Dr Zacharias Anastassi
Associate Professor in Computational Mathematics
Faculty of Computing, Engineering and Media
De Montfort University, UK

Introduction to Matlab

Dr Zacharias Anastassi

Generally

The following commands can be entered in the Command Window of Matlab followed by Enter

a = 2	Assign the value 2 to variable a.
b = 3;	Assign the value 3 to variable b, without showing the result.
c = a + b	Returns the sum a and b.
x = 1, y = -2	Two commands in one line, separated by comma (both results show).
x = -x	Stores the negative value of x to x (x is now equal to -1). This is an assignment, not an equation!
b	Returns the assigned value, that was previously entered (3).
clear b	Clears variable b from the memory.
b	Returns an error message, since we have cleared variable b.
clear	Clears all variables from the memory.

Entering Matrices

v = [7 -2 5] Creates a row-vector.

v = [7,-2, 5] Same as above.

w = [1; 2; 3] Creates a column-vector.

A = [4 -2 5; -1 7 -6; 2 3 5] Creates a 3x3 matrix.

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

Creates the same matrix as above (we press Enter at the end of each row).

Basic Functions

va = abs(v) Creates the vector va, of which each element is equal to the absolute value of the corresponding element of the vector v.
v is called the argument of the function

sq = sqrt(v) Creates the vector sq, of which each element is equal to the absolute value of the corresponding element of the vector v. If some element of v is negative, then the corresponding element of sq will be complex.

y = sin(pi/6) Returns the sin of $\pi/6$ in rads. Similarly for cos, tan, asin, acos, atan.

y = exp(4) The exponential function. Returns e^4.

v' Returns the transpose matrix of v.

C1 = eye(4) Returns the identity matrix I with dimension 4.

C2 = eye(5,3) Returns the identity matrix, which is extended to 5 rows filled with zeros.

C3 = ones(3) Returns the square matrix of dimension 3 and all elements equal to one.

C4 = ones(2,3) Returns the matrix with 2 rows and 3 columns with all elements one.

C5 = zeros(2) Returns the square matrix of dimension 2 and all elements equal to zero.

A1 = diag(v) If v is a vector, then it returns a square matrix of equal dimension, with the diagonal elements equal to the elements of v, and all other elements zero.
If v is a matrix, then it returns a column-vector with the elements of the main diagonal of v.

A2 = diag(v,k) (v is a vector). If k = 0, same as diag(v). If k = 1, then the diagonal is the one above the main diagonal. If k = -2, then the diagonal is the two below the main diagonal and so on. The dimensions of the matrix are the minimum necessary to include v.

w = diag(A,k) (A is a matrix). Returns the elements of the k-diagonal of matrix A and creates a column-vector.

NOTE: All functions (unless otherwise mentioned), when have a matrix as an argument, they return a matrix with the same dimensions, of which each element is equal to the value of the function at the point equal to the corresponding element of the initial matrix. This holds e.g. for sort, abs, sin etc.

Sequences

u1 = [4:10] Returns the vector with elements 4,5,6,7,8,9,10 (step 1).

v1 = [3:2:8] Returns the vector with elements 3,5,7 (step 2). Notice that the last element is not 8.

t1 = [1:-1/2:-1] Returns the vector with elements 1,1/2,0,-1/2,-1 (step -½).

u2 = linspace(4,10,7) Returns the vector with 7 numbers starting from 4 until 10. 10 is always included as the last element.

t1 = logspace(0,3,4) Returns the vector with 4 numbers 10^0, 10^1, 10^2, 10^3.

Dimensions and Indices

n = length(t1) Returns the length of t1, if it is a vector or the maximum dimension, if it is a matrix.

[nr,nc] = size(A) Assigns the rows of the matrix A in nr and the columns of A in nc.

A(1,2) Returns the element of the first row and second column.

A(4,5) = 3 Assigns the value 3 to the element at the 4th row and 5th column (The elements that were missing in the original matrix are filled with zeros, e.g. all elements of the 4th row, and 4th and 5th columns are 0 except for the element (4,5), which is 3).

B = [1:5, 2:6] Creates the matrix $\begin{bmatrix}1 & 2 & 3 & 4 & 5 & 2 & 3 & 4 & 5 & 6\end{bmatrix}$.

B = [1:5; 2:6] Creates the matrix $\begin{bmatrix}1 & 2 & 3 & 4 & 5\\ 2 & 3 & 4 & 5 & 6\end{bmatrix}$.

B(2,:) Returns the 2nd row of the matrix B in the form of row vector.

B(:,5) Returns the 5th column of the matrix B in the form of column vector.

B(1:2, end-1:end) Returns a matrix with 4 elements of the 2 first rows and the 2 last columns, thus the matrix $\begin{matrix}4 & 5\\ 5 & 6\end{matrix}$.

B(:, 2:4) = [] Deletes columns from 2 to 4 included.

Matrix Production with Random Elements

v = rand(1,3) Produces a matrix with 1 row and 3 columns and random real value elements from 0 to 1.

A = rand(3) Produces a square matrix with dimension 3 and random elements.

B = rand(3) Produces a matrix 3x3 with random elements (generally different to those of matrix A).

Matrix Operations

S = A+B Returns the sum of A and B matrices, which have equal dimensions.

P = A*B Calculates the product of A(m x n) times B(n x t) and assigns it to P(m x t).

X1.*X2	For the matrices X1 and X2, which have equal dimensions, creates the matrix with elements equal to the product of the X1, X2 corresponding elements.
X1./X2	For the matrices X1 and X2, which have equal dimensions, creates the matrix with elements equal to the quotient of the X1, X2 corresponding elements (X1 over X2).
A^3	Performs the multiplication A*A*A (only for square matrix).
X3 = X1.^3	Creates the X3 matrix with equal dimensions to X1 matrix and each single element equals to the corresponding elements of X1 power to 3.
X4 = X1.^X2	For the X1 and X2 matrices, which have equal dimensions, creates X4 matrix, of which every element is equal to the corresponding X1 element raised to the corresponding X2 element.
A^(-1)	Returns the inverse matrix of A.
inv(A)	Same as above.
A\B	Inverse division: Does the operation inv(A)*B, where B matrix is not necessarily a square matrix, but the number of the rows of B must be equal to the dimension of square matrix A. The inverse division is useful for the solution of the system $Ax = b$.
max(A)	If A is a matrix, it returns the row vector with elements equal to the maximum of the value of element of the column where belongs to. If A is a vector (column or row), it returns the maximum value.
max(A,[],1)	If A is matrix, it returns the row vector with elements equal to the maximum of the value of element of the column where belongs to.
max(A,[],2)	If A is matrix, it returns the column vector with elements equal to the maximum of the value of element of the row where belongs to.
min(A)	Same syntax with max, but returns the minima.
sum(A)	If A is matrix, it returns the row vector with elements equal to the sum of the elements of the column where belong to. If A is a vector (column or row), it returns the sum of all elements.
sum(A,1)	If A is matrix, it returns the row vector with elements equal to the sum of the elements of the column where they belong to.
sum(A,2)	If A is matrix, it returns the column vector with elements equal to the sum of the elements of the row where they belong to.
prod(A)	Same syntax with sum, but it returns the products.

Matrix Reductions

```
C = reshape([1   2   3;
             4   5   6;
```

7 8 9;
10 11 12], 2,6)

Reshapes the 4x3 matrix to the 2x6 matrix $C = \begin{bmatrix} 1 & 7 & 2 & 8 & 3 & 9 \\ 4 & 10 & 5 & 11 & 6 & 12 \end{bmatrix}$, this means that this command reads the elements column by column.

E = C(:) Reshape C matrix to a column vector.

Complex Numbers

z1 = 1-2*i Creates a complex number with the real part equal to 1 and imaginary part equal to -2.

z2 = 1-2j Equal to z1 (it's recommended that i and j are not used as names of variables).

real(z1) Returns the real part of z1.

imag(z1) Returns the imaginary part of z1.

angle(z1) Returns the phase angle of z1.

String Functions

s = 'A string example' Anything that is written inside single quotes is a string.

s = num2str(pi,3) Assigns the value '3.14' to s. Converts a number to string with 3 significant digits.

n = str2num('2.34') Assigns the value 2.340000000000000 to n.

Polynomials

polyval([1 0 -2 12], 1.5) Calculates the value of the polynomial $x^3 - 2x + 12$ at x = 1.5.

roots([1 0 -2 12]) Calculates all the real and complex roots of the polynomial $x^3 - 2x + 12$.

poly([0 -1 2]) Returns (the coefficients of the) polynomial that has roots the elements of the vector.

2D Graphs

t = [0: pi/20: 2*pi]; plot(t, cos(t)) Creates the vector $t \in [0, 2\pi]$ with step π/20 and draws the cos graph.

grid on Shows the grid on the last graph.

title('The Cos Function')	Shows the title on the graph.
xlabel('t'); ylabel('Cos')	Shows the names of x and y axes respectively.
axis([0 2*pi -1 1])	Set the limits of the graph for $t \in [0,2\pi]$ and for $y \in [-1,1]$.
plot(t,cos(t),'k-', t,sin(t),'ro')	Creates the cos function graph with continuous black line and the sin function graph with red cycles on the same graph.
legend('Cos', 'Sin')	Shows up the appendix of the graph.

Logarithmic Scale

semilogy(t, abs(cos(t)))	Draws the graph of the absolute value of cos, but the scale is logarithmic on the vertical axis.
semilogx(t, abs(cos(t)))	Same as above, but for the horizontal axis is logarithmic.
loglog(t, abs(cos(t)))	Logarithmic scale on both axes.

Multiple Graphs

subplot(2,2,1)	Creates 4 placeholders for graphs (2 rows x 2 columns) and sets the top left as active. The next plot command will plot on that placeholder.

3D Graphs

x = linspace(-20, 20, 80);

y = linspace(-2*pi, 2*pi, 160);

[X,Y] = meshgrid(x,y)	Creates X and Y vectors with dimensions length(y), length(x) and the elements of a column of X are equal to each other, as the elements of a row of Y are equals to each other. Thus, X matrix is the row vector x that is repeated vertically 160 times and Y matrix is the column vector y' that is repeated horizontally 80 times.
surf(X, Y, X.^2.*cos(Y))	Prints the 3D graph of the function $x^2 \cos(y)$, for $x \in [-20,20]$ and $y \in [-2\pi, 2\pi]$.

REMARK We use the full stop . before an operation symbol (.* ./ .^ .\) when we perform an element-wise operation.

More Commands

clc	Removes the previous commands and their outputs from the command window, but no the variables from the memory.
help exp	Gives information about the syntax of the exp function.
lookfor exponential	Returns a list with contents of the word 'exponential', which we can take a separate look for a single one with the command "help".

Keys Combinations

Ctrl+C ή Ctrl+Break	Breaks every single command is in progress (works only when the command window is active).

Script and Function Creation

File > New > M-file Creates a new m-file

File > Open or Ctrl+O Opens a file

- Press F5 from the m-file editor or type the name of a file without the file suffix on the command window to run the m-file.
- An m-file can contain a set of commands (like the ones entered in the command window) and executed consecutively. This file is considered a script, as opposed to a function, that is does not need input nor output arguments.

REMARK A valid filename starts with a letter, followed by letters or numbers or underscore.

- To set an m-file as a function with name namefun, input arguments in1,in2,in3 etc. and output arguments out1,out2, we write in the first line of the file

```
function [out1,out2] = namefun(in1,in2,in3)
```

or

```
function out = namefun(in1,in2,in3)
```

if the function returns only one output argument "out".

- Everything is after the sign % is a comment. The block of lines, which are comments and are at the beginning of the m-file or right after the function definition, appear with the "help" command.

Function Example

```
%This function finds the sum and the product of two numbers
function [sum1,prod1] = operations(a1,a2)
sum1 = sum([a1,a2]);
prod1 = prod([a1,a2]);
```

We save all the above lines in a file with the name `operations.m` and then we write on the command window `[s,p] = operations(2,3)` which assigns the value of sum1 (which is 5) to variable s and the value of prod1 (which is 6) to the variable p.

Other Basic Commands

`x = input('Enter a value for x')`
Notification for an external variable input from the user (not recommended).

`eps` Parameter that gives the maximum absolute error for every operation in Matlab for the specific computer (e.g. `2.22044604925031e-016`)

`format long` Set the default output number of decimal digits 15.

`format short` Set the default output number of decimal digits 4.

Reading and Writing a file

`save('xyfile','x','y')` Creates the file `xyfile.mat` and stores the variables x and y.

`load xyfile` Loads the variables x and y, which were saved in the `xyfile.mat`., to the workplace (working memory).

`save xyfile x y -ascii` Creates the file `xyfile.txt` and stores x and y in text form.

`D = load('xyfile.txt')` Creates from x and y, that were stored in `xyfile.txt` the matrix D. x and y must agree in dimensions (same number of columns). Not recommended

`fnum = fopen('results.dat','wt')`
Opens the `results.dat` file for writing (creates a new file whether it exists or not, deleting previous data) and writes a value (positive for successful creation, -1 for failed) to `fnum`, which is the file identification number.

- The license can be `'wt'`, as explained above, `'rt'`, where it opens an existing file for reading or 'at' where it creates the file, if it does not exist or adds new data to the end of the file if there is already.

`fprintf(fnum, 'Example %s:\n %7.4f %11.3e\n \t...\n', 'One', -100*pi, pi)`

Prints (writes) into the file with identification number `fnum` (or if `fnum` is omitted, prints to the command window) the variables `'One'` (string) `-100*pi` and `pi` in the format given by the second argument. `%s` displays the first variable in sequence as a string. `%7.4f` means that the (second) number will be displayed with 4 decimal places and with the digits of the integer, the dot and the sign will occupy <u>at least</u> 7 positions. Similarly for `%11.3e` except that it displays the number in format $a \cdot 10^b$. `\t` adds the tab character and `\n` changes the line. Each other character appears normally. The two numbers are right to their allocated space, because the first number after % is positive. If it was negative, they would be justified to the left. The result in the example is:

```
Example One:
 -314.1593  3.142e+000
    ...
```

`a = fgetl(fnum)` It reads a line from the file with identification number `fnum` and assigns it to variable `a` as a string. For the previous one gives:

```
a = Example One:
```

`b = fscanf(fnum,'%f %f',3)` Reads 2 numbers from the file with `fnum` number and stores them in the vector - column b. The third argument 3 means that it will read up to 3 numbers (here it does not affect). If the number is greater than the number of variables to be read through the 2nd argument, then repeat e.g. equivalent we could give

`b = fscanf(fnum,'%f',2)`. The appearance of the result is affected by the default format:

```
b = -314.1593
     3.1420
```

For both `fgetl` and `fscanf`, reading starts from the point where the reading was last stopped and ends when the line for `fgetl` ή or when it has read the variables for `fscanf`.

`fclose(fnum)` Ends the program communication with the file that has a `fnum` and saves the results.

`type results.dat` Displays the contents of the `results.dat` file in the active directory in the command window.

Variable Value Comparison

`x=3<4, y=4>=5` Returns `x=1` and `y=0`. 1 represents the trues and 0 is the false. 0 and 1 is logical type and not double. Also we use `==` (equality), `~` (opposite), `&` (and), `|` (or). e.g.

`a = 3==3 & (4~=4 | ~1)` gives `a=0`.

Commands under Conditions

```
if x>0
    disp('Positive')
elseif x==0
    disp('Zero')
elseif x<0
    disp('Negative')
else
    disp('Error')
end
```

Displays the corresponding message, positive, zero, negative and incorrect (complex or non-computable, eg NaN - Not a Number) respectively.

`disp` displays a message on the command window.

```
switch sign(x)
case 1
    disp('Positive')
case 0
    disp('Zero')
case -1
    disp('Negative')
otherwise
    disp('Error')
end
```

Equivalent code with the previous one. Less flexible, since the values to be considered must be integers, logically 0 and 1 or strings.

`sign` returns 1,0,-1 for positive, zero and negative field respectively.

Loops Interations

```
for k=1:10
    fprintf('%3.0f',k)
end
```

Outputs `k = 1  2  3  4  5  6  7  8  9 10`

```
k=1;
while k<=10
    fprintf('%3.0f',k)
    k=k+1;
end
```

Similar output as the latter

Various Commands

`break`	Immediate termination of loop iteration
`return`	Immediate termination of the function
`y = [3 6 9 12 15];`	
`x = [3 4 1  5  2];`	
`z = y(x)`	Returns `z = 9 12 3 15 6`. x works as index.
`zb = z>10`	Returns `zb = 0 1 0 1 0`. zb is a logical array.
`z(zb)`	Returns `12 15`. zb works as an index (logical). If we create a vector `zb = [0 1 0 1 0]` and give `z(zb)`, the result will be wrong, because 0 and 1 are not logical but integers.
`zi = find(z>10)`	Returns `2 4`, i.e. the indicators (positions) of the elements that satisfy the condition.
`z(zi)`	Returns `12 15`. It gives both `z(zb)`. Here is no error, because integers 2 and 4 are in the 1to length(z) range here 5.
`z(z>10)`	Same as before.
`all(zb)`	Returns 0 (false). Returns 1, when all elements are different from 0.
`any(zb)`	Returns 1 (true). Returns 1, when an element is different from 0.
`nargin, nargout`	Returns the number of input and output variables in a function.
`global V`	Makes the V parameter common for all functions that have this command (including the workspace).
`feval('cos',0)`	Returns 1, ie. `cos(0)` value 1
`trig_fun = inline('cos(x)*sin(x)')`	Specifies the function `trig_fun` shown by the workspace and each function, regardless of where we have defined it.