APPENDIX A: MATLAB Introduction

Introduction

MathWorks describes their software package MATLAB -

“MATLAB® is the high-level language and interactive environment used by millions of engineers and scientists worldwide. It lets you explore and visualize ideas and collaborate across disciplines including signal and image processing, communications, control systems, and computational finance.”

During the lab you will be using MATLAB to create plots, control and read data from the bench equipment, simple computations, and for the brave as an interface with the Arduino. Let’s start by using this powerful computational resource as a spreadsheet.
The MATLAB development environment – depending on your computer and operating system the windows may be in different places – this is how mine looks on my Mac. The look and placement of the windows is customizable and all are detachable.
Creating Plots in MATLAB

Introduction

MATLAB is an extremely powerful (if inelegant) computational and educational environment as you will come to understand as you use it more. For the ECE 110 lab you will primarily be using it to create easy to interpret graphs, as an automated interface to your bench equipment making tasks like find IV curves much faster, and as an interface to the Arduino microcontroller. This appendix introduces step-by-step how to generate graphs using two of the simpler methods.

The first method is to just use the working environment to input data by hand and use MATLAB in the same way you would use Excel – as a spreadsheet. The second method is more commonly used - graphing is usually done using MATLAB functions and variables either by entering them using the command line interface or by entering all of the commands into a script file so you only need to write the commands once.
Using MATLAB as a spreadsheet

**Entering data** *using the workspace window*

Step 1: create a new variable

- Adds a new variable
- Name the variable IVcurve and hit return
**Entering data** *using the workspace window*

**Step 2: enter data**

Double click on the icon to bring up the variable editor

Enter voltage in col. 1

Enter current in col. 2

**Notes:**
**Data plotting using the workspace window**

**Step 1: choose plot type**

1. In the workspace window make sure the variable IV curve is selected

2. Click the arrow beside the menu option plot(IVcurve) to bring down plot menu

3. Choose the 2-D line graph option
Data plotting *using the workspace window (cont.)* - futzing with the look of the graph

A new window should pop-up with the plot of data in column 1 against the row number (blue line on this graph) and the data in column 2 also against the row number.

This is NOT the graph we want. We want the data in col. 1 to be the x-values and the data in col. 2 to the y-values so we need to change how MATLAB is plotting the data.
Data plotting using the workspace window (cont.) – more futzing

In the “Edit” pull-down menu select “Current Object Properties”...

...and a window opens up underneath the plot.

Hitting the refresh data button re-plots the graph

Change x data source so that the voltage values are on x-axis - IVcurve(:,1) - and the current values are on the y-axis.

NOTE: IVcurve(:,1) indicates the values in the first column of the array IVCurve
Data plotting using the workspace window (cont.)

Now click on the other line and push the delete button...

...to get the I-V plot
Data plotting using the workspace window

Step 4: Add annotations by investigating editing functions. Give the x- and y-axis labels and add a title. Find out how to change the color of the line. Add some gridlines.
Plotting Using MATLAB’s Command Line Interface

As some of you know who have used MATLAB before, there is a much simpler way to create the same graphs by typing commands into the command window. These 6 lines of MATLAB code, entered line-by-line into the command window, creates the same plot generated through manipulating the variable Ivcurve in the workspace.

```
>> voltage=[0,1,2,3,4,5,6,7,8,9,10];
>> current=[0,1,4,9,16,25,36,49,64,81,100];
>> plot(voltage,current)
>> title('fake I-V curve')
>> xlabel('voltage (V)')
>> ylabel('current (A)')
```

The first two statements are used to enter the voltage and current data into two Matlab vector variables. The next 4 statements use pre-defined Matlab functions `plot`, `title`, `xlabel`, and `ylabel` open a window and create basic plot (plot), add a title to the plot (title), and add labels to the x- and y-axes (xlabel, ylabel).
How MATLAB Stores and Interprets Data – Data and Variable Types

Examples of inputting the 4 common data types...

Notice that each variable and its value is entered into the workspace as soon as the command is typed into the command window.
How MATLAB Stores and Interprets Data – Common Data Structures

- **Number**
  - a single stored value is considered a data structure, like a table with 1 row and 1 column

- **Vector**
  - a data structure that holds a list of values is stored as a 1xn data structure with 1 row and n columns

- **Matrix**
  - a general data structure with an arbitrary number of dimensions. *Numbers* and *vectors* are considered matrices

```
>> a=123
   a =
     123
>> size(a)
   ans =
     1 1
>> b=[1,2,3,4,5,6,7,8,9,10]
   b =
     1 2 3 4 5 6 7 8 9 10
>> size(b)
   ans =
     1 10
>> c=[11,12,13;21,22,23;31,32,33]
   c =
     11 12 13
     21 22 23
     31 32 33
>> size(c)
   ans =
     3 3
>> d=magic(15);
>> e=1:100;
>> f=[1:5:6:10;11:15]
```
Entering Different Data Types from the command line -

```
>> a=123
a =
 123
>> size(a)
ans =
  1  1
>> b=[1,2,3,4,5,6,7,8,9,10]
b =
  1  2  3  4  5  6  7  8  9  10
>> size(b)
ans =
   1   10
>> c=[11,12,13;21,22,23;31,32,33]
c =
  11  12  13
  21  22  23
  31  32  33
>> size(c)
ans =
   3   3
>> d=magic(15);
>> e=1:100;
f = [1;5;6;10;11;15]
```

Using `size(variable_name)` is a function that returns the size of each dimension of the matrix `variable_name`.

Entering matrices with 2 or more dimensions requires additional syntax – a semicolon is used in this example to separate data in different rows.

All of the variables show up in the Workspace.
Accessing Different Data Types from the command line -

```matlab
>> a1=[1,2,3;4,5,6;7,8,9]
a1 =
1 2 3
4 5 6
7 8 9
>> a2=[1,2,3,4,5,6,7,8,9];
>> a2=reshape(a2,3,3)
a2 =
1 4 7
2 5 8
3 6 9
>> a2=transpose(a2)
a2 =
1 2 3
4 5 6
7 8 9
```

2 ways to create the same 3x3 matrix

```matlab
>> a1(2,3)
an =
6
>> a1(:,3)
an =
3
6
9
>> a1(3,3)
an =
9
>> a1(3,4)
?? Index exceeds matrix dimensions.
>> a1(9)
an =
9
```

Accesses a single element of a the 3x3 matrix 'a1'

Accesses a whole column of a the 3x3 matrix 'a1'. Guess how you access a single row...

Matlab lets you know when you have breached the array boundaries

Isn't a1 a 3x3 matrix?
When you used the command...

```
>plot(voltage, current)
```

... you are using the pre-defined MATLAB function `plot`. Below is the help page describing how to use the `plot` function. As you can see there are many properties of the plot that you can set as a parameter inside the `plot` function. The title, x-axis label, and the y-axis label can all be specified inside the `plot` function. As you use MATLAB more and more you will realize that there are at least 5 different ways of doing any one task.

```
[output1,output2,output3,...]=function_name(input1,input2,input3)  + Standard syntax for calling functions
```

![MATLAB Help Page for plot function](image-url)