Getting Started With MATLAB

Prepared by the staff of the Indiana University Stat/Math Center

---

What is Matlab?

MATLAB is a computer program for people doing numerical computation, especially linear algebra (matrices). It began as a "MATrix LABoratory" program, intended to provide interactive access to the libraries Linpack and Eispack. It has since grown well beyond these libraries, to become a powerful tool for visualization, programming, research, engineering, and communication.

Matlab's strengths include cutting-edge algorithms, enormous data handling abilities, and powerful programming tools. The Matlab is not designed for symbolic computation, but does include a Maple kernel for such computations. The interface is mostly text-based, which may be disconcerting for some users.

Matlab is packaged as a core program with several "toolboxes", sold seperately. We will only cover the base package.

How to use this document

This document is intended to be used while sitting at a computer running either NT or the X windows system. It is assumed that you will enter the commands shown, and then think about the result.

The reader of this document should have at least a passing familiarity with linear algebra and be comfortable using computers. In order to be more broadly understood, we will not cover any engineering topics (e.g. signal processing, spectral analysis), though Matlab is commonly used for these tasks. No previous math software experience is necessary, though we will point out important differences between the various packages along the way.

If you are using Matlab over a text-based terminal (such as Telnet or SSH), you might want to follow the Using Math Software under UNIX tutorial after you are finished with this one.

Throughout this document, we will use the following conventions.

Example	Explanation
File -> Open	Choose the file menu, and select Open.
`a = 5`	Input to be typed at the Matlab prompt.
`a =` `5`	Output from Matlab.
	An important tip.

Where to find Matlab

MATLAB is available for many different kinds of computers at Indiana University Bloomington.

- All Student Technology Centers
- The central server Quarry.

A student edition is available from local bookstores for your personal Windows, Macintosh, and Linux systems.

How to Start and Exit Matlab

To start Matlab:

- On a Windows machine, choose Start Menu ->Programs->Statistics-Math->Matlab.
- On a Unix machine type `matlab` to run it interactively.
  Note that plots will be displayed in an X-window, if you have set the DISPLAY variable. You may also type `matlab -display hostname:0`.

Note that on some machines Matlab may take 15 to 20 seconds to load.
Eventually, you should see:

```
                    < M A T L A B >
Copyright 1984-2005 The MathWorks, Inc.
 Version 7.0.4.352 (R14) Service Pack 2
                    January 29, 2005


  New users may want to read "Getting Started with Matlab."
  at http://www.indiana.edu/~statmath/math/matlab/gettingstarted

  Please contact statmath@indiana.edu with any questions or comments.
```

The line >> is the Matlab prompt.
To exit Matlab, type (at the matlab prompt) `exit` or choose File->Exit MATLAB.

Finding Your Way Around the Window

The Command Window

The Command window is where you can interact with Matlab directly. Try typing the following:

```
foo = ones(2)
```

The output should look like this:

```
foo =

     1     1
     1     1
```

This represents a 2x2 matrix of ones. Matlab inserts extra blank lines between practically everything. To turn off this feature, type

```
format compact
```

Unlike Maple and Mathematica, previous input lines cannot be edited. You can recall previous input lines by typing the "up" cursor button. If your type some text before hitting the "up" cursor button, Matlab will recall lines matching the text you've type so far.

Cutting, copying and pasting is supported using the standards for your operating system.

The Toolbar

Systems that support graphics have a toolbar at the top of the screen, like this:

The buttons have the following functions:

 New File, Open File

 Cut, Copy, and Paste

 Undo last action

 Workspace Browser. Use this to graphically edit variables.

 Path browser. Use this to edit the paths that Matlab will look in for functions.

 Simulink Library Browser (not covered in this tutorial).

 Open help window.

The Help System

Built-in Help
If you know the name of a Matlab function you need help with, type
`help function-name`
to see the help text contained in the function definition itself. If you don't know the name of the function you need, try `lookfor keyword` . Matlab then searches through the first line of help text of each function for the keyword. However, the search is very slow, and often deluges the user with many items. In addition, a search for a concept, such as "matrix" will almost never return what you need.

 Most of the help files list function names in all capital letters for emphasis. However, all built-in matlab functions are in small letters.

Syntax

How Matlab Works

Matlab works by executing the mathematical statements you enter in the command window. By default, any output is immediately printed to the window.

You are also allowed to assign a name to an expression for your convenience. Keep in mind that the name you assign

is only a name, and it does not represent a mathematical variable (as it would in Maple, for example). Every name must have a value at all times. If you try to read the value of an unassigned name, you will get an error.

Nearly everything in Matlab is a matrix, whether it looks like it or not. This takes some getting used to. We'll be introducing matrix-style operations along with their scalar counterparts so you can understand the patterns that arise in the syntax.

Symbols and Punctuation

Matlab was designed to use fairly standard notation. Try these examples on your own computer.

Input	Output	Comments

`2 + 3`
`7-5`
`34*212`
`1234/5786`
`2^5`
	`ans = 5`
	`ans = 2`
	`ans = 7208`
	`ans = 0.2173`
	`ans = 32`
		Arithmetic works as expected. Note that the result is given the name "ans" each time.

`a = sqrt(2)`	`a = 1.4142`	You can choose your own names for things.

`b = a, pi, 2 + 3i`	`b = 1.4142`
	`ans = 3.1416`
	`ans = 2.0000 + 3.0000i`
		You can use commas to put more than one command on a line. Pi, i, and j are contants.

`c = sin(pi)` `eps`	`c = 1.2246e-016` `ans = 2.2204e-016`	"eps" is the current limit of precision. Anything smaller than eps is probably zero. Note that Matlab understands (and expects you to understand!) scientific notation.

`d = [1 2 3 4 5 6 7 8 9]`
`e = [1:9]`
`f = 1:9`
	`d = 1 2 3 4 5 6 7 8 9`
	`e = 1 2 3 4 5 6 7 8 9`
	`f = 1 2 3 4 5 6 7 8 9`
		"d", "e", and "f" are all vectors. They are equal. Note the use of the ":" operator - it counts (by ones) from one number to the next.

`g = 0:2:10`
`f(3)`
`f(2:7)`
`f(:)`
	`g = 0 2 4 6 8 10`
	`ans = 3`
	`ans = 2 3 4 5 6 7`
	`1 2 3 4 5 6 7 8 9`
		More uses of the colon. Note that you can use it to get slices of a vector (or matrix, or cube, etc), or get the whole thing.

`h = [1 2 3];`
`h'`
	(nothing)
	`ans = 1`
	`2`
	`3`
		A semi-colon ";" will prevent the output from being displayed. A single quote " ' " computes the transpose of a matrix, or in this case, switches between row and column vectors.

`h * h'`
`h .* h`
`h + h`
	`ans = 14`
	`ans = 1 4 9`
	`ans = 2 6 8`
		Operations on vectors. * is matrix multiplication, and so the dimensions must line up correctly.
		" .* " is entry-by-entry multiplication.

`g = [ 1 2 3; 4 5 6; 7 8 9]`	`g = 1 2 3`
	`4 5 6`
	`7 8 9`
		Entering a matrix.

`g(2,3)`
`g(3,:)`
`g(2,3) = 4`
	`ans = 6`
	`ans = 7 8 9`
	`g = 1 2 3`
	`4 5 4`
	`7 8 9`
		Accessing matrix elements.
		Note use of ":" to access an entire row.

`g^2`
 	
`g .^ 2`
	`ans = 30 36 42`
	`66 81 96`
	`102 126 150`
	`ans = 1 4 9`
	`16 25 36`
	`49 64 81`
		The first multiplies the matrix by itself. The second squares each entry in the matrix.

How to Control Output

Before we go any deeper into matrices, it would be wise to mention formatting issues.

Two useful commands are `format` and `more`.

- To control linespacing, use `format compact`.
- To see all 15 digits that were used in calculation, use `format long`.
- To see just 5 digits, use `format short`.
- To supress output completely, use a semi-colon at the end of the command.
- To see one page of output at a time use `more on`.

To see other options, type `help format` or `help more`.

Note that Matlab *always* uses "double" precision (about 15 digits) in its calculations. These commands merely adjust the display.

More on Matrices

More ways of constructing matrices

Built-in Constructions
There are many built-in matrix constructions. Here are a few:

Input	Output	Comments
`rand(2)`	`ans = 0.9501 0.6068` `0.2311 0.4860`	Generates a matrix with entries randomly distributed between 0 and 1
`rand(2,3)`	`ans = 0.8913 0.4565 0.8214` `0.7621 0.0185 0.4447`	
`zeros(2)`	`ans = 0 0` `0 0`	Generates a 2x2 matrix with all zero (or all ones) entries.
`ones(2)`	`ans = 1 1` `1 1`	
`eye(2)`	`ans = 1 0` `0 1`	Identity matrix I.

`hilb(3)`	`ans = 1.0000 0.5000 0.3333`
	`0.5000 0.3333 0.2500`
	`0.3333 0.2500 0.2000`
		3x3 Hilbert matrix.

To get more information on these, look at the help page for special matrices- type `help elmat`.

Concatenation
New matrices may be formed out of old ones. Suppose we have

```
a = [1 2; 3 4]
a = 1 2
    3 4
```

Then we can make a new matrix in any of the following ways.

Input	Output
`[a, a, a]`	`ans = 1 2 1 2 1 2` `3 4 3 4 3 4`

`[a; a; a]`	`ans = 1 2`
	`3 4`
	`1 2`
	`3 4`
	`1 2`
	`3 4`

`[a, zeros(2); zeros(2), a']`	`ans = 1 2 0 0`
	`3 4 0 0`
	`0 0 1 3`
	`0 0 2 4`

Programming
Matrices may also be constructed by programming. Here is an example using two "for" loops.

```
for i=1:10,
   for j=1:10,
      t(i,j) = i/j;
   end
end
```

Notice that there isn't any output, since the only line that would produce any (`t(i,j) = i/j;`) ends in a semi-colon. Without the semi-colon, Matlab would print the matrix t 100 times!

We'll cover programming in more detail later.

More matrix operations

As we saw earlier, +, -, *, and / are defined in an intuitive manner for matrices. When there is ambiguity about whether an operation will do matrix arithmetic (versus entry-by-entry arithmetic), note that ".*" (dot-star) will multiply entry-by-entry, and "*" (star) will do matrix multiplication.

Scalars
A scalar is just a number. Matlab stores them internally as 1x1 matrices, but treats them as if they were numbers.

All operation involving a scalar and a matrix affect the matrix on an entry-by-entry basis, with one exception: the power ("^") operator. With a as defined above, try these:

Input	Output	Comments
`b=2`	`b=2`	Define b to be a scalar.
`a + b`	`ans = 3 4` `5 6`	Addition works entry-by-entry.
`a * b`	`ans = 2 4` `6 8`	So does multiplication.
`a ^ b`	`ans = 7 10` `15 22`	This is *matrix* power - a*a
`a .^ b`	`ans = 1 4` `9 16`	Entry-by-entry power.

Vectors
A vector is just a matrix with only one row (or column). Matlab makes a distinction between row vectors and column vectors, and will complain if it doesn't get what it expects.

In all arithmetic operations, Matlab treats a vector as a matrix, so we already know how to multiply a vector by a scalar. Matlab also gives you the following operations:

Input	Output	Comments
`v = [1 2 3]` `u = [3 2 1]`	`v = [1 2 3]` `u = [3 2 1]`	Define a pair of vectors.
`v * u`	`Error`	The dimensions don't agree.
`v * u'`	`ans = 10`	Taking the transpose works.
`dot(v,u)`	`ans = 10`	The dot product is the same thing.
`cross(v,u)`	`ans = -4 8 -4`	The cross product works only for 3-d vectors.

Matrices
Matlab has all the common operations built-in, as well as most of the obscure ones.

Input	Output	Comments

`k = 16 2 3`
`5 11 10`
`9 7 6`
	`k = 16 2 3`
	`5 11 10`
	`9 7 6`
		Define a matrix.

`rref(k)`	`ans = 1 0 0`
	`0 1 0`
	`0 0 1`
		Row-reduced echelon form

`rank(k)`	`ans = 3`	The rank.
`det(k)`	`ans = -136`	The determinant.

`inv(k)`	`ans = 0.0294 -0.0662 0.0956`
	`-0.4412 -0.5074 1.0662`
	`0.4706 0.6912 -1.2206`
		Inverse of the matrix

`[vec,val] = eig(k)`	`vec = -0.4712 -0.4975 -0.0621`
	`-0.6884 0.8282 -0.6379`
	`-0.5514 0.2581 0.7676`
	`val = 22.4319 0 0`
	`0 11.1136 0`
	`0 0 -0.5455`
		Eigenvectors and eigenvalues of the matrix.
		The columns of "vec" are the eigenvectors,
		and the diagonal entries of "val" are the eigenvaules.

There are many more functions like these. Type `help matfun` to see them all.

Solving Equations

One of the main uses of matrices is in representing systems of linear equations. If `a` is a matrix containing the coefficients of a system of linear equations, `x` is a column vector containing the "unknowns," and `b` is the column vector of "right-hand sides," the constant terms, then the matrix equation

$$a\,x = b$$

represents the system of equations.

If you know a little linear algebra, you could use the Matlab commands from above to augment the matrix and then find the row-reduced echelon form. However, Matlab provides a more simple mechanism for solving linear equations:
`x = a \ b`
This can be read "x equals a-inverse times b."
Try it with
`a = [1 2 3; 4 5 6; 7 8 10]; b = [1 1 1]';`
You should get
`x = -1 1 0`

To verify this assertion, try this:
```
a*x, a*x - b, eps
```
The results are:
```
ans = 1 1 1
ans = 1.0e-015 *
    -0.1110
    -0.6661
    -0.2220
ans = 2.2204e-016
```
Notice that `a*x - b` is very close to `eps` - which means that it is as close to zero as possible.

If there is no solution, a "least-squares" solution is provided (a*x - b is as small as possible). Enter
```
a(3,3) = 9; b = [1 1 0]';
```
(which makes the matrix singular and changes b) and enter the above commands again, using the up-arrow to recall them. Notice that the solution is quite inaccurate.

Saving and loading matrices

When you exit Matlab, you will not be prompted to save your work. If you would like a record of your work, you can turn on "logging" by typing

`diary 'm:\scratch\session.txt'` on Windows machines
`diary '~/session.txt'` on Unix.

Notice that output will be saved alongside your input, so the file can't be used directly as a script.

You may just want to save one or more matrices. The diary is a very clumsy way to do this.
To save the value of the variable "x" to a plain text file named "x.value" use
```
save x.value x -ascii
```
To save all variables in a file named "mysession.mat" in reloadable format, use
```
save mysession
```
To restore the session, use
```
load mysession
```
Then, to see the saved files from your session, on UNIX systems type the commands:

```
% more session.txt
% more x.value
```

Under Windows open the appropriate file with Notepad.

Graphics

Matlab has outstanding graphics capabilities (you must be using a terminal which supports graphics to use them). However, graphing in Matlab is conceptually different than graphing in Maple or Mathematica.

Graphics Concepts

Before looking at the plotting capabilities of Matlab, consider what a graph really *is*. A graph is a collection of points, in 2,3 or even 4 dimensions, that may or may not be connected by lines or polygons. Most math software packages hide this from the user by sampling a continuous function to generate the points.

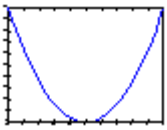
Matlab is designed to work with matrices, rather than functions. Matrices are a convenient way to store a collection of numbers - which is exactly what is needed when graphing. Thus all graphing commands in Matlab accept *matrices* as their argument, rather than a function. If you are used to function-style plotting, Matlab may take some getting used to. On the other hand, Matlab's approach makes it very easy to visualize data and to create graphics based on lists of

points.

Another unique feature of Matlab's graphics engine is the way in which it displays graphical output. In Matlab, there is (usually) only one plotting window. Subsequent plotting commands will overwrite the old plot, unless you request a new one be made. This allows a plot to be made, then adjusted later to suit your needs.

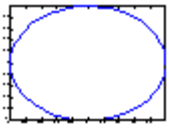
Basic 2-D Graphics

Now that you understand a bit more about graphing, try these examples. Be sure to follow along exactly, or you may not get the same results.

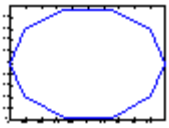
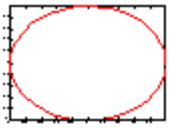
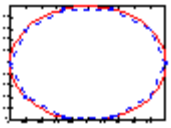
Input	Output	Comments
`x = -10:0.5:10;` `y = x .^ 2;`	(none)	Create a 1x41 matrix that counts from -10 to 10, and a 1x41 matrix made by squaring the entries of the first matrix.
`plot(x,y)`		Plot the points. Each pair is plotted, so <x(1), y(1)> is a point, <x(2), y(2)> is a point, etc.

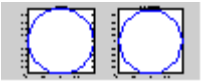
`x = -10:0.5:10;`
`plot(x,sin(x))`
`hold on`
`plot(x,cos(x))`
`hold off`
	y=x^2	Plot one plot over another.

`t = 0:0.1:2*pi;`
`x = cos(t);`
`y = sin(t);`
	(none)	Generate some new 1x63 matrices.

`plot(x,y)`		Plot the points. This is called a parametric plot. Notice that it replaces the previous plot.

`t = 0:pi/5:2*pi;`
`u = cos(t);`
`v = sin(t);`
	(none)	Generate some new 1x11 matrices. This shows how to control the "resolution" of the plot.

`figure` `plot(u,v)`		Create a new plot window, and plot the points. Notice how jagged the circle is, since we only used 11 sample points.
`plot(x,y, 'ro-')`		Plot the hi-res version again in red, with circles at the data points, connected by lines.
`plot(x,y, 'r-',` `u,v,'b*:')`		On the same plot, plot the hi-res version in red, and plot the lo-res version with blue stars at the data points and dotted lines.

`figure`
`subplot(1,2,1)`
`plot(x,y)`
`title('Fine Mesh')`
`subplot(1,2,2)`
`plot(u,v)`
`title('Coarse Mesh')`
		Create a new figure.
		Divide it into one row, two columns, and pay attention to the first cell.
		Plot.
		Give the current plot a title.
		Pay attention to the second cell.
		Plot.
		Give it a title.

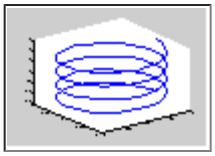
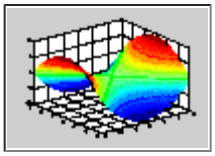
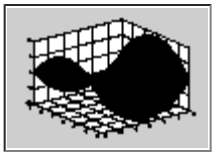
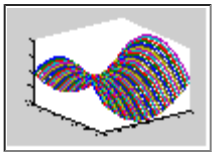
See the following help files for more options and ideas: `help plot`, `help comet`, `help semilogy` and `help fill` .

Matlab provides very powerful features in the figure window. Use the toolbar at the top to add arrows, lines, and text comments to your plot.

Basic 3-D Graphics

Input	Output	Comments

`t = -4*pi:pi/16:4*pi;`
`x = cos(t);`
`y = sin(t);`
`z = t;`
	(none)	Generate the data...

`plot3(x,y,z)`		... and draw a helix.
`[x, y] = meshgrid(-3:0.1:3,-3:0.1:3);` `z = x.^2 - y.^2;`	(none)	Generate data for a surface plot.
`mesh(x,y,z)`		Draw the surface using a mesh.
`surf(x,y,z)`		Draw the surface as a "patched" surface.
`plot3(x,y,z)`		Notice that it still plots, but as a set of arches.

Advanced Plotting

Of course, Matlab can do a lot more than these simple exercises. If you'd like to know more, try these help files: `help slice` , `help movie` , `help getframe` , `help graph2d` , `help graph3d` , `help graphics` .

Programming

- How to program
- Functions
- Batch programming
- Programming ideas

How to program

Matlab statements can be prepared with any editor, and stored in a file for later use. The file is referred to as a script, or an "m-file" (since they must have names of the form `foo.m`). You may prefer to do this, if you use the same data repeatedly, or have an editor that you like to use. Writing m-files will make you much more productive.

Select File -> New ->M-file to open Matlab's built-in editor/debugger, or use your favorite editor. Then create the

following file, and save it as sketch.m:

```
[x y] = meshgrid(-3:.1:3, -3:.1:3);
z = x.^2 - y.^2;
mesh(x,y,z);
```

Next, in Matlab, be sure that the directory where the m-file resides is in the path. Check this by typing `pathtool` and making sure that your directory is present. (Users without graphics can use `addpath` *`directory`* .)

Now enter

`sketch`

The result is the same as if you had entered the three lines of the file, at the prompt.

You can also enter data this way: if a file named mymatrix.m in the current working directory contains the lines

```
A = [2 3 4; 5 6 7; 8 9 0];
```

then the command `mymatrix` reads that file and generates A. However, for large matrices, Matlab's own `save` and `load` commands are much safer to use.

Functions

Functions are like any other m-file, but they accept arguments, and they are compiled the first time they are used in a given session (for speed).

Use your favorite editor to create a file named sqroot.m, containing the following lines.

```
function sqroot(x)
% SQROOT compute squares root by Newton's method

% Initial guess
xstart = 1;

for i = 1:100
        xnew = ( xstart + x/xstart)/2;
        disp(xnew);
        if abs(xnew - xstart)/xnew < eps, break, end;
        xstart = xnew;
end
```

In Matlab enter the commands

```
format long
sqroot(19)
```

You should see the output from your function.

Two caveats:

- A function has access to the variables in the "workspace" from which it was called, but the variables created within the function (xstart and xnew, in the preceeding example) are local, which means that they are not shared with the calling workspace. For more information, see the chapter "M-File Programming" in the manual, *Using Matlab*.
- Note: if you edit a function during a session, use `clear` *`function_name`* to remove the compiled version, so that the new one will be read.

Batch Programming

MATLAB may be ran in "batch mode," in a very similar way. If a file named "test.m" contains the (non-graphics) commands you want processed, at the UNIX prompt type:

```
% matlab < test.m > homework.out
```

This is read, "Run MATLAB, with input from test.in, and output to test.out." The input file does not need to be named "something-dot-m," but it **must** end with `quit` .

Programming ideas

The "m-files" which came with MATLAB provide lots of examples! To find their location, use `path` .This will also lead you to some demos.

You might also try typing `demo` to get a feel for the breadth of tasks that can be accomplished with Matlab.

See `help function` for an exercise for programmers.

Further Reading

- The Matlab page should be especially useful.
- If you use UNIX, Using Math Software under UNIX may be helpful.
- Manuals for statistical and mathematical software are kept in the Swain Hall, Business/SPEA, Education, and IUPUI University Libraries, and at the IU Center for Statistical and Mathematical Computing. The manuals which come with Matlab are especially good, and provide much more information than this short course.
- If you are affiliated with Indiana University, you may also get help from the Stat/Math Center.
