Matlab Guide

Dr. Richter
Cleveland State University, Dept of Mechanical Engineering

September 15, 2008

1 Introduction

Matlab is a very powerful software package intended for scientific and engineering computation. It is widely used in industry, academia and government organizations, worldwide. Matlab functionality is contained in three broad areas:

1. Core package: A wide variety of standard mathematical functions: root finding, matrix manipulations etc. Scripiting language (m-files).
2. Toolboxes: Specialized packages for control design, statistics, image processing, finance, digital signal processing and much more.
3. Simulink: Graphical environment for simulation of dynamic systems.

As you will appreciate soon, one of the most convenient features of Matlab is a unified syntax for real and complex numbers, scalars, vectors and matrices.

1.1 Obtaining Help

At the prompt, you can simply type `help` followed by the command name when you know it. If you don't know the command name for a particular operation, just type `help` and locate the topic containing the operation. For example, I don't know the command name for finding the hyperbolic arc-cosecant of a number. I type `help` and I see that elementary math functions are in `matlab\elfun`, so I type `help matlab\elfun`. Now I see that the command is `acsch`. For details on syntax, I type `help acsch`. Of course, you can also use the standard help on the menus and search and navigate. You can use [1] and [2] for more details.

2 Elementary Math

Let's add two numbers:

```
>>1+1
```

Let's suppress the output:

```
>>1+1;
```

Let's create a matrix:

```
>>A=[1 2; 3 4]
```

As you see, rows are separated by semicolons, and entries within a row are separated by spaces, optionally with commas. Let's create a complex column vector:

```
>>v=[1+2*i;3-pi*i]
```

Note that if the variable `i` hasn't been defined before, it is assumed by Matlab to be $\sqrt{-1}$. The same is true for the symbol `j`. Now multiply the matrix and the vector just as if they were real scalars:

```
>>A*v
```

Now let's learn how to reference matrix entries. Let's create some matrix:

```
>>M=rand(5,5);
```

Let's reference all elements on the third column:

```
>>M(:,3)
```

Let's reference all elements on the second row:

```
>>M(2,:)
```

Let's sum the elements on the third row:

```
>>sum(M(3,:))
```

Let's invert the matrix

```
>>inv(M)
```

Type `help matlab\matfun` to see how to find determinants, norms and how to perform a variety of matrix operations.

3 Program Control Structures

We'll only show how to do a "for" loop and the "if" condition here. Type `help while` and `switch` for examples of other program control structures. You may use a "for" loop at the prompt or within a Matlab script, which we'll see later. Find the trace (sum of diagonal elements) of M:

```
>>s=0;
>>for i=1:5,
s=s+M(i,i);
end
>>s
```

Now compare your calculation with the output of the built-in command for traces:

```
>>s>trace(M)
>>s<trace(M)
>>s==trace(M)
>>s>=trace(M)
>>s~=trace(M)
```

Note that the `~=` operator means "different than". For a list of relational and logic operators type `help relop`. The following lists all even numbers less than or equal to 100 that have integer square roots. It shows you how to count by twos using "for", how to detect integer numbers and how to use the "if' statement.

```
>>for i=0:2:100,
i
if round(sqrt(i))==sqrt(i),
disp('is a perfect square')
else
disp('is not a perfect square')
end   %end for the if
end   %end for the for
```

Now use the `find` command to find the elements of an array satisfying a criterion:

```
>>vector=[4 3 2 1];
>>find(vector==2)
>>aux=find(vector>=5)
>>isempty(aux)
```

Now let's control the dispayed precision. Please note that Matlab always works internally with double precision, but the number of decimals displayed can be adjusted.

```
>>number=rand(1,1)
>>format long
>>number
>>format short
>>number
```

4 Element-wise operations

The `*` operator is used for general matrix multiplication (remember that a plain number or a vector are special cases of matrices) and the user is responsible for dimensional compatibility:

```
>>v*A
```

Remember that vector multiplication is given by the inner product. You can obtain the inner product using `dot`. For 3D vectors only, you can calculate the cross product with `cross`. In some cases, you will need to perform element-wise multiplication between vectors or to apply an inherently scalar operation to all elements of a vector. Then you must use the special syntax forms `.*`, `.^` and `./`. Here are some examples. Create two vectors and obtain a third one by element-wise multiplication:

```
>>aa=[0:2:10];
>>bb=[5:-1:0];
>>cc=aa.*bb
```

You're responsible for making sure both vectors have the same length, which you can check with guess which command [1]. Now let's raise the elements of one vector to a power given by the elements of another. Finally, find the reciprocal of each element of the result:

```
>>bases=[-1:5];
>>exponents=[1:7];
>>res1=bases.^exponents
>>reciprocals=1./res1
```

5 String Manipulation

Some self-explanatory string manipulations:

```
>>msg='Hello, world';
>>msg(2)
>>vect=double(msg) %gives you the ASCII code of each cl
>>same_msg=char(vect) %the inverse operation
>>other_msg=' goodbye, world';
```

[1] length

```
>>final_msg=strcat(msg,other_msg)
>>length(final_msg)
```

6 Functions and M-files

Let's convert our commands that find perfect squares into a program that tells how many perfect squares are found. The main function will take two arguments: the upper limit (i.e, 100) and the step (i.e., 2). The auxiliary function `is_perf_sq` just tells us if a given number is a perfect square. This is all very simple, but the point is to learn how to write Matlab programs. Open the editor and type the following:

```
function yes_no=is_perf_sq(number)
yes_no=0;
if round(sqrt(number))==sqrt(number),
     yes_no=1;
end
```

It is *very* important that you save the function using its name as filename, that is, you should save the script as `is_perf_sq.m`. Now create another m-file with the contents:

```
function xx=how_many(limit, step)
accum=0;
for i=0:step:limit,
  if is_perf_sq(i),
     accum=accum+1;
  end
end
xx=accum;
```

Save it with you know what filename. Now test it:

```
>>how_many(100,2)
```

All variables in an m-file are local. The `who` command displays all variables in the workspace. As you see, `accum` is not there, nor are there the other local variables. To clear a variable type `clear` followed by the variable name. Careful, `clear` alone clears all workspace variables. To clear the screen use `clc`.

7 Graphics and plotting

Matlab is very powerful for specialized graphics. The manual is hundreds of pages long, so we'll just cover the very basics. Create x axis data and evaluate the function `sin` at every point. Then plot.

```
>>x=[-pi:0.1:pi];
>>y=sin(x);
>>plot(x,y)
>>axis([-pi pi -1 1])
>>title('Sinewave')
>>xlabel('Angle \theta')
>>ylabel('sin(\theta)')
>>grid
>>grid off
```

Now let's plot two series of data on a semilog chart and create a legend:

```
x=logspace(-1,1,10) %type help logspace
y1=log10(x);
y2=2*log10(x);
semilogx(x,y1,'r*',x,y2,'b^')
grid
legend('First data set','Second data set')
```

Explanation: The `logspace(d1,d2,N)` command creates a vector of N points starting with 10^{d_1} and ending in 10^{d_2}, such that their logarithms are equally spaced. The command for decimal logs is `log10`, while `log` is used for natural logs. The `semilogx` uses a logarithmic scale for the x axis only. Otherwise, it works the same as `plot`. The argument in quotes after the x-y pair is a string that selects the character to be used in the plot and the initial of the color name. For example `'b^'` plots blue triangle symbols. For black use k. Try adding a hyphen to the string for connecting the characters with a line (`'-b^'`). The associations in `legend` are in the order of plotting. You can edit a plot by clicking on the figure window. Note that a new plot replaces previous plots. If you want to add a data set after the plot is done, you can issue the command `hold on`. Finally, let's do 4 copies of the same 3D plot.

```
>>[X,Y]=meshgrid(-1:0.2:1);
>>Z=X^2-X.*Y+Y^2;
>>subplot(2,2,1)
>>mesh(X,Y,Z)
>>subplot(2,2,2)
>>mesh(X,Y,Z)
>>subplot(2,2,3)
>>mesh(X,Y,Z)
>>subplot(2,2,4)
>>mesh(X,Y,Z)
```

Explanation: `meshgrid` creates specially formatted matrices X and Y so that the calculation used for Z takes all possible combinations between elements of

X and Y (a grid of points on the X-Y plane, if you will). Note that we must use the `.*` syntax for multiplication. The first two arguments to `subplot` are the number of rows and columns for the plot array. The third indicates the active plot. Numbering is from left to right and from top to bottom, as you can see. Feel free to use the interactive zoom and rotation tools.

8 Importing and Saving Data

To save all commands and workspace replies in a session, type `diary` when you want to start logging. To stop logging type `diary` again. A text file will be created in the current directory with all of your commands. To see the current directory type `pwd`. To see the files in the current directory quickly, type `dir`. Try it by typing `diary` now and typing in a few commands. Then turn the diary off and look for the file it created and inspect its contents using, for instance the Matlab editor. The above only saves commands. Suppose you want to save variables `x` and `y` from before (hopefully you haven't cleared them, check with `who`). Type `save mydata x y`. This will create a file named **mydata.mat** in the current directory. Now do clear all variables with `clear`. If you type `who`, there should be nothing there. Load your data by typing `load mydata` and type `who` again. If you had typed just `save mydata`, all variables in the workspace would have been saved. Finally, suppose you captured some data from an experiment and you want to import it into Matlab. Go to Excel and create some uniform columns of data (the data must be a rectangular block). In Excel, export the data to a plain text file (tab-separated) named `export_data.txt`. (actually any extension will do). Place the file in the current Matlab directory (use `pwd` to see it). Now type `load -ascii export_data.txt`. If you type `who`, you should see a variable with the same name as the file name. Type `size(export_data)` to see how many rows and columns exist. To display the data just invoke its name by typing `export_data`.

9 Exercises for Experienced Users

- Make a surface plot of the function $z = x\sin(4xy)$ and use 3D rotation to look at it.
- Find a symbolic solution to the linear system of equations in x, y and z:

$$ax + by + cz = 0$$
$$dx + ey + fz = 0$$
$$gx + hy + iz = 1$$

  You have to use the maple feature and matrix operations.

- Try `der=diff('b*a^2',a)`. What does this operation accomplish?
- Now assign a value to `a` and `b` and type `eval(der)`.

References

[1] A. Gilat. *Matlab: An Introduction with Applications*. Wiley, 2004.

[2] B. Hahn. *Essential Matlab for Scientists and Engineers*. Butterworth and Heinemann, 2002.