

AAE 320 / ME 345
A brief introduction to MATLAB
Written by Burak Ozturk
Last modification: January 18, 2001

MATLAB is an interactive, matrix-based system for scientific and engineering calculations. You can solve complex numerical problems without actually writing a program. The name MATLAB is an abbreviation for MATRix LABoratory.

The purpose of this document is to help you begin to use MATLAB. You are encouraged to work at the computer as you read this document and freely experiment with examples. You should liberally use the on-line help facility for more detailed information. After entering MATLAB as described in section 1, the command “help” will display a list of functions for which on-line help is available; the command “help function_name” will give information about a specific function. The command “help eig”, for example, will give information about the eigenvalue function ‘eig’.

MATLAB is licensed by The MathWorks, Inc., Cochituate Place, 24 Prime Park Way, Natick, MA 01760, (508)653-1415, Fax: (508)653-2997, Email: info@mathworks.com, URL: www.mathworks.com.

1. Accessing MATLAB

On most systems, after logging in one can enter MATLAB with the system command “matlab” and exit MATLAB with the command “exit” or “quit”. On a PC, for example, if properly installed, one may enter MATLAB with the command:

```
C> matlab
```

and exit it with the command:

```
>> quit
```

On the EWS machines, you should use the same commands:

```
cehpx22> matlab
```

```
>> exit
```

or

```
>> quit
```

2. Entering matrices

MATLAB works with essentially only one kind of object---a rectangular numerical matrix with possibly complex entries; all variables represent matrices. In some situations, 1-by-1 matrices are interpreted as scalars and matrices with only one row or one column are interpreted as vectors.

Matrices can be introduced into MATLAB in several different ways:

- Entered by an explicit list of elements,
- Generated by built-in statements and functions,
- Created in M-files (see Sections 11 and 13 below),

For example, either of the statements

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

and

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

create the obvious 3-by-3 matrix and assigns it to a variable A. Try it. The elements within a row of a matrix may be separated by commas as well as by a blank.

When listing a number in exponential form (e.g. 2.34e-9), blank spaces must be avoided. Listing entries of a large matrix is best done in an M-file, where errors can be easily edited away (see Sections 11 and 13).

The command `rand(n)` will create an $n \times n$ matrix with randomly generated entries distributed uniformly between 0 and 1, while `rand(m,n)` will create an $m \times n$ one.

Individual matrix and vector entries can be referenced with indices inside parentheses in the usual manner. For example, `A(2,3)` denotes the entry in the second row, third column of matrix A and `x(3)` denotes the third coordinate of vector x. Try it. A matrix or a vector will only accept positive integers as indices.

3. Matrix operations, array operations

The following matrix operations are available in MATLAB:

+	Addition
-	Subtraction
*	Multiplication
^	Power
'	Transpose
\	Left division
/	Right division

These matrix operations apply, of course, to scalars (1-by-1 matrices) as well. If the sizes of the matrices are incompatible for the matrix operation, an error message will result, except in the

case of scalar-matrix operations (for addition, subtraction, and division as well as for multiplication) in which case each entry of the matrix is operated on by the scalar.

The "matrix division" operations deserve special comment. If A is an invertible square matrix and b is a compatible column, resp. row, vector, then

$x = A \setminus b$ is the solution of $A * x = b$ and, resp.,
 $x = b / A$ is the solution of $x * A = b$.

In left division, if A is square, then it is factored using Gaussian elimination and these factors are used to solve $A * x = b$. If A is not square, it is factored using Householder orthogonalization with column pivoting and the factors are used to solve the under- or over-determined system in the least squares sense. Right division is defined in terms of left division by $b / A = (A' \setminus b)'$.

Array operations: The matrix operations of addition and subtraction already operate entry-wise but the other matrix operations given above do not--they are matrix operations. It is important to observe that these other operations, $*$, $^{\wedge}$, \setminus , and $/$, can be made to operate entry-wise by preceding them by a period. For example, either $[1,2,3,4].*[1,2,3,4]$ or $[1,2,3,4].^{\wedge}2$ will yield $[1,4,9,16]$. Try it. This is particularly useful when using Matlab graphics.

4. Statements, expressions, and variables; saving a session

MATLAB is an expression language; the expressions you type are interpreted and evaluated. MATLAB statements are usually of the form

variable = expression, or simply
expression

Expressions are usually composed from operators, functions, and variable names. Evaluation of the expression produces a matrix, which is then displayed on the screen and assigned to the variable for future use. If the variable name and = sign are omitted, a variable `ans` (for answer) is automatically created to which the result is assigned.

A statement is normally terminated with the carriage return. However, a statement can be continued to the next line with three or more periods followed by a carriage return. On the other hand, several statements can be placed on a single line if separated by commas or semicolons.

If the last character of a statement is a semicolon, the printing is suppressed, but the assignment is carried out. This is essential in suppressing unwanted printing of intermediate results.

MATLAB is case-sensitive in the names of commands, functions, and variables. For example, `solveUT` is not the same as `solveut`.

The command who will list the variables currently in the workspace. A variable can be cleared from the workspace with the command clear variable name. The command clear alone will clear all nonpermanent variables.

5. Matrix building functions

Convenient matrix building functions are

eye	Identity matrix
zeros	Matrix of zeros
ones	Matrix of ones
rand	Randomly generated matrix

For example, zeros(m,n) produces an m-by-n matrix of zeros and zeros(n) produces an n-by-n one; if A is a matrix, then zeros(A) produces a matrix of zeros of the same size as A.

Matrices can be built from blocks. For example, if A is a 3-by-3 matrix, then

```
B = [A, zeros(3,2); zeros(2,3), eye(2)]
```

will build a certain 5-by-5 matrix. Try it.

6. For, while, if --- and relations

In their basic forms, these MATLAB flow control statements operate like those in most computer languages.

For: For example, for a given n, the statement

```
x = [ ]; for i = 1:n, x=[x,i^2], end
```

or

```
x = [ ];  
for i = 1:n  
    x = [x,i^2]  
end
```

will produce a certain n-vector and the statement

```
x = [ ]; for i = n:-1:1, x=[x,i^2], end
```

will produce the same vector in reverse order. Try them. Note that a matrix may be empty (such as x = []).

While: The general form of a while loop is

```
while relation
    statements
end
```

The statements will be repeatedly executed as long as the relation remains true. For example, for a given number a , the following will compute and display the smallest nonnegative integer n such that $2^n \geq a$:

```
n = 0;
while 2^n < a
    n = n + 1;
end
n
```

If: The general form of a simple if statement is

```
if relation
    statements
end
```

The statements will be executed only if the relation is true. Multiple branching is also possible, as is illustrated by

```
if n < 0
    parity = 0;
elseif rem(n,2) == 0
    parity = 2;
else
    parity = 1;
end
```

In two-way branching the elseif portion would, of course, be omitted.

Relations: The relational operators in MATLAB are

<	Less than
>	Greater than
<=	Less than or equal
>=	Greater than or equal
==	Equal
~=	Not equal

Note that "=" is used in an assignment statement while "==" is used in a relation. Relations may be connected or quantified by the logical operators

&	and
	or
~	not

A relation between matrices is interpreted by while and if to be true if each entry of the relation matrix is nonzero. Hence, if you wish to execute statement when matrices A and B are equal you could type

```
if A == B
    statement
end
```

but if you wish to execute statement when A and B are not equal, you would type

```
if any(any(A ~ = B))
    statement
end
```

or, more simply,

```
if A == B else
    statement
end
```

Note that the seemingly obvious

```
if A ~ = B, statement, end
```

will not give what is intended since statement would execute only if each of the corresponding entries of A and B differ. The functions any and all can be creatively used to reduce matrix relations to vectors or scalars. Two any's are required above since any is a vector operator (see Section 8).

7. Scalar functions

Certain MATLAB functions operate essentially on scalars, but operate element-wise when applied to a matrix. The most common such functions are sin, asin, exp, abs, round, cos, acos, log (natural log), sqrt, tan, atan.

8. Vector functions

Other MATLAB functions operate essentially on a vector (row or column), but act on an m-by-n matrix ($m \geq 2$) in a column-by-column fashion to produce a row vector containing the results of their application to each column. Row-by-row action can be obtained by using the transpose; for example, `mean(A')`. A few of these functions are max, sum, median, min, prod, mean.

For example, the maximum entry in a matrix A is given by $\max(\max(A))$ rather than $\max(A)$. Try it.

9. Matrix functions

Much of MATLAB's power comes from its matrix functions. The most useful ones are

eig	eigenvalues and eigenvectors
inv	inverse
det	determinant
size	size

MATLAB functions may have single or multiple output arguments. For example,

$$y = \text{eig}(A), \text{ or simply } \text{eig}(A)$$

produces a column vector containing the eigenvalues of A while

$$[U,D] = \text{eig}(A)$$

produces a matrix U whose columns are the eigenvectors of A and a diagonal matrix D with the eigenvalues of A on its diagonal. Try it.

10. Command line editing and recall

The command line in MATLAB can be easily edited. The cursor can be positioned with the left/right arrows and the Backspace (or Delete) key used to delete the character to the left of the cursor. Other editing features are also available.

A convenient feature is use of the up/down arrows to scroll through the stack of previous commands. One can, therefore, recall a previous command line, edit it, and execute the revised command line.

11. M-files

MATLAB can execute a sequence of statements stored on diskfiles. Such files are called "M-files" because they must have the file type of ".m" as the last part of their filename. Much of your work with MATLAB will be in creating and refining M-files.

There are two types of M-files: script files and function files.

Script files: A script file consists of a sequence of normal MATLAB statements. If the file has the filename, say, rotate.m, then the MATLAB command "rotate" will cause the statements in the file to be executed.

E.g., to call a script file “script.m”, type the name of the script file without the “.m” extension at the MATLAB prompt, as follows:

```
>> script
```

Variables in a script file are global and will change the value of variables of the same name in the environment of the current MATLAB session.

Script files are often used to enter data into a large matrix; in such a file, entry errors can be easily edited out. If, for example, one enters in a diskfile data.m

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

then the MATLAB statement data will cause the assignment given in data.m to be carried out. An M-file can reference other M-files, including referencing itself recursively.

Function files: Function files provide extensibility to MATLAB. You can create new functions specific to your problem which will then have the same status as other MATLAB functions. Variables in a function file are by default local.

We first illustrate with a simple example of a function file.

```
function a = randint(m,n)  
%RANDINT Randomly generated integral matrix.  
% randint(m,n) returns an m-by-n such matrix with entries  
% between 0 and 9.  
a = floor(10*rand(m,n));
```

This should be placed in a diskfile with filename randint.m (corresponding to the function name). The first line declares the function name, input arguments, and output arguments; without this line the file would be a script file. Then a MATLAB statement

```
z = randint(4,5),
```

for example, will cause the numbers 4 and 5 to be passed to the variables m and n in the function file with the output result being passed out to the variable z.

Since variables in a function file are local, their names are independent of those in the current MATLAB environment.

Note that use of nargin (“number of input arguments”) permits one to set a default value of an omitted input variable---such as a and b in the example.

A function may also have multiple output arguments. For example:

```
function [mean, stdev] = stat(x)  
% STAT Mean and standard deviation  
% For a vector x, stat(x) returns the
```

```

% mean and standard deviation of x.
% For a matrix x, stat(x) returns two row vectors containing,
% respectively, the mean and standard deviation of each column.
[m n] = size(x);
if m == 1
    m = n; % handle case of a row vector
end
mean = sum(x)/m;
stdev = sqrt(sum(x.^2)/m - mean.^2);

```

Once this is placed in a diskfile stat.m, a MATLAB command `[xm, xd] = stat(x)`, for example, will assign the mean and standard deviation of the entries in the vector `x` to `xm` and `xd`, respectively. Single assignments can also be made with a function having multiple output arguments. For example, `xm = stat(x)` (no brackets needed around `xm`) will assign the mean of `x` to `xm`.

Important: The `%` symbol indicates that the rest of the line is a comment; MATLAB will ignore the rest of the line. However, the first few comment lines, which document the M-file, are available to the on-line help facility and will be displayed if, for example, `help stat` is entered. Such documentation should always be included in a function file.

This function illustrates some of the MATLAB features that can be used to produce efficient code. Note, for example, that `x.^2` is the matrix of squares of the entries of `x`, that `sum` is a vector function (section 8), that `sqrt` is a scalar function (section 7), and that the division in `sum(x)/m` is a matrix-scalar operation.

The following function, which gives the greatest common divisor of two integers via the Euclidean algorithm, illustrates the use of an error message (see the next section).

```

function a = gcd(a,b)
% GCD Greatest common divisor
% gcd(a,b) is the greatest common divisor of
% the integers a and b, not both zero.
a = round(abs(a)); b = round(abs(b));
if a == 0 & b == 0
    error('The gcd is not defined when both numbers are zero')
else
    while b ~= 0
        r = rem(a,b)
        a = b; b = r;
    end
end
end

```

Important: Some of MATLAB's functions are built-in while others are distributed as M-files. The actual listing of any M-file---MATLAB's or your own---can be viewed with the MATLAB command “`type functionname`”. Try entering `type eig`, `type vander`, and `type rank`.

12. Text strings, error messages, input

Text strings are entered into MATLAB surrounded by single quotes. For example,

```
s = 'This is a test'
```

assigns the given text string to the variable `s`.

Text strings can be displayed with the function `disp`. For example:

```
disp('this message is hereby displayed')
```

Error messages are best displayed with the function `error`

```
error('Sorry, the matrix must be symmetric')
```

since when placed in an M-File, it causes execution to exit the M-file.

13. Managing M-files

While using MATLAB one frequently wishes to create or edit an M-file and then return to MATLAB. One wishes to keep MATLAB active while editing a file since otherwise all variables would be lost upon exiting.

This can be easily done using the `!`-feature. If, while in MATLAB, you precede it with an `!`, any system command---such as those for editing, printing, or copying a file---can be executed without exiting MATLAB. If, for example, the system command `ed` accesses your editor, the MATLAB command

```
>> !ed rotate.m
```

will let you edit the file named `rotate.m` using your local editor. Upon leaving the editor, you will be returned to MATLAB just where you left it.

As noted in Section 1, on systems permitting multiple processes, such as one running Unix, it may be preferable to keep both MATLAB and your local editor active, keeping one process suspended while working in the other. If these processes can be run in multiple windows, as on a workstation, you will want to keep MATLAB active in one window and your editor active in another.

When in MATLAB, the command `dir` will list the contents of the current directory while the command `what` will list only the M-files in the directory. The MATLAB commands `delete` and `type` can be used to delete a diskfile and print a file to the screen, respectively, and `chdir` can be used to change the working directory. While these commands may duplicate system commands, they avoid the use of an `!`.

M-files must be accessible to MATLAB. On most mainframe or workstation network installations, personal M-files which are stored in a subdirectory of one's home directory named matlab will be accessible to MATLAB from any directory in which one is working.

14. Comparing efficiency of algorithms: flops and etime

Two measures of the efficiency of an algorithm are the number of floating point operations (flops) performed, and the elapsed time. The MATLAB function flops keeps a running total of the flops performed. The command flops(0) (not flops = 0!) will reset flops to 0. Hence, entering flops(0) immediately before executing an algorithm and flops immediately after gives the flop count for the algorithm.

The MATLAB function clock gives the current time accurate to a hundredth of a second (see help clock). Given two such times t1 and t2, etime(t2,t1) gives the elapsed time from t1 to t2. One can, for example, measure the time required to solve a given linear system $Ax = b$ using Gaussian elimination as follows:

```
t = clock; x = A\b; time = etime(clock,t)
```

It should be noted that, on timesharing machines, etime may not be a reliable measure of the efficiency of an algorithm since the rate of execution depends on how busy the computer is at the time. But "flops" is always a respectable measure of the efficiency of your program.

15. Output format

While all computations in MATLAB are performed in double precision, the format of the displayed output can be controlled by the following commands.

```
format short      fixed point with 4 decimal places (the default)
format long       fixed point with 14 decimal place
```

Once invoked, the chosen format remains in effect until changed.

16. Hardcopy

Hardcopy is most easily obtained with the diary command. The command

```
>> diary filename
```

when typed in MATLAB prompt causes what appears subsequently on the screen (except graphics) to be written to the named diskfile (if the filename is omitted it will be written to a default file named diary) until one gives the command diary off; the command diary on will cause writing to the file to resume, etc. When finished, you can edit the file as desired and print it out on the local system. The !-feature (see section 13) will permit you to edit and print the file without leaving MATLAB.

17. Graphics

MATLAB can produce both planar plots and 3-D mesh surface plots.

Planar plots: The plot command creates linear x-y plots; if x and y are vectors of the same length, the command plot(x,y) opens a graphics window and draws an x-y plot of the elements of x versus the elements of y. You can, for example, draw the graph of the sine function over the interval -4 to 4 with the following commands:

```
x = -4:.01:4; y = sin(x); plot(x,y)
```

When in the graphics screen, pressing any key will return you to the command screen while the command shg (show graph) will then return you to the current graphics screen. If your machine supports multiple windows with a separate graphics window, you will want to keep the graphics window exposed--but moved to the side---and the command window active.

As a second example, you can draw the graph of $y = e^{-x^2}$ over the interval -1.5 to 1.5 as follows:

```
x = -1.5:.01:1.5; y = exp(-x.^2); plot(x,y)
```

Plots of parametrically defined curves can also be made. Try, for example,

```
t=0:.001:2*pi; x=cos(3*t); y=sin(2*t); plot(x,y)
```

The command grid will place grid lines on the current graph.

The graphs can be given titles, axes labeled, and text placed within the graph with the following commands which take a string as an argument.

title	graph title
xlabel	x-axis label
ylabel	y-axis label
gtext	interactively-positioned text
text	position text at specified coordinates

18. Reference

Many MATLAB features cannot be included in these introductory notes. For more detailed information about MATLAB, you can check the following web sites:

1) The official Matlab web site. It contains a wide range of documents, including the complete user manual (more than 800 pages).

<http://www.mathworks.com/products/matlab/>

2) The EWS Matlab web site, which contains a short tutorial used in the CEE Department and other tutorials created at MIT and at the University of British Columbia.

<http://www.ews.uiuc.edu/ews/software/software.cgi?matlab>