

Computer Science/Mathematics 555

Matlab Tutorial

MATLAB is very powerful and varied software package for scientific computing. It is based upon matrices and m-files.

It is invoked by typing

```
% matlab
```

I recommend the creation of a directory called “matlab.” Use the two commands

```
% mkdir matlab (done only once)
```

```
% cd matlab
```

before invoking matlab. Put all .m file in this directory.

Matrices are the main data element. They can be introduced in the following four ways.

1. As an explicit list of elements.
2. Generated by built-in statements or functions.
3. Created by m-files.
4. Loaded from external data files.

For instance consider the MATLAB command

```
>> A = [ 1 2 3 ; 4 5 6 ; 7 8 9]
```

It immediately outputs

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

(no parentheses or brackets). Or if you create a file called gena.m and type

```
A = [ 1 2 3 ; 4 5 6 ; 7 8 9]
```

in that file.

Then type

```
>> gena
```

it will also produce A exactly as above. The file gena.m is called a script file.

The line
» gena;

will also produce A , but will not print it on the screen.

It is important to remember the distinction between row and column vectors in MATLAB. If you type
» $x = [-1.3 \text{ sqrt}(3) (1+2+3)*4/5]$
you get

$$x = -1.3 \quad 1.7321 \quad 4.8000$$

Row vectors are default.

On the other hand, if you type
» $x = [-1.3; \text{sqrt}(3); (1+2+3)*4/5]$
you get

$$x = \begin{array}{c} -1.3 \\ 1.7321 \\ 4.8000 \end{array} .$$

The most important MATLAB command is “help.”

» help

This lists all help topics

» help inv

Gives the help information for the function “inv” for inverting a matrix.

» help help

Explains the “help” command.

MATLAB m-files contain two types of items

1. scripts
2. user-defined functions

The gena.m file described above is an example of a script. It just lines of text consisting of MATLAB commands. Invoking the script (done when typing “gena” or the name of the script file without the “.m”) is equivalent to typing the commands directly into MATLAB.

Functions have “function” statements as their first line. Two examples are

```
function A=fun1(x,y,z)
function [A,B,C]=fun2(x,y,z)
```

These should be placed in files called “fun1.m” and “fun2.m” respectively. Here “x,y,z” are input arguments, and “A,B,C” are output arguments. Any of these can be simple variables, vectors, or matrices. For examples of functions look at the files “cosx.m” and “bisect2.m” on the class web page.

MATLAB supports the usual programming structures “if” , “for” , “while” and “switch.” The syntax is a bit different from C/C++ syntax, but the use is much the same. Information about any of them may be obtained from the help command. The function “bisect2.m” contains “if” and “while” statements. The following “for” statement sums the first n integers.

```
sum=0;
for k=1:n
    sum=sum+k;
end;
```

Moreover, many C/C++ structures are supported in MATLAB. The I/O commands “fscanf” and “fprintf” are like their C counterparts *except that they can deal with matrix and vector input*. These allow you to do input and output with files.

Two useful MATLAB command are “save” and “load.” MATLAB creates a workspace for all of your variables. When you are done with a MATLAB session, you type

```
>> quit
```

and you are out of MATLAB. However, your workspace and all of the variables that you are using vanish, they will not be there when you log on to MATLAB again.

If, before you quit, you type

```
>> save mysession
```

the workspace will be saved in the file “mysession.mat.” If you type

```
>> save mysession X Y Z
```

then only the variables X , Y , and Z will be saved. “save” is a completely destructive operation, “mysession.mat” is completely overwritten.

If you want to retrieve your workspace from the previous session type
» load mysession

To know what variables you have type
» who

Most matrix operations in MATLAB are one line.

The matrix transpose operation is

» $B = A'$;

Thus if A has the contents

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{pmatrix}$$

then B will have the contents

$$B = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 0 \end{pmatrix}$$

Here are some other functions

» $C = A+B$

just adds A and B .

» $C = A*B$

just multiplies A and B .

» $C = X'*Y$

does the operation $C = X^T Y$.

» $b = A*x$

is a matrix–vector multiply. If the dimensions are wrong in any of these operations, you will get an error message.

There are two important functions associated with matrices, “size” and “length.” The statement

» $[m,n]=size(A)$

returns the dimensions of the array A . If $m = 1$, A is just a row vector, if $n = 1$ then A is column vector, and if $m = n = 1$, then A is scalar.

The statement

» $p=length(A)$

returns the larger of the two outputs of “size.”

There are a number of ways to generate special vectors and matrices. Here are some examples

```
>> x = 1:5
```

yields

$$x = 1 \ 2 \ 3 \ 4 \ 5 .$$

```
>> y = 0:pi/4:pi
```

yields

$$y = 0 \ \pi/4 \ \pi/2 \ 3\pi/4 \ \pi .$$

```
>> z = 6:-1:1
```

yields

$$z = 6 \ 5 \ 4 \ 3 \ 2 \ 1 .$$

The statement

```
>> y =exp(z)
```

yields

$$y = e^6 \ e^5 \ e^4 \ e^3 \ e^2 \ e^1 .$$

Actually, it produces the appropriate floating point numbers. Of course, “exp” is exponential function.

MATLAB supports lots of math functions, such as exp, log, log10, sin, cos, tan, asin, atan, and some you have probably never heard of.

It also support many special operations that will become useful later.

So far, I have given a small portion of MATLAB. Throughout the semester, I will give you more pieces of it. MATLAB is set up so that a computer literate person can learn much on his/her own.

We will make some use of the Optimization Toolbox. This is NOT available in the student version of MATLAB that you can purchase at the bookstore. To learn about this type

```
>> tutdemo
```

and a number of optimization that use the package will be shown to you. However, this package does too much for our purposes. The ideas we will be talking about all semester will be hidden from you. Thus you will

mostly use these tools to verify that you have the correct answer! The important routines there are **fminunc** (unconstrained minimization), **fmincon** (constrained minimization), and **optimset** creates options for **fminunc** and **fmincon**. Since I will mostly have you writing codes on your own, you won't actually use these to solve your problems.