MATLAB Quick Reference Guide – Basic Commands

Help

`help command`	quick help for a given *command*
`doc command`	extensive help for a given *command*
`lookfor keyword`	search for *keyword* in help files
`ver`	displays Matlab version and toolboxes

note: sometimes it can be faster to just google "matlab" followed by whatever topic you want.

General Commands

`whos`	list all variables currently defined
`clear`	delete all variables
`clear a`	delete variable *a*
`close all`	close all figure windows
`format short`	shows fewer sig. figs. (default)
`format long`	shows lots of sig. figs.
`format short e`	scientific notation with a few sig. figs.
`format long e`	scientific notation with lots of sig. figs.

Special Numbers

`pi`	variable pi = 3.14159…
`i or j`	$\sqrt{-1}$
`ans`	most recent answer from calculation
`eps`	smallest floating-point value on your computer

Defining Variables

`x = 3`	define variable x to be 3
`m = 9.11e-31`	scientific notation ($m = 9.11 \times 10^{-31}$)
`x = 3+2i`	complex number
`phi = pi/2`	use descriptive names for variable
`y = x`	set `y` equal to current value of `x`

Basic Functions

`x+y`	addition	$x + y$
`x-y`	subtraction	$x - y$
`x*y`	multiplication	$x \times y$
`x/y`	division	x / y
`x^y`	raise to the power:	x^y
`sqrt(x)`	square root:	$\sqrt{x}$
`exp(x)`	exponential:	e^x
`log(x)`	natural log:	$\ln(x)$
`log10(x)`	log base-10:	$\log_{10}(x)$
`abs(x)`	absolute value:	$\lvert x \rvert$
`(x-y)/(x+y)`	group variables using parentheses	

Rounding and Integer Functions

`round(x)`	rounds to nearest integer
`floor(x)`	nearest integer less than `x`
`ceil(x)`	nearest integer greater than `x`
`fix(x)`	nearest integer to `x` looking toward 0
`sign(x)`	returns the sign of a number (±1 or 0)
`mod(x,y)`	modulus after division of `x` / `y`

Trig Functions

Radians:

`sin(phi)`	`cos(phi)`	`tan(phi)`	
`asin(x)`	`acos(x)`	`atan(x)`	`atan2(y,x)`

Degrees:

`sind(phi)`	`cosd(phi)`	`tand(phi)`	
`asind(x)`	`acosd(x)`	`atand(x)`	`atan2d(y,x)`

Hyperbolic Functions:

`sinh(x)`	`cosh(x)`	`tanh(x)`
`asinh(x)`	`acosh(x)`	`atanh(x)`

Other Functions

`besselj(x)`	Bessel function of 1st kind
`besseli(x)`	Modified Bessel function
`bessely(x)`	Bessel function of 2nd kind
`besselh(x)`	Bessel function of 3nd kind
`erf(x)`	Error function
`erfinv(x)`	Inverse error function
`airy(x)`	Airy function
`gamma(x)`	gamma function
`factoral(x)`	factorial
`expint(x)`	exponential integral function
`rand()`	random number drawn from a uniform distribution on [0,1)
`randn()`	random number drawn from a Gaussian distribution ($\bar{x} = 0, \sigma = 1$)

Creating Vectors and Matrices

`x = [1 2 3]`	set x to be a 1x3 row vector
`x = [1, 2, 3]`	same as above (1x3 row vector)
`x = [1; 2; 3]`	set x to be a 3x1 column vector
`x = [1 2; 3 4]`	set x to be a 2x2 matrix
`x = []`	removes contents of variable x

Create a Vector Using Colon Notation

`1:5`	creates row matrix [1 2 3 4 5]
`0:2:10`	count by twos [0 2 4 6 8 10]
`0:3:10`	count by threes [0 3 6 9]
`5:-1:0`	count backwards [5 4 3 2 1 0]
`0:pi/10:pi/2`	increments of pi/10 from 0 to pi/2

More Ways to Create Matrices

`linspace(a,b,n)`	*n* equally spaced numbers from *a* to *b*
`logspace(a,b,n)`	*n* logarithmically spaced numbers from 10^a to 10^b
`zeros(2)`	fill a 2x2 matrix with zeros
`zeros(1,3)`	fill a 1x3 row vector with zeros
`ones(2,3)`	fill a 2x3 matrix with ones
`eye(3)`	3x3 identity matrix I
`rand(1,10)`	fill a 1x10 row vector with uniform random numbers on [0,1)

Referencing Cells in a Row or Column Vector

`x(3)`	3rd element of vector `x`
`x(3:8)`	3rd to 8th elements of `x`
`x(end)`	last element of `x`
`x(3:end)`	3rd to last element of `x`
`x(1:2:end)`	odd elements of x, i.e. 1, 3, 5,...
`x(end:-1:1)`	returns elements in reverse order

Referencing Cells in an n x m Matrix

`A(2,3)`	cell in 2nd row, 3rd column of matrix `A`
`A(5,:)`	5th row of x → row vector
`A(:,4)`	4th column of x → column vec.
`A(1:2,3:4)`	extracts a 2x2 matrix
`A(:,[1 3])`	1st and 3rd columns of matrix `A`

Operations with Vectors and Matrices

`x * 3`	multiply every element of `x` by 3
`x + 3`	add 3 to every element of `x`
`x .* y`	element-wise product of vectors `x` and `y` (`x` and `y` must be same length)
`x * y`	inner (dot) product of row vector `x` and column vector `y`
`A * y`	matrix product of matrix `A` and vec `y`
`x.^2`	square very element of `x`
`sqrt(x)`	square root of every element of x
`sin(x) ./ x`	element-wise calculation of $\sin(x)/x$

Matrix Properties and Stats

`size(A)`	# rows and # columns of matrix `A`
`length(x)`	# elements in a vector or maximum dimension of a matrix
`sum(x)`	sum of elements in vector x
`min(x)`	minimum value of cells in vector x
`max(x)`	maximum value of cells in vector x
`[xmin,imin]=min(x)`	min value and index of vector x
`[xmax,imax]=max(x)`	max value and index of vector x
`mean(x)`	mean value of cells in vector x
`std(x)`	standard deviation of vector x
`mode(x)`	mode (most common value) of vector x

Transpose and Adjoint Operators

`x.'`	transpose of vector `x` (converts btween column vector ↔ row vector)
`x'`	adjoint of vector x (complex conjugate of transpose)

Dot Product and Outer Product

Let `x` and `y` be $1 \times n$ row vectors

`x * y'`	inner (dot) product of two row vectors `x` and `y`
`x' * y`	outer product of a column vector `x'` and a row vector `y`
`dot(x,y)`	another way of writing the dot product works even if `x` and `y` are both row or column vectors

Common Errors

`"inner matrix dimensions must agree"`

You multiplied two matrices improperly. If you wanted to multiply them element-by-element, make sure you used `".*"` rather than `"*"`. Make sure the matrices have the same length.

`"index exceeds matrix dimensions"`

You tried to access an element of an array that was larger than the size of the array, i.e. `x(3)` gives the error if `x = [1 2]`

`"undefined function or variable"`

If you get this message, check your spelling and case. Case matters. Most Matlab commands are lower case. The command `sin(pi)` is OK, while `Sin(pi)` will give an error. Also, make sure you are consistent with the case of your variable names: `phi` and `Phi` are NOT the same variables.

`ctrl-C`

Stops runaway code

Sometimes a calculation produces one of the following:

- `inf` = infinity (produced if you type `1/0`)
- `NaN` = Not a Number (produced if you type `0/0`) in this case the result is undefined

Parentheses `( )` are used as follows:

- to reference elements of a matrix, e.g. `A(2,3)`
- to pass values to functions, e.g. `sin(phi)`
- to group arithmetic elements, e.g. `ave = (x+y)/2`

Square Brackets `[ ]` are used as follows:

- to create vectors or matrices, e.g. `x = [1 3 5]`
- to concatenate vectors or matrices, e.g. `z = [x y]` where `x` and `y` have the same number of rows or `z = [x; y]` where `x` and `y` have the same number of columns

Curly Brackets `{}` are used to create cell arrays.

- example: `x = {1, 2, 3, 'frog'}`

Order of Operations Examples

`2*x + 3`	$2x + 3$
`3/x^2 + 5`	$\frac{3}{x^2} + 5$
`(3/x)^2 + 5`	$\left(\frac{3}{x}\right)^2 + 5$
`2e-3`	2×10^{-3}
`1/2e-3`	$\frac{1}{2 \times 10^{-3}}$
`1/x*y`	$\frac{y}{x}$ (bad style. Better to write: `y/x`)
`1/(x*y)`	$\frac{1}{xy}$

Tip: Only use parentheses when needed or to make calculations more legible. Don't overuse them.