MatLab

Collection Editor:
Serhat Beyenir

MatLab

Collection Editor:
Serhat Beyenir

Authors:
University Of Washington Dept. of Electrical Engineering
Anders Gjendemsjø
Louis Scharf

Online:
< http://cnx.org/content/col11141/1.1/ >

C O N N E X I O N S

Rice University, Houston, Texas

Collection structure revised: December 6, 2009
PDF generated: February 6, 2011
For copyright and attribution information for the modules contained in this collection, see p. 15.

Table of Contents

Chapter 1

An Introduction to MATLAB[1]

MATLAB, short for Matrix Laboratory, is a simple and flexible programming environment for a wide range of problems such as signal processing, optimization, linear programming and so on. The basic MATLAB software package can be extended by using add-on toolboxes. Examples of such toolboxes are: Signal Processing, Filter Design, Statistics and Symbolic Math.

Comprehensive documentation for MATLAB is available at Mathworks.com[2] . In particular, an excellent (extensive) getting started guide is available at Getting started with MATLAB[3] . There is also a very active newsgroup for MATLAB related questions, **comp.soft-sys.matlab**

MATLAB is an interpreted language. This implies that the source code is not compiled but interpreted on the fly. This is both an advantage and a disadvantage. MATLAB allows for easy numerical calculation and visualization of the results without the need for advanced and time consuming programming. The disadvantage is that it can be slow, especially when bad programming practices are applied.

[1] This content is available online at <http://cnx.org/content/m13255/1.1/>.
[2] http://www.mathworks.com
[3] http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/getstart.pdf

Chapter 2

An Introduction to MATLAB: Running MATLAB (Macintosh)[1]

NOTE: This module is part of the collection, *A First Course in Electrical and Computer Engineering*. The LaTeX source files for this collection were created using an optical character recognition technology, and because of this process there may be more errors than usual. Please contact us if you discover any errors.

In order to run MATLAB on a Macintosh SE or PLUS computer, you need the program called EDU-MATLAB. The program requires at least 1 Mbyte of memory, System 3.0 or above, Finder version 3.0 or above, and an 800K drive. A hard disc drive is highly recommended. In order to run MATLAB on a Macintosh II, IIx, IIcx, or $SE/30$, you need the program called MacII-MATLAB, and the same system requirements apply.

To start MATLAB, you may need to open the folder containing the MATLAB program. Then just "double-click" the program icon or the program name (for example, EDU-MATLAB). Figure A.1 shows a typical organization of the folder containing Mac II-MATLAB. It contains the main program, the settings file, the demonstrations folder, and any toolbox folders. The double-click on Mac II-MATLAB produces the *Command* window as shown in Figure A.2. You will also see a *Graph* window partially hidden behind it. (The fact that the window is not in ffont means that it is opened but not currently active.) If you do not know what "clicking," "dragging," "pop-up menu," and "trash" mean, you should stop reading now and familiarize yourself with the Macintosh.

[1]This content is available online at <http://cnx.org/content/m21394/1.4/>.

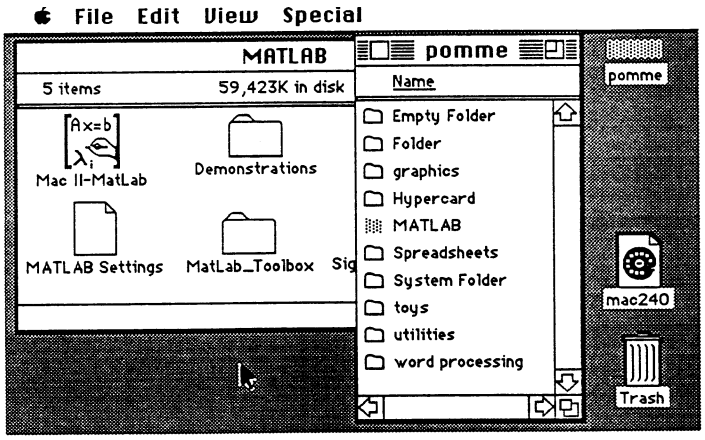

Figure 2.1: The MATLAB Folder (Computer, Inc., used with permission.)

In the command window, you should see the prompt ≫. The program interpreter is waiting for you to enter instructions. At this point it is a good idea to run the demonstration programs that are available in the "About MATLAB" menu under the Apple menu. Just click on the "demos" button and select a demo. During pauses, strike any key to continue. Whenever you have a MATLAB file in any folder, then you may double-click the file to launch the program. This allows you to have your own folder containing your own MATLAB files, separated from the MATLAB folder.

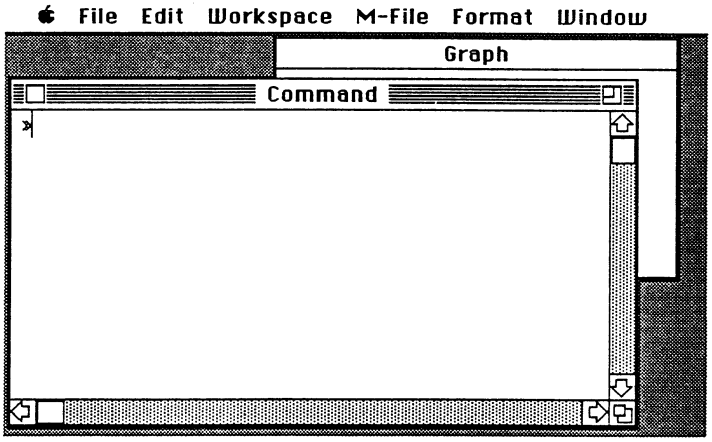

Figure 2.2: The Command Window (Computer, Inc., used with permission.)

MATLAB has four types of windows:

i. *Command* for computing, programming, and designing input/output displays;

ii. *Graph* for displaying plots and graphs;
iii. *Edit* for creating and modifying your own files; and
iv. *Help* for getting on-line help and for running demos.

All windows follow the traditional behavior of Macintosh windows. You can resize them (actually the help window has a fixed size) or move them. For more details on menus and windows, see the Macintosh and MATLAB manuals.

Chapter 3

An Introduction to MATLAB: Running MATLAB (PC)[1]

NOTE: This module is part of the collection, *A First Course in Electrical and Computer Engineering*. The LaTeX source files for this collection were created using an optical character recognition technology, and because of this process there may be more errors than usual. Please contact us if you discover any errors.

In order to run MATLAB (Version 3.5) on an IBM or compatible personal computer, you must have a floating point math coprocessor ($80x87$)installed and at least 512 kbytes of memory. The program is called PCMAT-LAB.EXE, but it is usually invoked via the batch file MATLAB.BAT in the MATLAB subdirectory. If you are using a menu system and MATLAB is one of your choices, just choose it. Otherwise, go to the MATLAB suiUirectory and type MATLAB.

You may be able to usea more powerful implementation of MATLAB if you have an 80286 or 80386 machine. AT-MATLAB runs on an 80286 with at least 1 Mbyte of extended memory. AT-MATLAB is distributed with PC-MATLAB. 386-MATLAB, a special version for 80386 or 80486 machines with virtual memory support and no limits on variable size, is sold separately.

When you run MATLAB, you should see the prompt $\gg$. The program interpreter is waiting for you to enter instructions. Some MATLAB instructions, such as plot, are graphics-type instructions which plot results and data. Execution of one of these graphics instructions puts the PC screen into the *graphics mode*, which displays the resulting plot. No instructions can be executed in the graphics mode other than a screen-dump function. Striking any other key will return the PC to the *command mode*, but the graphics are temporarily stored (like variables) and can be recalled by the **shg** (show graphics) instruction. If you wish, you may run some of the demonstration programs now by entering demo and following the on-screen instructions.

[1]This content is available online at <http://cnx.org/content/m21393/1.4/>.

Chapter 4

Introduction to MATLAB and Scripts[1]

4.1 Introduction

The goal of this lab is to provide exercises that will help you get started learning MATLAB. You will learn about the help function, vectors, complex numbers, simple math operations, and 2-D plots. You may find it useful to try some of the built-in demos in Matlab. Type `demo` to see the choices. In particular, look at the demo on "Basic matrix operations" (under "Mathematics") and on "2-D" plots (under "Graphics"). We will also look at script files in MATLAB, which we will refer to as M-files and have the file extension `*.m`.

4.2 Getting Started

Start MATLAB by clicking on it in the start menu. Once MATLAB is running you will see a screen similar to Figure 1. The command window, (A), is where you can enter commands. The current working directory, (B), displays the directory that MATLAB will look first for any commands. For example, if you made a new MATLAB function called `myfunc.m`, then this will need to be placed in the current working directory. You can change the working directory by typing it in the box, clicking the "..." button to browse to a folder, or typing `cd [directory name]` (i.e. `cd 'H:\ee235\lab0\'`), which is similar to the DOS/Linux cd command).

> NOTE: MATLAB supports tab completion. This means that you can type part of a command and then press tab and it will fill in the rest of the command automatically.

The workspace displays information about all the variables currently active and is shown in (C). The files in the current directory can also be displayed in (C) by clicking on the tab labeled `Current Directory`. A history of your commands is shown in (D). If you find that you do not need some of these windows open you can close them by clicking on the small x in that section of the window.

[1] This content is available online at <http://cnx.org/content/m13554/1.18/>.

The MATLAB GUI

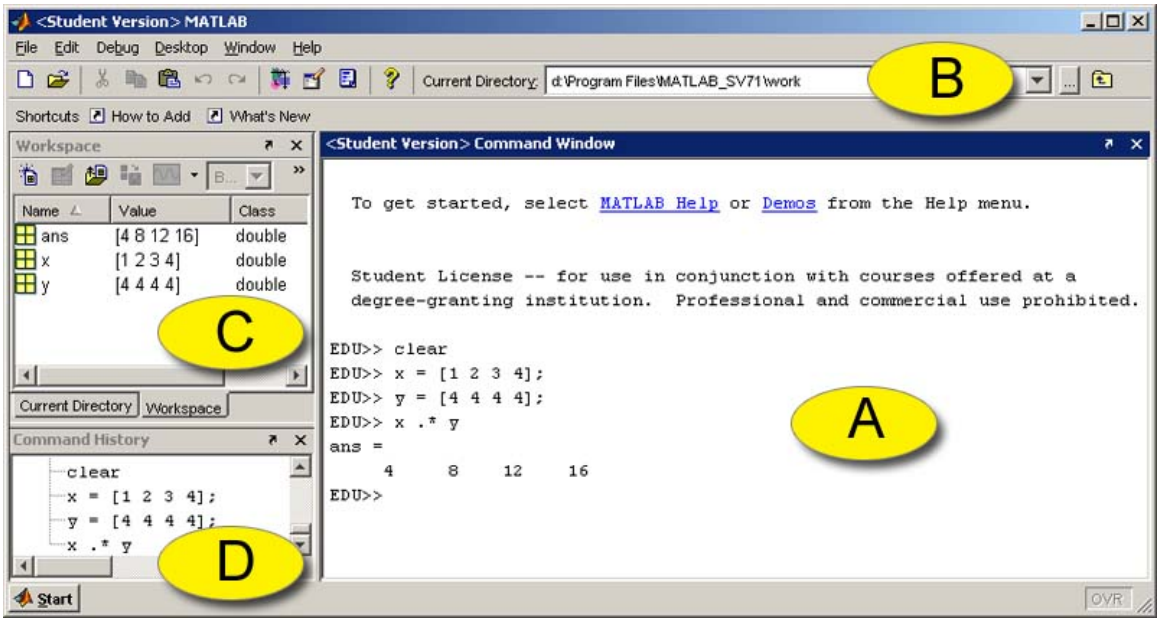

Figure 4.1: (a) Command window, (b) working directory, (c) workspace, (d) command history.

NOTE: There are a number of different ways to use MATLAB on Linux. Typing `matlab` at the command prompt will run MATLAB in X-Windows (warning, MATLAB in X-Windows can be slow when connecting off campus). To run MATLAB without X-Windows type `matlab -nodisplay`. You can also run MATLAB using the current terminal for commands and use X-Windows for everything else (like figures) by typing `matlab -nodesktop`.

4.3 MATLAB Commands

MATLAB works with matrices and therefore treats all variables as a matrix. To enter the matrix

$$x = \begin{pmatrix} 3 & 1 & 5 \\ 6 & 4 & 1 \end{pmatrix}$$

type the command `x = [3 1 5; 6 4 1]`. We can represent an array with a vector, which is a special case of a matrix. A vector can be entered in by typing `y = [1 2 3]`. Now try entering `z = [1 2 3]'`. Is the output what you expect?

1. Familiarize yourself with the help command. Typing `help` gives you a list of all help topics. Typing `help <topicname>` gives help on a specific MATLAB function. For example, use `help plot` to learn about the plot command.

   Rule 4.1: More useful commands

   - `whos` lists all variables
   - `clear` clears all variables

2. Perform the following operations in MATLAB:

a. Generate the following column vectors as MATLAB variables: $x = \begin{pmatrix} 2 \\ 4 \end{pmatrix}$ and $y = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$

b. Using the computer, issue the following MATLAB commands

```
x * y'
x' * y
x .* y
```

Be sure you understand the differences between each of these and you know what the `'`, `*`, and `.*` operators do.

c. Convince yourself that the answer makes sense by checking the matrix dimension and computing each result by hand.

Rule 4.2: Plot and Subplot

The Matlab command `plot` allows you to graphically display vector data (in our case here, the two signals). For example, if you had the variables `t` for time, and `y` for the signal, typing the command `plot(t, y);` will display a plot of `t` vs. `y`. See `help plot` for more information.

Annotating your plots is very IMPORTANT! Here are a few annotation commands.

3.
- `title('Here is a title');` - Adds the text "Here is a title" to the top of the plot.
- `xlabel('Control Voltage (mV)');` - Adds text to the X-axis.
- `ylabel('Current (mA)');` - Adds text to the Y-axis.
- `grid on;` - Adds a grid to the plot.

In order to display multiple plots in one window you must use the `subplot` command. This command takes three arguments, `subplot(m, n, p)`. The first two breaks the window into a m by n matrix of smaller plot windows. The third argument, p, selects which plot is active. For example, if we have three signals x, y, and z, that we want to plot against time, t, then we can use the subplot command to produce a single window with three plots stacked on top of each other.

```
subplot(3,1,1);
plot(t,x);
subplot(3,1,2);
plot(t,y);
subplot(3,1,3);
plot(t,z);
```

See `help subplot` and `help plot` for much more information.

Create and plot a signal $x_0(t) = te^{-|t|}$ over the time range [-10,10] using the following MATLAB commands:

```
t = -10:0.1:10;
xo = t .* exp(-abs(t));
plot(t, xo); grid;
```

The first command defines an array with time values having an 0.1 increment. The ";" is used to suppress printout of the arrays (which are large), and the "`grid`" command makes the plot easier to read. Now create the signals:

$$x_e(t) = |t|e^{-|t|} \tag{4.1}$$

$$x(t) = 0.5 * [x_0(t) + x_e(t)] \tag{4.2}$$

Plot all signals together using 3 plots stacked on top of each other with the subplot command.

```
subplot(3,1,1);
plot(t,xo);
subplot(3,1,2);
plot(t,xe);
subplot(3,1,3);
plot(t,x);
```

Note that $x_o(t)$ and $x_e(t)$ are the odd and even components, respectively, of $x(t) = te^{-t}u(t)$

4. **Complex Numbers:** One of the strengths of MATLAB is that most of its commands work with complex numbers. Perform the following computations in MATLAB.
   a. MATLAB recognizes `i` as an imaginary number. Try entering `sqrt(-1)` into MATLAB, does the result make sense?
   b. MATLAB uses the letter `i` instead of `j` by default. Electrical Engineers prefer using `j` however, and MATLAB will recognize that as well. Try entering `i+j`, does this make sense.

      NOTE: If you are using complex numbers in your code, it's a good idea to avoid using `i` and `j` as variables to prevent confusion.

   c. Define $z_1 = 1 + j$. Find the magnitude, phase, real and imaginary parts of z (using `abs()`, `angle()`, `real()`, `imag()`, respectively). Is the phase in radians or degrees?
   d. Find the magnitude of $z_1 + z_2$ where $z_2 = 2e^{\frac{j\pi}{3}}$
   e. Compute the value of j^j . Is the result what you expect?
5. **Complex Functions:** MATLAB also handles complex time functions in the same way (again, implemented as vectors) . Create a signal $x_1(t) = te^{jt}$ over the range [-10,10], as in part 3. Next plot the real and imaginary parts of the signal in two plots, one over the other using the subplot command. Notice that one plot is odd and one is even. Try proving to your self analytically that this is what you would expect.
6. **Playing and Plotting a Sound** Load the built-in data named "handel" and play it:

```
load handel;
plot(linspace(0,9,73113),y);
sound(y);
```

NOTE: You can use the `clear` command in MATLAB to clear all of the varibles

4.4 Script Files

Scripts are `m-files` files that contain a sequence of commands that are executed exactly as if they were manually typed into the MATLAB console. Script files are useful for writing things in MATLAB that you want to save and run later. You can enter all the commands into a script file that you can run at a later time, and have the ability to go back and edit it later.

You need to use a text editor to create script files, e.g. `Notepad` on the PC's (`pico`, `emacs`, or `vi` on Linux machines). MATLAB also has an internal editor that you can start by clicking on a `.m` file within MATLAB's file browser. All are easy to learn and will produce text files that MATLAB can read.

Click here[2] to download the `dampedCosine.m` script and be sure to save it with that name to follow the instructions here exactly. It is very important that script filenames end in `.m`. Be sure that MATLAB's working directory is set to the location of where you saved the script file. Type dampedCosine at the MATLAB prompt. Look at the m-file in a text editor and verify that you get the plot predicted in the comment field of the script.

NOTE: The `%` character marks the rest of the line as a comment.

Exercise 4.1
Scripts
Now we are going to edit the `dampedCosine.m` file to create our own script file. At the top of the file, you will see the following commands

```
diary 'your_name_Lab1.txt'
disp('NAME: your name')
disp('SECTION:your section')
```

1. Edit the dampedCosine.m (download from link above) script and enter your name and section where indicated. Save this new version of the script as `yourName_dampedCosine.m`
2. Edit the script to create a second signal where the cosine with twice the period (which gives half the frequency) of the first.
3. Add to the script the commands to plot these together with the first signal on top and the second on the bottom. In other words, you should have a single figure with two different plots, one on top and one on bottom. You will need to use `subplot` and `plot`. Save this plot as `yourName_dampedCosine.fig`.
4. Show the TA your dampedCosine plot. What is the period of the cosine?

Exercise 4.2
Complex exponentials
Download and run compexp.m[3] , which includes a 3-D plot of a complex exponential, $y(t)$, as well as 2-D magnitude/phase and real/imaginary plots. You need 2 2-D plots to have the same information as the 3-D plot. How would you change the script to make the oscillation frequency lower by half? How would you change the script to make the decay faster? Show the TA your plots.

[2] http://cnx.org/content/m13554/latest/dampedCosine.m
[3] http://cnx.org/content/m13554/latest/compexp.m

Index of Keywords and Terms

Keywords are listed by the section with that keyword (page numbers are in parentheses). Keywords do not necessarily appear in the text of the page. They are merely associated with that section. *Ex.* apples, § 1.1 (1) **Terms** are referenced by the page they appear on. *Ex.* apples, 1

Attributions

Collection: *MatLab*
Edited by: Serhat Beyenir
URL: http://cnx.org/content/col11141/1.1/
License: http://creativecommons.org/licenses/by/3.0/

Module: "An Introduction to MATLAB"
By: Anders Gjendemsjø
URL: http://cnx.org/content/m13255/1.1/
Page: 1
Copyright: Anders Gjendemsjø
License: http://creativecommons.org/licenses/by/2.0/

Module: "An Introduction to MATLAB: Running MATLAB (Macintosh)"
By: Louis Scharf
URL: http://cnx.org/content/m21394/1.4/
Pages: 3-5
Copyright: Louis Scharf
License: http://creativecommons.org/licenses/by/3.0/

Module: "An Introduction to MATLAB: Running MATLAB (PC)"
By: Louis Scharf
URL: http://cnx.org/content/m21393/1.4/
Page: 7
Copyright: Louis Scharf
License: http://creativecommons.org/licenses/by/3.0/

Module: "Introduction to MATLAB and Scripts"
By: University Of Washington Dept. of Electrical Engineering
URL: http://cnx.org/content/m13554/1.18/
Pages: 9-13
Copyright: University Of Washington Dept. of Electrical Engineering
License: http://creativecommons.org/licenses/by/2.0/

MatLab
A Brief MatLab Intro

About Connexions
Since 1999, Connexions has been pioneering a global system where anyone can create course materials and make them fully accessible and easily reusable free of charge. We are a Web-based authoring, teaching and learning environment open to anyone interested in education, including students, teachers, professors and lifelong learners. We connect ideas and facilitate educational communities.

Connexions's modular, interactive courses are in use worldwide by universities, community colleges, K-12 schools, distance learners, and lifelong learners. Connexions materials are in many languages, including English, Spanish, Chinese, Japanese, Italian, Vietnamese, French, Portuguese, and Thai. Connexions is part of an exciting new information distribution system that allows for **Print on Demand Books**. Connexions has partnered with innovative on-demand publisher QOOP to accelerate the delivery of printed course materials and textbooks into classrooms worldwide at lower prices than traditional academic publishers.