

MATLAB PRIMER

by Michael O'Sullivan

WHAT IS MATLAB?

Matlab is a software package available via your leland account. You can also run it on your PC or MAC if you have it. Matlab is designed for the manipulation and visualization of matrices and analysis of large amounts of data.

HOW DO I START MATLAB?

Once you have logged on to your leland account simply type "matlab". e.g., (Please note that all the commands are case sensitive)

```
epic15:~/eesor251>matlab
```

Matlab responds with:

```
< M A T L A B (R) >  
(c) Copyright 1984-97 The MathWorks, Inc.  
All Rights Reserved  
Version 5.1.0.421  
May 25 1997
```

To get started, type one of these commands: `helpwin`, `helpdesk`, or `demo`.
For information on all of the MathWorks products, type `tour`.

```
>>
```

The above `>>` is the Matlab prompt. Matlab commands are executed by typing them at this prompt and then striking the ENTER key. This is followed by Matlab replies. In the sequel the lines with `>>` are command lines and the rest are Matlab replies.

HOW DO I QUIT?

Type "quit" from the command line. e.g.,

```
>> quit
```

No flops.

```
epic15:~/eesor251>
```

NAVIGATING MATLAB

Typing “help general” will give information on general purpose commands useful for navigating Matlab. UNIX shell commands may also be executed by putting a ! before the command. For example, to edit the file “words.txt” using emacs while running Matlab type “!emacs words.txt”.

MATLAB VARIABLES AND THE WORKSPACE

Matlab was originally written as a matrix manipulation program, and therefore tends to try to deal with everything as a matrix. Although it is possible to input equations, assign variables, and use a lot of mathematical functions, in the end it is necessary to understand matrices to understand Matlab. To enter a small matrix, type

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

and Matlab responds with

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

This matrix can also be entered in its usual form with carriage returns replacing the semicolons.

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

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

When entering a matrix, elements are separated by a space, rows are separated by a semicolon, and the matrix is enclosed in square brackets.

The above assignment “A =” creates a variable A with the above matrix as its value.

Matlab stores variables in an initially empty workspace. Once a variable has been assigned it is stored in the workspace and may be referred to again until redefined. To get a list of variables in the workspace type “who”.

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

A =
     1     2     3
     4     5     6
     7     8     9

>> b = [1 2 3]

b =
     1     2     3

>> who

Your variables are:
A      b

or "whos" for a more detailed description

>> whos

Name      Size      Bytes      Class
-----
A         3x3         72      double array
b         1x3         24      double array

Grand total is 12 elements using 96 bytes
```

DISPLAYING AND SUPPRESSING OUTPUT (COMMENTS)

After typing a Matlab command, the output of that command is displayed.

```
>> c = 5

c =
     5
```

To execute a command and not display the output follow the command with a semicolon, e.g., typing "c;" at the command line does not display the value of c whereas typing c without the semicolon displays the value of c as the following examples illustrate.

```
>> c;

>> c

c =
     5
```

Use the symbol “%” to make comments. Anything after a % is not seen as a command, e.g.,

```
>> 5 % The rest is a comment
c =
    5
```

MATRIX OPERATIONS AND SOLVING MATRIX EQUATIONS

Scalar multiplication of matrices is supported. For example, if A is a matrix, then “ $2*A$ ” returns the matrix formed by multiplying each element of A by 2.

The usual matrix operations are supported, viz., $+$, $-$ and $*$, the last being matrix multiplication.

Exponents are also supported by $^$, e.g., $2^3 = 8$, $4^{0.5} = 2$. This works for square matrices too. For example, if A is a square matrix, A^3 returns the cube of A , i.e., $A*A*A$, and $A^{0.5}$ returns the square root of A , i.e., the matrix whose square is A . If A is also nonsingular, this works for negative integer powers of A too, e.g., A^{-2} returns the square of the inverse of A . Alternately, the built-in function $\text{inv}(A)$ also returns the inverse of A .

Division is done a little differently. If A is a nonsingular matrix and B is a matrix with the same number of rows as A , then $A \setminus b$ returns $(\text{inverse of } A)*B$. Thus, for example, the solution of the linear equations $Ax = b$ is $x = A \setminus b$. Similarly, if B is a matrix with the same number of columns as A , then B/A returns $B*(\text{inverse of } A)$. Thus, the solution to the linear equations $yA = b$ is $y = b/A$.

A matrix may be transposed with $'$, i.e., A' returns the transpose of the matrix A .

If A is a vector, $\text{max}(A)$ returns the largest element of A .

If A is a matrix, $M = \text{max}(A)$ returns the row vector M containing the maximum element in each column of A . Also, $[M, I] = \text{max}(A)$ returns both M and the indices I of the maximum elements. If there are several such indices, the index is the smallest one. For example,

```
>> A = [1 3 0
        0 4 0]
A =
    1 3 0
    0 4 0
>> [M, I] = max(A)
M =
    1 4 0
I =
    1 2 1
```

The function $\text{min}(A)$ works analogously after replacing max by min everywhere.

Other functions of a square matrix A are built-in too (see a later section), e.g., the matrix exponential $\expm(A)$ of A , the eigenvalues $\text{eig}(A)$ of A , the determinant $\det(A)$ of A , etc.

SPECIAL MATRICES (IDENTITY, ZEROS, ONES)

The function `eye(n)` returns the identity matrix of order n .

The function `zeros(m, n)` returns the $m \times n$ matrix of zeros.

The function `zeros(n)` returns the $n \times n$ matrix of zeros.

The function `ones(m, n)` returns the $m \times n$ matrix of ones.

The function `ones(n)` returns the $n \times n$ matrix of ones.

SELECTING ELEMENTS AND SUBMATRICES OF A MATRIX

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

```
>> A(2,3)
```

```
ans =  
     6
```

Submatrices may be extracted using vectors, e.g.,

```
>> u = 1:2;
```

```
>> v = [1 3];
```

```
>> A(u, v)
```

```
ans =  
     1     3  
     4     6
```

gives the submatrix defined by the first two rows and the first and third columns. An entire row or column may be selected by using a colon `:` instead of an index or a vector, e.g.,

```
>> A(:, 2)
```

```
ans =  
     2  
     5  
     8
```

gives the second column of A .

MORE MATRIX FUNCTIONS

```
>> help matlab/matfun
Matrix functions - numerical linear algebra.
Matrix analysis.
norm      - Matrix or vector norm.
normest   - Estimate the matrix 2-norm.
rank      - Matrix rank.
def       - Determinant.
trace     - Sum of diagonal elements.
null      - Null space.
orth      - Orthogonalization.
rref      - Reduced row echelon form.
subspace  - Angle between two subspaces.
:
```

Typing "help rank" will give information about the function rank, e.g.,

```
>> help rank
RANK      Matrix rank.
RANK(A) provides an estimate of the number of linearly independent
rows or columns of a matrix A.
RANK(A,tol) is the number of singular values of A that are larger
than tol.
RANK(A) uses the default tol = max(size(A))*norm(A)*eps.
Overloaded methods
help sym/rank.m
```

In general, to find out about the predefined function "func", type "help func".

GRAPHS AND PLOTTING

Matlab supports graphing 2-D plots. Suppose y is a matrix. Then `plot(y)` graphs the columns of y versus their indices.

Suppose also that x is a vector with the same number of elements that y has rows. Then `plot(x, y)` graphs the columns of y versus x .

```
>> x = -5:5
x =
    -5    -4    -3    -2    -1     0     1     2     3     4     5
>> plot(x, x.^2)
```

plots x on the x -axis and x squared on the y -axis.

The title, axis, axis labels, etc. of graphs can be altered with built-in functions.

Matlab also supports 3-D plots.

There are two areas `matlab/graph2d` and `matlab/graph3d` containing built-in functions for graphs.

CONTROL EXPRESSIONS AND LOOPING

Matlab supports `if`, `for`, and `while` statements. The following examples show how to use these expressions

```
if x == 0,  
    x = x + 1;  
elseif x == 1,  
    x = x - 1;  
else x = 0;  
end
```

```
for i = 1:10,  
    j = 10*i;  
end
```

```
i = 0;  
while i < 10,  
    i = i + 1;  
end
```

M-FILES

We can write Matlab “programs” using `m`-files. An `m`-file is a script of Matlab commands in a text file with the extension “.`m`”. For example, consider the file “`twoA.m`” with the single line

```
B = 2*A
```

When in Matlab, create any matrix `A`

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

then run the `m`-file `twoA.m` by typing

```
>> twoA
```

Matlab responds with

```
>> B
```

```
B =  
    2  4  
    6  8
```

In general, executing the m-file “file.m” is done by typing “file” in Matlab. Note that file.m must be in the current directory.

USER-WRITTEN FUNCTIONS

User-written functions are also supported by Matlab. They are similar to m-files with the following differences:

- the first line of a function file defines the function
- they accept input and output
- they don't affect the workspace

Consider the function “twoA” that produces the same result as the m-file twoA.m

```
function B = twoA(A)  
B = 2*A;
```

The first line specifies that this file defines a function “twoA” that takes one input A and returns one output B. The second line defines B to be 2*A. The function may then be used in Matlab. Create any matrix A in Matlab

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

and suppose the matrix B also exists

```
>> B = [1 1];
```

The function is called as if it were a built-in function

```
>> A = twoA(A);  
>> A
```

```
A =  
    2  4  
    6  8
```

The function multiplied A by 2 as expected. Even though B is used inside the function file, B is not affected in the workspace.

```
>> B  
B =  
    1 1
```

RANDOM NUMBERS

It is very easy to generate random numbers in Matlab as follows. The “rand” command produces uniform random numbers between 0 and 1.

```
>> rand  
ans =  
    0.9501  
rand(5,1) produces a vector of 5 uniform random numbers
```

```
>> rand(5,1)  
ans =  
    0.1987  
    0.6038  
    0.2722  
    0.1988  
    0.0153
```

randn produces a Normal(0,1) random variable (mean 0 and variance 1).

```
>> randn  
ans =  
   -0.4626
```

Do a "help stats" to get a complete list of available commands.

```
>> help stats
```

```
Statistics Toolbox.
```

```
Version 2.1.0 03-Apr-1997
```

```
New Features
```

```
  Readme      -Version 2.1.0 synopsis of new functionality.
```

```
Distributions.
```

```
Parameter estimation.
```

```
  betafit     -Beta parameter estimation.
```

```
  binofit     -Binomial parameter estimation.
```

```
  expfit      -Exponential parameter estimation.
```

```
  gamfit      -Gamma parameter estimation.
```

```
  mle         -Maximum likelihood estimation (MLE).
```

```
  :
```