Information Technology
Rice University

Document UNIX 17
July 11, 2002

Introduction to Matlab

Basic Matlab Commands and Syntax

This document teaches the user how to create Matlab matrices, learn about Matlab plots and printing, and discover how to use mathematical equations in Matlab, and to acquire rudimentary Matlab skills.

Table of Contents

If you have any comments or suggestions about this document, send them to problem@rice.edu via electronic mail.

What is Pro-Matlab?

Pro-Matlab (Matlab) is an interactive software package from the Math Works Inc. Matlab stands for Matrix Laboratory. It will define and perform operations on both numbers and characters. There are a number of higher math functions, such as filtering and numeric integration, built into the package. Matlab also has a graphics interface, and the capablility for programming through M-files.

Entering and Leaving Matlab

There are two ways to start Matlab on the Rice Unix Facility. (These methods will also work for Information Systems and Owlnet.) One is to type

```
% matlab
```

in an xterm. Another is to use the right button menu option for Matlab. The test accounts used for this course have this option. If you have a personal Unix account on RUF, IS, or Owlnet and you do not have Matlab as a menu option, you can add the following line to your .twmrc file.

```
“Matlab” ! "xterm -n Matlab -e sh -c /usr/local/gomatlab &”
```

To exit Matlab. you simply type:

```
quit
```

in the Matlab window.

Try one of these methods to start Matlab. Exit, and try the other. Please do not run more than one copy of Matlab at any given time on this system, as we only have a limited number of licenses, and if you run two another user may not be able to run any.

Variables and the Workspace

Defining Numbers and Characters

To define a scalar we may simply use the equal sign.

```
a=3
```

To define a vector we use the equal sign and square brackets.

```
v=[3 5 7]
```

Matrix definition follows the same form as vector definition. The matrix rows are separated by semicolons, or by returns.

```
ml=[1 2 3; 4 5 6]
```

or

```
m2=[1 2 3

4 5 6]
```

To define character variables we use the equal sign and single quotes.

```
c=’abc’
```

Examining Numbers and Characters

who	This command lists the names of all currently defined variables.
size	Size (name) returns the size of the variable (name).
clear	Erases the values of all defined variables.

Workspace Commands

save	Saves all of the defined variables into a workspace called matlab.mat.
save (name)	Saves variables to a workspace called name.mat.
load	Loads the workspace matlab.mat.
load (name)	Loads the workspace name.mat.
diary	Saves everything you see in the Matlab window into a text file.
help	Lists all of the available operators and functions.
help (cmd)	Lists the help file for (cmd).

Mathematical Functions

Basic Math

+	Is the addition operator.
-	Is the subtraction operator.
*	Performs multiplication.
/	Performs right division. (b/a = b*Inv(a))
\	Performs left division. (b\a= Inv(a)*b)
^	Performs exponentiation.

For scalar-scalar or scalar-matrix operations these perform as we expect them to. For matrix-matrix operations Matlab assumes it is attempting to perform matrix math operations. To perform element to element operations you must place a . before the operator.

Note: if the left hand argument is an integer, there should be a space between the integer and the . operator, or else two ..’S.

A Little Higher Math

Matlab has a wide variety of built in higher math functions. We won't be able to cover all of them here, but for further information you can consult the on-line demo, and the tutorial and reference sections of the Math-Works Pro-MATLAB manual.

Polynomials

There are a number of operations we may perform involving polynomials. One of these is root finding. Matlab will automatically find the roots of a polynomial with the roots function. The syntax is:

```
roots(coeff)
```

where coeff is a vector of the polynomial's coefficients ordered by descending powers of x.

Polynomials are also useful in data fitting. Given two vectors of experimental points we can fit different orders of polynomial to them using **polyfit**, and **polyval**. **polyfit(x,y,n)** returns the coefficients of a polynomial of order n that fits x to y. **polyval(c,x)** returns the value of a polynomial in x with coefficients c. Try out polyfit and polyval on the vectors xdata and ydata in the workspace test1. Compare the ydata values and the yexpected values for several orders of polynomial.

Linear Equation Solving

Matlab is very handy to solve series of linear equations. We define the equation coefficients as a matrix, and the scalar answers as a transposed vector. For instance, if we wanted to solve the following equations:

3x1 +5x2=6
4x1 - 8x2=1

we would define
mat=[3 5; 4 -8]
vec=[6 1]

To solve we need to divide. However, the matrix inner dimensions do not agree, so we can't. We need to transpose the vector, using a single quote.

vec=vec’

Then we can divide
mat \vec

and solve the equations.

Graphing

Creating the Graph

The basic graphics command in Matlab is **plot**. Plot will take multiple arguments, always in x,y pairs. For instance:
plot(x,y,x,z,w,d)

will plot 3 curves: y versus x, z versus x and d versus w. Each of the lines will be a different linetype. The axes will be set to accomodate all 3 lines fully. Each new plot command clears the screen before plotting. If you wish to overlay curves made from multiple plot commands you need to use the **hold** command. It may be used alone, as a toggle, or with on and off to specify hold on, hold off. To plot points, instead of lines, you give the plot command an extra argument, that being the point style you wish to use. For instance:
plot(x,y,'o',x,z,'+')
will plot y versus x points as open circles, and z versus x points as +'s.

Warning: Matlab will automatically furnish you with a graphics window. Do NOT kill this window. It will exit automatically when you leave Matlab. If you kill the graphics window and then try to plot again, Matlab will spontaneously exit, taking all your hard work with it.

The graph may also be labeled. To label the x and y axes we use the **xlabel** and **ylabel** commands. For a title, we use the title command.

xlabel(‘*This is the x axis')*

ylabel (‘*This is the y axis')*

title (‘ *Nifty y x plot.')*

Printing the Graph

Printing is extremely simple in Matlab. All you need to do, once the graph is just how you want it, is type **print** in the Matlab window. Matlab will automatically select the default laser printer. On some systems you may be charged for this printing.
If you would rather save your plot to a file, the command of interest is **meta**. Simply type meta in the Matlab window, and your plot will be saved to a file called matlab.met. If this file already exists, your plot will be appended to the end of it. To print this file you will need to convert into a filetype that your printer of choice can interpret. Matlab comes packaged with a graphics post processor for just this purpose. Typing !gpp in the matlab window will give a fairly self explanatory manual page for gpp.
Note: ! is the shell escape for matlab. Thus, if you would prefer, the gpp stuff can be done in an xterminal.

M-files

Scripts and Functions

Matlab can also be used as a programming language. To program in Matlab you simply create a text file containing Matlab commands exactly as you would type them interactively in the Matlab window. The file may have any legal unix name, and should end with a .m extension. These files may be placed in your root directory, or a directory named /matlab. (Any other directories would have to be explicitly added to your Matlabpath.)
There are two types of m-files in matlab. One is called a script. This is simply a list of Matlab commands with no header. The other type is a function. Functions have a header line that may look something like:

function *y=fun1(x)*

Functions may be passed arguments, and may return results. To invoke a script or a function you simply type the filename (without the .m extension) into the Matlab window. You will also notice that m-files which you create are included in the help listing. If you perform a help on a specific m-file, help will return any comments which appear before the first line of actual code in the m-file.

Control Flow

Matlab does have flow control statements. Although these may be used in an interactive fashion, they are frequently less confusing and generally more useful in conjuntion with m-file programming. There are three main constructions.

For Loops

To iterate in Matlab we may employ a “for” loop. The syntax is:

```
for i = 1:n,
v(i)=i+2
end
```

While

While statements employ the logical operators. For instance:

```
while n˜=1
n=n/2
end
```

If Then

Matlab also has the conditional if, elseif, then statements. For example,

```
if i ==j,
a(i,j) = 2;
elseif (i-j) == 1,
a(i,j) = -1;
else
a(i,j) = 0;
end
```

Function Functions

There is a class of Matlab routines which require you to define a function to use them. Several of them are worth examining.

Numerical Integration

There are two numerical integration functions built into Matlab. These are **quad** and **quad 8**. The syntax for both of them is:

quad *(‘f’,a,b)*

Where a and b define the range to integrate the function over, and 'f’ comes from an m-file called f.m which returns the value of f(x) .

Finding Zeros

There are two routines for finding the zeros of functions. These are **fmin**, for 1 variable functions, and **fmins**, for multivariable functions. Fmin has the same syntax as quad and quad8. Since fmins uses a simplex routine it needs only a starting point, not a range to work in.

fmin *(‘f’,a,b)*

fmins *(‘f’,x)*

Differential Equation Solving

Matlab is equipped with 2 numerical differential equation solvers. These are **ode23** and **ode45**. These routines take the following arguments:

ode45('yprime',t0, tf, y0)

where yprime is an m-file containing the system of ode's, t0 and tf are the interval to solve over, and y0 are the initial conditions to apply.

Problems or Questions

Faculty, Staff, and Graduate Students:

If you have a problem, contact your computing support representative by sending an e-mail message to problem@rice.edu detailing your question. Your query is examined by a staff dispatcher for severity and assigned to the appropriate staff. This is the most effective communication method since computing support staff are often working in the field and unreachable by phone. In addition, the dispatcher is aware of who is on vacation or out ill.

Undergraduates:

If you have a problem, contact your computing support representative by sending an e-mail message to problem@rice.edu detailing your question. Your query is automatically assigned to your College Computing Associate (CCA).

If you need immediate assistance during normal business hours, you can call the Consulting Center at 713.348.4983. During the semester, the Consulting Center has limited evening and weekend hours as well.

To report emergencies, which are urgent system-wide problems (i.e.: all Wiess' network connections are down or all the PCs in a lab are non-functional), contact the Operations Center at 713.348.4989. Staff work 24 hours a day, 365 day a year and can page appropriate administrators for major network or computing problems.

More information is available at http://www.rice.edu/Computer/student.html