VMI-Specific Chapter Instructions

Chapter 1

How to optionally obtain and install Matlab

Matlab is available on the desktop of every lab computer in the Department of Electrical and Computer Engineering, so you do not need to install it on your personal laptop. In fact, I encourage you to use it in the department computer laboratories, like in NEB 404, 426, and 428 so you can work with the other ECE majors. However, if you want a personal copy, we have a campus-wide licensing agreement and you can take your laptop to the Barracks Help Desk (inside the Barracks Study Room and across from the VMI Post Office, open weekdays 0700-1600) and ask to have it installed it for free. This is a remarkable deal VMI has worked out; I had to purchase a private copy for my consulting work, identical to the ones at VMI except adding two things: permission to be used for private consulting work, and a $2,500 price tag.

Matlab, toolboxes, Simulink, and blocksets

If you do decide to install a VMI-licensed copy on your laptop, be aware that Matlab is actually a family of products that currently includes over 30 add-ons called "*toolboxes*", a separate simulation program called "*Simulink*" and over a dozen add-ons to Simulink called "*blocksets*". VMI has licenses to use all of them. Ask the helpdesk not to install anything other than Matlab until you have finished this class, otherwise it will swell the size of your installation and clutter your search for embedded help by over a factor of ten, effectively hiding the information you want to find in unrelated hits.

Practice problems

Do the class problems either on your own before you come to class or during class. The more work you do before class, the more you can concentrate on working the lab problems. Do the class problems in a Microsoft Word document. For every class problem followed by a © symbol, write the Matlab **c**ommand that solves the problem into the Word document. An ® symbol means to write the **r**esult that Matlab returns into your Word document . Do not re-type anything; cut-and-paste the command or result.

Lab problems

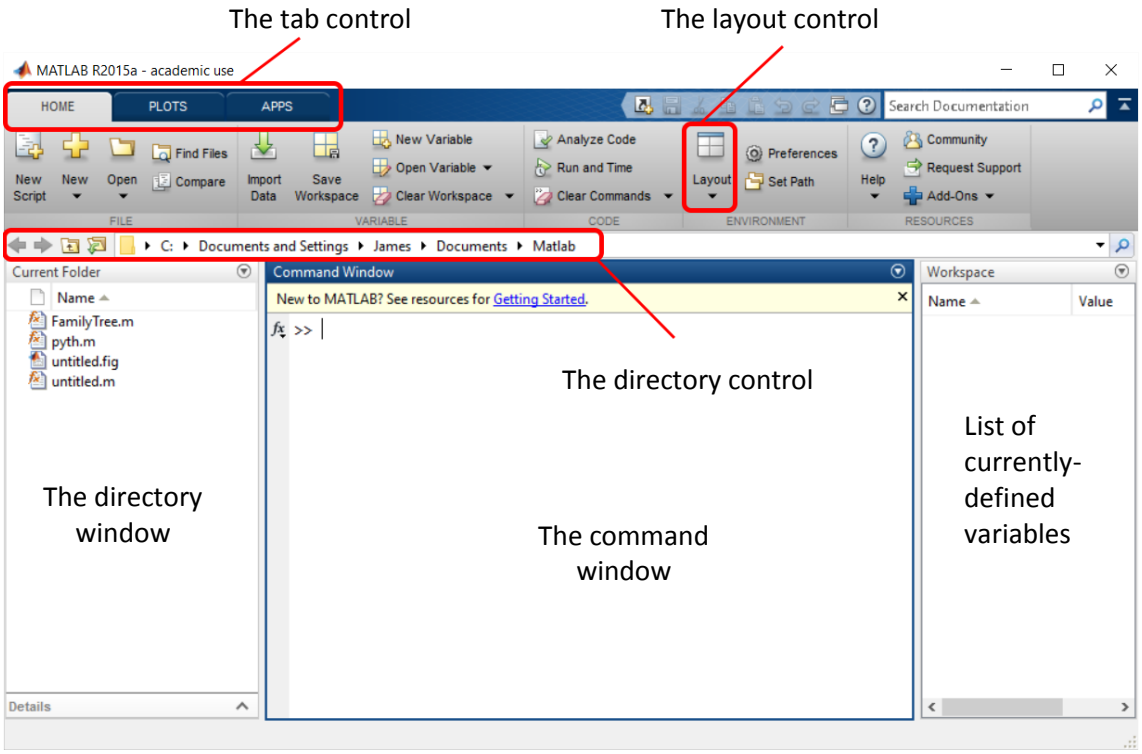
These problems require more thought than the Practice Problems that were designed to check general comprehension. Create a Word document for your answers. For each question, record the Matlab **c**ommand if there is a © next to the problem, and the Matlab **r**esult if there is a ® next to the problem. Example of a question and answer with both a ©® requirement:

Question: Use Matlab to evaluate $4 + \sqrt[6]{5}$ ©®

Answer: Command: 4+(5^(1/6))
Result: 5.3077

Starting Matlab and the workspace

The Matlab program icon is shown to the right. It may take a minute to load, especially if the installation includes many toolboxes or if this is the first time it is run. Matlab will open its standard workspace as shown below.



Matlab is composed of several controls and windows. The most important window is the center "command" window, into which all Matlab commands are entered.

If the Matlab environment looks substantially different than the image above, set the tab control in the upper left to HOME and choose DEFAULT from the dropdown layout control in the upper toolbar. Keep the tab control in the HOME position; the other positions offer various wizards, for example to assist in plotting or filter design, that tend to be more cumbersome to use than learning the direct syntax, as is described in this text.

Next, create a data folder and make it Matlab's current ***working directory***. This is the directory in which Matlab expects to find the user's data and programs.

1) Locate the directory control in the toolbar at the top left of the Matlab main window.
2) Navigate to your personal directory on the network. This will be on the M drive.
3) Inside your personal directory, create a new subfolder called EE120.
4) Inside that folder, create a subdirectory called Chapter1.
5) Note that the Current Directory type-in box in the toolbar now points to your Chapter1 directory.

Chapters 4 and 5: Lab Programming Problems

These problems will be graded by a computer program, and so must be in a certain format to earn points. A large part of your overall grade is based on whether or not your homework submission follows this format – don't miss these easy points!

Unlike previous lab assignments, you will email these to me as a collection of functions saved in a single .zip file, and named with the names of the people in your lab group separated by commas, for example "Ron Howard, Henry Winkler, Erin Moran.zip". This alone is 10% of your overall grade. seven files (or eight if you attempt the bonus) named "problem1.m", "problem2.m" etc. The names must be in lowercase. This is 10% of your overall grade. If the zip files contain anything else (a zip file within a zip file, extra .m files, favorite chocolate cake recipes) then it will fail the automated grading and you will lose a whole letter grade, since if any are incorrect it will require debugging of your entire file to fix and award partial credit.

Each of the functions must have the proper function declaration statement that is provided with the lab problem, and should not generate an error using the test input supplied for each problem. This is worth 10% of the problem grade. **Even if you cannot solve the problem, if your programs do not generate a Matlab error when run you will still earn this 10% for every problem**, but if just one of your functions generates an a Matlab error you will lose the 10% on every problem. This is easy to ensure; if the function declaration given is, for instance,
result = problem10(x1,x2)
Then simply assigning the output value to something, like
result = 0;
will prevent the program from generating an error and earn 10% of the grade. It is possible to earn (or lose) about one-third of your grade by submitting a properly-formatted set of problems.

Note that displaying a number to the screen is not the same as having a function return a number. For instance, say a requirement was to write a function that took no arguments and always returned the number "2". The below function is correct:

```
function out = goodfunction()
out = 2;
```

The function below is not correct, although it does write "2" to the screen, because it doesn't return the number.

```
function badfunction()
fprintf('2');
```

The reason functions return numbers is so their results can be saved in a variable.
x = sin(2); saves the sin(2) in variable x. If the sin() function didn't return the variable but just wrote it to the screen, it would not be of much use. Calling the above functions from the command line yields the following results:

```
x = goodfunction();% This returns 2 saved in variable x.
x = badfunction(); Errors because it doesn't return a number.
```

Problem 8 in Chapters 4 and 5 are bonus problems, and if solved correctly give +15 points, making it possible to earn over 100 points. No partial credit is awarded for this problem, and it is possible to get a perfect 100 score without attempting it. This is a tough bonus problem designed to stretch the abilities of students who find the earlier problems straightforward.

Note: Chapter 5 Problem 7 is an exception; this is a Word file with the graphics of Problem 7, in .docx format, not .m, so your Chapter 5 .zip bundle will have seven .m files and one .docx file.