MATLAB Tutorial

You need a small number of basic commands to start using MATLAB. This short tutorial describes those fundamental commands. You need to *create* vectors and matrices, to *change* them, and to *operate* with them. Those are all short high-level commands, because MATLAB constantly works with matrices. I believe that you will like the power that this software gives, to do linear algebra by a series of short instructions:

create E	*create u*	*change E*	*multiply Eu*
$E = \text{eye}(3)$	$u = E(:,1)$	$E(3,1) = 5$	$v = E * u$
$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 5 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 5 \end{bmatrix}$

The word eye stands for the identity matrix. The submatrix $u = E(:,1)$ picks out column 1. The instruction $E(3,1) = 5$ resets the $(3,1)$ entry to 5. The command $E * u$ multiplies the matrices E and u. All these commands are repeated in our list below. Here is an example of inverting a matrix and solving a linear system:

create A	*create b*	*invert A*	*solve* $Ax = b$
$A = \text{ones}(3) + \text{eye}(3)$	$b = A(:,3)$	$C = \text{inv}(A)$	$x = A\backslash b$ or $x = C * b$
$\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$	$\begin{bmatrix} .75 & -.25 & -.25 \\ -.25 & .75 & -.25 \\ -.25 & -.25 & .75 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

The matrix of all ones was added to eye(3), and b is its third column. Then inv(A) produces the inverse matrix (normally in decimals; for fractions use *format rat*). The system $Ax = b$ is solved by $x = \text{inv}(A) * b$, which is the slow way. The backslash command $x = A\backslash b$ uses Gaussian elimination if A is square and never computes the inverse matrix. When the right side b equals the third column of A, the solution x must be $[0\ 0\ 1]'$. (*The transpose symbol* ′ *makes x a column vector*.) Then $A * x$ picks out the third column of A, and we have $Ax = b$.

Here are a few comments. The comment symbol is %:

% The symbols a and A are *different*: MATLAB is case-sensitive.

% Type *help slash* for a description of how to use the backslash symbol. The word *help* can be followed by a MATLAB symbol or command name or M-file name.

Note: The command name is upper case in the description given by help, but must be lower case in actual use. And the backslash $A\backslash b$ is different when A is not square.

% To display all 16 digits type *format long*. The normal *format short* gives 4 digits after the decimal.

% A semicolon after a command avoids display of the result.

$A = \text{ones}(3)$; will not display the 3×3 identity matrix.

% Use the up-arrow cursor to return to previous commands.

How to input a row or column vector

$u = [2\ 4\ 5]$	has one row with three components (a 1×3 matrix)
$v = [2; 4; 5]$	has three rows separated by semicolons (a 3×1 matrix)
$v = [2\ 4\ 5]'$	or $v = u'$ *transposes* u to produce the same v
$w = 2{:}5$	generates the row vector $w = [2\ 3\ 4\ 5]$ with unit steps
$u = 1{:}2{:}7$	takes steps of 2 to give $u = [1\ 3\ 5\ 7]$

How to input a matrix (a row at a time)

$A = [1\ 2\ 3; 4\ 5\ 6]$	has two rows (always a semicolon between rows)
$A = [1\ 2\ 3$ $\quad 4\ 5\ 6]$	also produces the matrix A but is harder to type
$B = [1\ 2\ 3; 4\ 5\ 6]'$	is the *transpose* of A. Thus A^T is A' in MATLAB

How to create special matrices

diag(v) produces the diagonal matrix with vector v on its diagonal

toeplitz(v) gives the symmetric *constant-diagonal* matrix with v as first row and first column

toeplitz(w, v) gives the constant-diagonal matrix with w as first column and v as first row

ones(n) gives an $n \times n$ matrix of ones

zeros(n) gives an $n \times n$ matrix of zeros

eye(n) gives the $n \times n$ identity matrix

rand(n) gives an $n \times n$ matrix with random entries between 0 and 1 (uniform distribution)

randn(n) gives an $n \times n$ matrix with normally distributed entries (mean 0 and variance 1)

ones(m, n) **zeros**(m, n) **rand**(m, n) give $m \times n$ matrices

ones(size(A)**)** **zeros(size**(A)**)** **eye(size**(A)**)** give matrices of the same shape as A

How to change entries in a given matrix A

$A(3,2) = 7$ resets the $(3,2)$ entry to equal 7
$A(3,:) = v$ resets the third row to equal v
$A(:,2) = w$ resets the second column to equal w
The colon symbol : stands for *all* (all columns or all rows)
$A([2\ \ 3],:) = A([3\ \ 2],:)$ exchanges rows 2 and 3 of A

How to create submatrices of an $m \times n$ matrix A

$A(i,j)$	returns the (i,j) entry of the matrix A (scalar = 1×1 matrix)
$A(i,:)$	returns the ith row of A (as row vector)
$A(:,j)$	returns the jth column of A (as column vector)
$A(2:4, 3:7)$	returns rows from 2 to 4 and columns from 3 to 7 (as 3×5 matrix)
$A([2\ 4],:)$	returns rows 2 and 4 and all columns (as $2 \times n$ matrix)
$A(:)$	returns one long column formed from the columns of A ($mn \times 1$ matrix)

triu(A) sets all entries below the main diagonal to zero (upper triangular)
tril(A) sets all entries above the main diagonal to zero (lower triangular)

Matrix multiplication and inversion

$A * B$	gives the matrix product AB (if A can multiply B)
$A. * B$	gives the entry-by-entry product (if $\text{size}(A) = \text{size}(B)$)
$\text{inv}(A)$	gives A^{-1} if A is square and invertible
$\text{pinv}(A)$	gives the pseudoinverse of A
$A\backslash B$	gives $\text{inv}(A) * B$ if $\text{inv}(A)$ exists: *backslash* is left division
$x = A\backslash b$	gives the solution to $Ax = b$ if $\text{inv}(A)$ exists

See *help slash* when A is a rectangular matrix!

Numbers and matrices associated with A

det(A) is the *determinant* (if A is a square matrix)

rank(A) is the *rank* (number of pivots = dimension of row space and of column space)

size(A) is the pair of numbers $[m \;\; n]$

trace(A) is the *trace* = sum of diagonal entries = sum of eigenvalues

null(A) is a matrix whose $n - r$ columns are an orthogonal basis for the nullspace of A

orth(A) is a matrix whose r columns are an orthogonal basis for the column space of A

Examples

$E = \text{eye}(4); E(2,1) = -3$ creates a 4×4 elementary elimination matrix
$E*A$ subtracts 3 times row 1 of A from row 2.
$B = [A \;\; b]$ creates the augmented matrix with b as extra column
$E = \text{eye}(3); P = E([2 \;\; 1 \;\; 3], :)$ creates a permutation matrix
Note that $\text{triu}(A) + \text{tril}(A) - \text{diag}(\text{diag}(A))$ equals A

Built-in M-files for matrix factorizations (all important!)

$[L, U, P] = \mathbf{lu}(A)$ gives three matrices with $PA = LU$

$e = \mathbf{eig}(A)$ is a vector containing the eigenvalues of A

$[S, E] = \mathbf{eig}(A)$ gives a diagonal eigenvalue matrix E and eigenvector matrix S with $AS = SE$. If A is not diagonalizable (too few eigenvectors) then S is not invertible.

$[Q, R] = \mathbf{qr}(A)$ gives an $m \times m$ orthogonal matrix Q and $m \times n$ triangular R with $A = QR$

Creating M-files

M-files are text files ending with .m which MATLAB uses for functions and scripts. A script is a sequence of commands which may be executed often, and can be placed in an m-file so the commands do not have to be retyped. MATLAB's demos are examples of these scripts. An example is the demo called *house*. Most of MATLAB's functions are actually m-files, and can be viewed by writing *type xxx* where *xxx* is the name of the function.

To write your own scripts or functions, you have to create a new text file with any name you like, provided it ends with .m, so MATLAB will recognize it. Text files can be created, edited and saved with any text editor, like *emacs*, *EZ*, or *vi*. A script file is simply a list of MATLAB commands. When the file name is typed at the MATLAB prompt, the contents of the file will be executed. For an m-file to be a function it must start with the word *function* followed by the output variables in brackets, the function name, and the input variables.

Examples

```
function [C]=mult(A)
r=rank(A);
C = A' * A;
```

Save the above commands into a text file named mult.m Then this funtion will take a matrix A and return only the matrix product C. The variable r is not returned because it was not included as an output variable. The commands are followed by *;* so that they will not be printed to the MATLAB window every time they are executed. It is useful when dealing with large matrices. Here is another example:

```
function [V,D,r]=properties(A)
% This function finds the rank, eigenvalues and eigenvectors of A
[m,n]=size(A);
if m==n
[V,D]=eig(A);
r=rank(A);
else
disp('Error: The matrix must be square');
end
```

Here the function takes the matrix A as input and only returns two matrices and the rank as output. The % is used as a comment. The function checks to see if the input matrix is square and then finds the rank, eigenvalues and eigenvectors of a matrix A. Typing *properties(A)* only returns the first output, V, the matrix of eigenvectors. You must type *[V,D,r]=properties(A)* to get all three outputs.

Keeping a diary of your work

The command **diary('file')** tells MATLAB to record everything done in the MATLAB window, and save the results in the text file named 'file'. Typing **diary on** or **diary off** toggles the recording. Old diary files can be viewed using a text editor, or printed using *lpr* in unix. In MATLAB, they can be viewed using the **type** *file* command.

Saving your variables and matrices

The command **diary** saves the commands you typed as well as MATLAB's output, but it does not save the content of your variables and matrices. These variables can be listed by the command **whos** which also lists the sizes of the matrices. The command **save 'xxx'** will save the matrices and all variables listed by the **whos** command into the file named *xxx*. MATLAB labels these files with a .mat extension instead of .m which are scripts or functions. *xxx.mat* files can be read by MATLAB at a later time by typing **load** *xxx*.

Graphics

The simplest command is $\textbf{plot}(x, y)$ which uses two vectors x and y of the same length. The points (x_i, y_i) will be plotted and connected by solid lines.

If no vector x is given, MATLAB assumes that $x(i) = i$. Then $\textbf{plot}(y)$ has equal spacing on the x-axis: the points are $(i, y(i))$.

The type and color of the line between points can be changed by a third argument. The default with no argument is a solid black line "–". Use *help plot* for many options, we indicate only a few:

MATLAB 5: $\textbf{plot}(x, y,' r+ :')$ plots in r = red with + for points and dotted line

MATLAB 4: $\textbf{plot}(x, y,' --')$ is a dashed line and $\textbf{plot}(x, y,' \cdot')$ is a dotted line

You can omit the lines and plot only the discrete points in different ways:

$\textbf{plot}(x, y,' o')$ gives circles. Other options are $'+'$ or $'\times'$ or $'*'$

For two graphs on the same axes use $\textbf{plot}(x, y, X, Y)$. Replace **plot** by **loglog** or **semilogy** or **semilogx** to change one or both axes to logarithmic scale. The command

axis ([$a\ \ b\ \ c\ \ d$]) will scale the graph to lie in the rectangle $a \leq x \leq b$, $c \leq y \leq d$. To title the graph or label the x-axis or the y-axis, put the desired label in quotes as in these examples:

title ('height of satellite')   *xlabel* ('time in seconds')   *ylabel* ('height in meters')

The command **hold** keeps the current graph as you plot a new graph. Repeating **hold** will clear the screen. To print, or save the graphics window in a file, see *help print* or use

print -Pprintername   print -d   filename