Getting Acquainted with MATLAB

Michael Penna, Indiana University – Purdue University, Indianapolis

Objective

This project introduces you to MATLAB. Since MATLAB can run on different types of computers and under different operating systems, and since the choice of computer and operating system used for this and future projects is yours, we make minimal platform specific comments. Questions regarding specific platforms — including how to get MATLAB up and running, how to save, edit, and print — are left to you to resolve.

Narrative

MATLAB is a software package designed for numeric, symbolic, and graphical analysis. MATLAB's strength lies in the many functions it makes available for doing numeric and graphical analysis. In fact, MATLAB is short for MATrix LABoratory, and the primary data structure in MATLAB is the matrix. In these projects, however, we use MATLAB as a tool for doing symbolic and graphical analysis. (We do not use matrices.)

MATLAB functions can be characterized as either "general purpose" or "special purpose". MATLAB's general purpose functions are always available to the user when MATLAB is running. MATLAB's special purpose functions — functions such as the functions we will be using for our symbolic analysis — are organized into "Toolboxes". MATLAB's functions for doing symbolic analysis are in its Symbolic Toolbox, and MATLAB must be properly configured for these functions to be used. We will discuss how this is done in a moment. First, however, we must say a few words about MATLAB's "Desktop".

As soon as you get MATLAB up and running, you will see MATLAB's Desktop. MATLAB's Desktop consists of a menu bar and several window panes. One pane is the Command Window; this is where you enter input and get MATLAB's written output. (MATLAB's graphical ouput appears in "Figure Windows"; we will discuss these later.) In the default configuration for the Desktop, the Command Window is on the right-hand side of the Desktop. In the default configuration for the Desktop, one pane (in the upper left-hand corner) contains the Current Directory and Workspace, and one (in the lower left-hand corner) contains the Command History.

MATLAB's Symbolic Toolbox is "loaded into" (or, perhaps more precisely, made available for use in) MATLAB by entering it into MATLAB's path structure. This can be done by using "Set Path ..." under "File" in MATLAB's menu bar. Since the details of doing this are installation specific, we say no more about how to set the correct path here, but instead refer you to your local computer guru or to your MATLAB documentation.

Task

1. We begin by investigating some of MATLAB's general purpose commands. Type the commands in the left-hand column below into MATLAB's Command Window in the order listed, ending each command line by hitting the Enter key. The effect of each command is described in the right-hand column for your reference. (Do *not* type what is written in the right-hand column.) Make sure you spell things correctly: MATLAB is very sensitive to spelling (and misspelling). In particular, pay attention to the use of upper and lower case letters: MATLAB is case sensitive. (It distinguishes between "A" and "a", for example.) Also be very careful with spaces and punctuation: MATLAB is sensitive to the use (and misuse) of these. If (or when) you make a typographical error (we all make them sooner or later) see Comment 1 at the end of this project.

`>> % Your name, today´s date`	This is a comment. Comments clarify your code but they do not affect computation. Comments always begin with a "%". Enter your name and today's date here.
`>> % Getting Acquainted with MATLAB`	This is the title of this project.
`>> % Task 1:  General Purpose Commands`	
`>> clear all`	Clear MATLAB's memory.
`>> 1+2`	Add 1 and 2.
`>> 9-3`	Subtract 3 from 9.
`>> 4*3`	Multiply 4 by 3.
`>> 24/2`	Divide 24 by 2.
`>> 1/2+1/3`	Add 1/2 and 1/3.
`>> 3^2`	Find 3^2.
`>> a = 3`	Let $a = 3$.
`>> a = 3;`	Let $a = 3$ and suppress output.
`>> a = 3, b = sqrt(2)`	Let $a = 3$ and $b = \sqrt{2}$.
`>> a = 3; b = sqrt(2)`	Let $a = 3$ and $b = \sqrt{2}$.
`>> c = a*b`	Let $c = a * b$.
`>> pi`	What is π?
`>> Pi`	What is `Pi`? Observe that MATLAB is case sensitive: it thinks of `pi` and `Pi` as different quantities.
`>> format long`	Henceforth use 15 digit decimal precision.
`>> pi`	What is π?
`>> format short`	Henceforth use 4 digit decimal precision.
`>> pi`	What is π?

Note that in defining b above we used of one of MATLAB's many built-in functions. To see what other functions are built into MATLAB, see MATLAB's helpdesk. (Type "`helpdesk`" in MATLAB's Command Window, or use MATLAB's pull-down Help menu.) We discuss user-defined functions in our next project. Also observe that if a command is not terminated by a ";" then MATLAB returns either output associated with the name of the computed value or — if the computed value is unnamed — "`ans`"; but if a command is terminated with a "`;`" then output is suppressed. Further observe that multiple commands can be entered on a single line. Finally observe that there's more than one format for numbers in MATLAB. To see what other formats are available, again see MATLAB's helpdesk.

2. a) We now investigate some of MATLAB's Symbolic Toolbox commands. First, however, we note that in mathematics we use not only numeric quantities — quantities such as 2 and 3.14159265 — which have a finite decimal representation, but also symbolic quantities. Symbolic quantities include constants such as $\sqrt{2}$ and π which do not have a finite decimal representation, "generic" constants (which appear, for example, in a discussion of laws such as "$a(b+c) = ab+ac$"), and variables (such as x and y). Only numeric quantities can be handled with "general purpose" MATLAB; numeric *and* symbolic quantities can be handled with MATLAB's Symbolic Toolbox.

   Assuming MATLAB's Symbolic Toolbox has been properly entered into MATLAB's path structure, continue by typing the following commands in the left-hand column below into MATLAB in the order listed, again ending each command line by hitting the Enter key. The effect of each command is described in the right-hand column for your reference.

`>> % Task 2:  Symbolic Toolbox Commands`	
`>> clear all`	Clear MATLAB's memory.
`>> a`	MATLAB does not know what a is (since we cleared its value.)
`>> syms a b c x`	Let a, b, c and x be *symbolic* quantities.
`>> a`	Now MATLAB knows a as a symbolic quantity!

`>> a*(b+c)`	Write the expression $a(b+c)$.
`>> expand(a*(b+c))`	Expand the expression $a(b+c)$.

As this code illustrates, MATLAB can either associate a numeric value to a quantity such as a, or treat it as a symbolic quantity; to treat it as a symbolic quantity, it must be declared symbolic using `syms`.

b) Continue by typing the following commands in the left-hand column below into MATLAB in the order listed. They illustrate a few more Symbolic Toolbox commands.

`>> factor(a*b+a*c)`	Factor $ab+ac$.
`>> expand((a+b)^5)`	Expand $(a+b)^5$.
`>> simplify((x^2-1)/(x-1))`	Simplify $(x^2-1)/(x-1)$.
`>> subs(x^2+3*x-2,x,1)`	Substitute 1 for x in the expression x^2+3x-2.
`>> pretty((x^2-1)/(x-1))`	Write the expression $(x^2-1)/(x-1)$ in algebraic form.
`>> solve(x^2-1,x)`	Solve the equation $x^2-1=0$ for x.

At this time, make a hard copy of MATLAB's Command Window; it will be your lab report for this project. If you made any errors, neatly draw a line through them and any resulting MATLAB output by hand.

Comments

1. If (or when) you make a typographical error in a command line you can correct the error by simply editing it if you act before hitting the Enter key. If you don't find the error until after you Enter it, there are still ways to correct things without having to retype an entire line of code. With an operating system that supports copying, pasting, and editing, you can copy the code you want to correct (just as if you were using a word processor), paste it at the next insertion point, make whatever changes you wish, and then hitting the Enter key. With some operating systems you can use the up-arrow key: Hitting the up-arrow key once results in MATLAB retyping the last line of code; if this is where your error occurred, you can make whatever changes you wish, and then Enter the line of code; if your error occurred prior to the last line of code, you can continue to hit the up-arrow key as many times as necessary to get to the line you wish to alter, then make whatever changes you wish, and hit the Enter key. (Using the up-arrow is also useful for retyping a line code even if no changes are necessary!)

2. Be sure to terminate a line of code with ";" when told to do so: *not* doing so can result in pages and pages of unwanted output.

3. In addition to being careful about *what* you type in each command line, you must be careful about the *order* in which you type command lines in MATLAB. Throughout these projects you will be asked to enter various commands into MATLAB, and when doing this it will always be assumed that you are typing them into MATLAB *in the order in which they are listed.*

4. It is wise, for many reasons, to document your code with your name, the date of your work, the project number and project title. You will, in fact, be asked to do this throughout these projects.

5. While MATLAB allows you to save the quantities defined and computed in a session using "Save Workspace As ..." under "File" in MATLAB's menu bar, MATLAB does *not* allow you to save everything you typed in the Command Window during a session. Thus it is very important for you to make a hard-copy of the Command Window before quitting a MATLAB session (if you ever want to document what you did).