handout for more information and an example.

Modify your M-file to make it a function. When you’re done, you should be able to get the parallel combination of two resistors by typing

```
Rcombo = rpar(800, 500)
```

where 800 and 500 are the two resistances you want to combine. You’ve now got that function built into MATLAB!

9. To see how the variable names work, try doing the following in the command window:

```
x = 800;
y = 500;
z = rpar(x, y)
```

Notice that this works even though you (presumably) used a different name for the inputs inside your function. Those variables are local, within your function. The fact that  $R1$  gets set to  $x$  is determined by that being the first variable in the list, not by the variable name.

10. Now you will make a more sophisticated function that can calculate the combination of three resistors, and can do it either for resistors in series or for resistors in parallel. Here are some hints on each aspect of that task. The skills you will learn in this exercise will be essential for using the `ode45` function you will be using extensively on the next homework.

a. How to calculate three resistors in parallel. You may know a formula for that, or know how to calculate it, but you don't need to. Why? Because if we have three resistors in parallel, and start by combining just two of them in parallel, we will then have two resistors in parallel, and we not only know how to calculate that, but we've got a function to do the work for us. So we can just do something like:

```
first_two = rpar( r1, r2 )
all_three_in_parallel = rpar( first_two, r3 )
```

Or you can do it in just one line:

```
all_three_in_parallel = rpar( rpar( r1, r2), r3);
```

b. To be able to do this with series resistors too, you might want to make a series resistor function, `rser.m`. It could be for just two resistors.

c. You'll need to be able to tell your do-it-all function whether the resistors are in series or in parallel. To do this, you can use one of your input arguments to pass the name the function you want to use (either `rser` or `rpar`). That name is not a numerical value but a string. In MATLAB, you can put strings into variables almost the same way you put numerical values in:

```
foo = 'hello'
```

will make the variable `foo` hold the string "hello". Then, if you were to call a function using the variable `foo`, the string "hello" would get passed into the function. For example, the function `disp` will display a variable, so if you were to type

```
disp(foo)
```

or

```
disp('hello')
```

either one would display the word "hello".

To check your understanding, see if you can figure out what would happen if you typed

```
foo = 'goodbye'
disp('foo')
```

and see if you get what you expect. If you don't, check with a TA for help understanding it.

d. Suppose you have the name of a function in a variable (e.g. `foobar = 'sin'`) and you want to execute that function. If you just type `foobar(pi/4)` Matlab will think you want the  $\pi/4^{\text{th}}$  element of the vector `foobar` and will give you an error messages. You need to tell Matlab to execute the function named by the string in your variable. You can do that with the built-in MATLAB function called `feval`, like this:

```
feval(foobar,pi/4)
```

e. Now you should have all the pieces you need to put together a function that will take an input like

```
fancyfunction('rpar', 500, 600, 700)
```

and calculate the parallel combination of the three resistors (or the series combination if you use `rser` as the first argument).

Once you get the function working, demonstrate it to a TA.

**10. Setting the MATLAB path. (Optional)**

The MATLAB search path is a list of directories (folders) where MATLAB looks for functions. You can add other directories to this path by opening the path browser (“Set Path” under the “File” menu). This will be useful if you want to set up more directories to keep track of different assignments.

**13. Quitting (Not Optional):**

13.1 Exit MATLAB (type “exit” or select it from the file menu)

13.2 Log off the computer to keep your files on the T: drive safe.

**Appendix: Visualizing Matrix Multiplication**

I like to write matrix products, such as  $\mathbf{AB} = \mathbf{C}$  with the matrices positioned like this:

$$\begin{array}{ccc} & & \mathbf{B} \\ & & \left[ \begin{array}{cc} e & f \\ g & h \end{array} \right] \\ \mathbf{A} & \mathbf{C} & \\ \left[ \begin{array}{cc} a & b \\ c & d \end{array} \right] & = & \left[ \begin{array}{cc} & \\ & \end{array} \right] \end{array}$$

This makes it easy to visually see both which elements get multiplied, and where they go in the product, and it makes it easy to see what size and shape the product should be. For example, to compute  $\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix}$ ,

I write:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

Then sighting horizontally across the matrix on the left shows you the elements involved in a given element of the product, and sighting vertically down the top matrix shows you the elements of it that are involved.

$$\begin{array}{ccc} & & \begin{bmatrix} e & f \\ g & h \end{bmatrix} \\ & \swarrow & \downarrow \\ \begin{bmatrix} a & b \\ c & d \end{bmatrix} & \rightarrow & \begin{bmatrix} ae + bg \\ & \end{bmatrix} \end{array}$$

This also allows you to quickly see that a row times a column gives a scalar:

$$\begin{bmatrix} \circ & \circ & \circ \end{bmatrix} \begin{bmatrix} \circ \\ \circ \\ \circ \end{bmatrix}$$

and that a column times a row gives a matrix:

$$\begin{bmatrix} \circ \\ \circ \\ \circ \end{bmatrix} \begin{bmatrix} \circ & \circ & \circ \\ \circ & \circ & \circ \\ \circ & \circ & \circ \end{bmatrix}$$