**THIS IS NOT FOR THE PART OF PROJECT. IT IS JUST FIRST TIME USERS!!!! JUST FOR EXPLANATION**

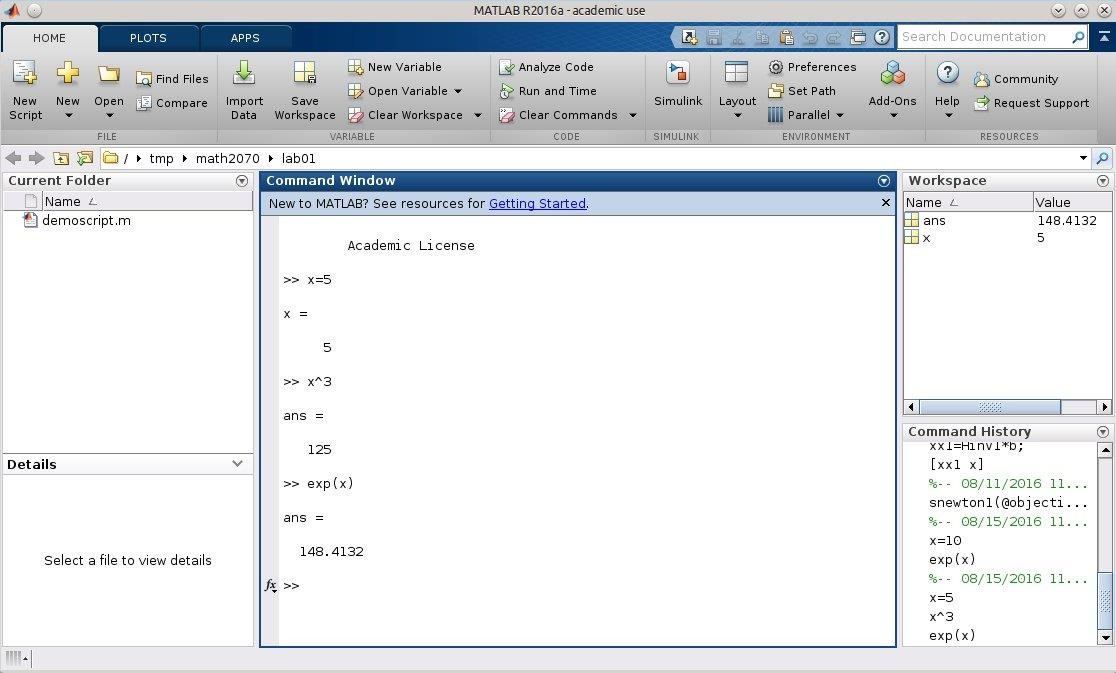
**Preliminaries**

# 

# Starting Up Matlab

In this section you will see how to start up recent versions of Matlab that use the Java windowing interface. These instructions are the same for the Unix and MS-Windows versions of Matlab. I will also give the command-line equivalents of many of the commands. These command-line equivalents are valid for all versions of Matlab, and many of them are valid for the Matlab clone named Octave. Generally speaking, anything you can do using a menu can also be done with command-line equivalents. You would use the equivalents when writing scripts and the menus when working interactively.

1. Start up Matlab by (a) double-clicking the icon on the status bar at the This window will look something like the following:



The default window is divided into a larger pane on the right, and two panes, each with identification tabs at the bottoms, on the left. The right windowpane is a command windowpane and you will be typing Matlab commands in it. On the left side of this command windowpane somewhere (near the bottom in the figure) there will be a prompt of the form ». Your typed commands go next to this prompt.

There are two choices for display on each of the two windowpanes on the left. I suggest that you choose ``Workspace'' on the top and ``Command History'' on the bottom in general, although we will have occasion to use ``Current Directory'' on the bottom.

1. It is a good idea to organize your work into directories I suggest that you use a directory named math381 for all the work in this lab, and subdirectories lab01, lab02, ..., lab10 for each of the labs. You will need to create these directories before you can use them, and the ``Current Directory'' windowpane has the appropriate buttons. After a directory has been created, you can also use the small box named ``Current Directory'' in the middle of the line of icons near the top of the Matlab window.

If you wish to use the command line to create and switch directories, you create a

directory named math381 with the command

cd math381

and you can make it the current directory with the command

cd math381

and you can get a listing of the files in the current directory with either the command

dir or the command

ls

Before going on, create the directory math381, with subdirectories lab01, lab02, ..., lab09.

3. The ``Command History'' windowpane is convenient for recalling what commands you have used recently. In addition, Matlab provides the capability to keep a record of *both* the commands *and* their output. The command to do this is

4.

1. diary diary.txt

The name diary.txt is actually the name of the file that is created and you can use any name you like. It is a text file, and you should use .txt to name it. Before going on, type this command in the ``Command'' windowpane. Terminate the command by hitting the ``Enter'' key.

You should also, type a comment line that will help you identify your work later. It should start with the comment character, a percent sign %, and include the lab number and date. This line will serve as an identifier when you look at the file.

**Note:** The diary file contains a complete record of all the commands you enter in Matlab and the responses. You are to retain this *complete* record to send to me for grading. Please do not turn the diary file on and off or edit the diary file to hide your errors. There are times when you write an explanation that is not, in my opinion, correct and I will need the complete diary to establish what really did happen.

1. Note that you can double-click a command in the ``Command History'' windowpane and it will be executed again. You can also drag a command into the ``Command'' windowpane and then change it using the arrow keys.
2. You can also recover old commands by using the up-arrow key.
3. After you have completed a lab, exit from Matlab with the command quit or use the File->Close Matlab menu choice.

# Using a browser to download files

Some of the labs require that you download files from the web in order to use them. The following exercise illustrates how to download files. The file you will download is a very simple script file.

**Exercise 1**:

Right-mouse click on the file [demoscript.m](file:///C:\Users\Songul\Desktop\m381Fall2017\demoscript.m) to bring up a menu. Choose ``Save link as'' and a file save box will pop up. Navigate to the directory you made: math381/lab01, and save the file with the name demoscript.m. You must use the .m extension to tell Matlab that the file contains Matlab commands. Return to the Matlab command window. The file should be visible to Matlab, a fact that you can confirm with the dir command or by its presence in the ``Current Directory'' windowpane.

Edit the file by typing the command

edit demoscript.m

or by double-clicking on the file in the ``Current Directory'' windowpane, or using the

``FileOpen'' menu pick. An edit window will show up. Read through the file: the comments make it self-explanitory.

You can tell Matlab to execute the file by typing its name, without the ``.m'', on the command line, or by choosing ``Run'' from the Debug menu on the edit window. (The final statements in the file and comments refer to the following exercise.)

The following two exercises illustrate the use of the debugging capability of Matlab. Most of the time you will be able to see what is wrong from the Matlab error messages, but sometimes the error is not obvious. In Exercise 2 below, you will see what you might do when you just cannot see why something is wrong.

**Exercise 2**:

1. Turn on debugging with the commands

2.

1. dbstop if error
2. dbstop if naninf or through the Breakpoints on the Edit window.
3. Now, look at demoscript.m in the edit window. At the bottom there is a statement %bad=1/(x-1);. The percent is a comment character, so this statement is not executed. Make this statement active by deleting the percent character and save the changed file.
4. Execute the script file by typing its name without the .m at the command line or by choosing ``Run'' from the Debug menu on the Edit window.
5. The division by zero caused an exception and Matlab popped up a window with the offending source line highlighted. You should also note the changed prompt in the command window.
6. Place the mouse pointer on top of the x in the expression bad=1/(x-1) and leave it there motionless for a second or so (this is called ``hovering''). The value of x should be displayed. You can then see why the error occurred.
7. Look at the prompt in the command window. It has changed to K». You can do any legal Matlab command at this changed prompt. In particular, you can type the name x to get Matlab to print out its value. This is an alternative way to see the value of a variable in a debug situation.
8. You can exit from debugging mode either using the menu in the edit window or with the command

11.

1. dbquit

This action will return Matlab to its usual » prompt.

1. You can turn off debugging feature with the command

14.

15. dbclear all or from the debugging menu in the edit window.

The following exercise illustrates how you can use the debugger to trace execution. It uses the same demoscript.m file as before.

**Exercise 3**:

1. Type the command clear at the command line. This will return Matlab to its state just after starting up. No variables will show in the ``Workspace'' windowpane.
2. Click on the first executable line (x=1.3) of the demoscript.m file. Go to the ``Breakpoints'' menu and ``set'' a breakpoint there. When a breakpoint is set, a red dot will be placed next to the line number in the edit window.
3. Begin executing the file either by choosing ``Run'' from the debug menu, by pressing the ``Run'' button, or by typing the name demoscript at the command line. The edit window will show that the script is poised to execute the first line of the file. (That line has not yet been executed.)
4. Choose ``Step'' from the Debug menu , press the ``Step'' button (hovering over a button brings up its description) or issue the command ``dbstep'' at the command line. This will cause the current line (line 8) to be executed. You will see the variable x appear in the Workspace windowpane and x=1.3000 appear in the command window.
5. Continue using ``Step'' and watch the script execute, one line at a time.
6. Pressing the ``Continue'' button causes Matlab to continue executing the script until another breakpoint is reached or until the end of the script is reached.
7. Press the ``End Debugging'' button to exit from debug mode. Alternatively, choose Debug->Exit Debug Mode from the menus or use the command dbquit.

# Getting help

It is important to be able to get help when you need it. Matlab provides two help facilities from inside Matlab itself and a third on the web. From the Matlab prompt you can type

help diary

or, in fact, help for any command. You will get a short description of how to use the command. When you write your own Matlab macro files, you should always include some special comments in the beginning of the file. The comments up to the first executable statement or blank line will be printed out in response to the help command. For example, the command

help demoscript

will give a quick help message from the first three lines of demoscript.m. You may notice that the first of these lines is included in the file listing in the ``Current Directory'' windowpane.

A second way to get help from the command prompt is the following.

helpdesk

This command brings up a comprehensive help facility, the same one that the Help menu brings up. This help facility is very similar to the one on the web from the URL: <http://www.mathworks.com/access/helpdesk/help/techdoc/>We will be looking at the help facility on the web later in this lab.

# Quitting

You exit Matlab by typing quit at the command line or by using the FileExit menu choice.

# More on Matlab

The Mathworks, maker of Matlab, includes a short tutorial on using Matlab called [Getting Started](http://www.mathworks.com/access/helpdesk/help/techdoc/learn_matlab/learn_matlab.shtml) This tutorial is also available from the Matlab command prompt with the command helpdesk and also from the Help menu, and, if you have your own copy of the Matlab manuals, it comprises the ``Getting Started'' book.

Begin this tutorial now, during this lab session. Read as much of it as you can now, and complete it at home or from any convenient computer connected to the web. The list of topics that follows (roughly in order of appearance) are topics that will be necessary for this semester's labs. The next lab, addresses some of these topics, but you will be well-served if you have seen the introduction.

1. What is a matrix in Matlab? What is a vector? How are they different from plain numbers?
2. How to you refer to the elements of a matrix or vector in Matlab?
3. How to you use a colon to represent a range of numbers or subscripts?
4. What symbols are associated with the elementary arithmetic operations? (Addition, multiplication, exponentiation, etc.)
5. What meanings do the special constants pi, i, j, inf (infinity), and NaN (not-anumber) have?
6. How do you use the ellipsis ... to put a single statement on more than one line?
7. What are script m-files? What are function m-files? What is the difference?
8. What are anonymous functions and why would you use them?
9. What does format long mean and what other options might be useful?
10. How are the arrow keys used to edit commands in the command window?
11. What is the form of the plot command?
12. What are the hold on and hold off commands used for?
13. How do you indicate matrix sum, product, inverse, and transpose?
14. What do the element-by-element matrix operations such as product mean and how are they denoted?
15. What is an if ... else ... end statement used for? Can you give an example?
16. How do you use the for statement to write a loop?
17. How is the break statement used in a loop?
18. What is the feval statement used for?