MATLAB Quick Reference

Getting Started

`edit` *prog*	Starts MATLAB Editor/Debugger; may add the optional argument *prog*, to open a specific program (omit program's `.m` extension).
`simulink`	Starts SIMULINK with new model.
`ee2311`	Opens the SIMULINK EE 2311 Blockset.

Help Commands

`helpwin`	Starts MATLAB Help Window.
`helpdesk`	Starts MATLAB Help Desk (HTML).
`help` *cmnd*	Runs help file description for the Matlab function *cmnd*.
`demo`	Starts MATLAB Demo Window.

Plotting Commands

`plot(`*x_1*`,`*y_1*`,`*x_2*`,`*y_2*`,...,`*options*`)`	Graphs the values in vector *y_n* versus the values in vector *x_n*, onto two linear axes (as many pairs of vectors can be plotted as is desired).
`semilogx(...)`	Same as `plot`, but graphs values onto a logarithmic x-axis and linear y-axis.
`semilogy(...)`	Same as `plot`, but graphs values onto a linear x-axis and logarithmic y-axis.
`loglog(...)`	Same as `plot`, but graphs values onto two logarithmic axes.
`subplot(`*rows*`,`*cols*`,`*plot_num*`)`	Creates a *rows* by *cols* arrangement of graphs on a single figure; would be used directly before plotting the *plot_num*-th graph.
`axis([`*x_min x_max y_min y_max*`])`	Re-sizes the axes of the current figure to the given x and y ranges.
`xlabel(`*str*`)`	Draws a label for the x-axis of the current figure, using the string *str*.
`ylabel(`*str*`)`	Draws a label for the y-axis of the current figure, using the string *str*.
`title(`*str*`)`	Draws a title for the current figure, using the string *str*.
`grid`	Toggles the grid on or off for the current figure.
`hold`	Toggles the hold for the current figure (when hold is on, new plotting commands are added to the current figure; when off, new plotting commands erase the current figure and write over it).

Basic Housekeeping Commands

`clc`	Clears MATLAB Command Window; returns cursor to upper right.
`clf`	Clears current figure (current figure can be changed with the command `figure`).
`figure(`*fig*`)`	Creates or selects figure specified by *fig*; figure becomes the current figure for commands such as `clf` and `plot`.
`format` *type*	Changes number output format to *type* (such as `short` or `long`).
`workspace`	Starts MATLAB Workspace Browser.
`clear(`*var*`)`	Removes variable *var* from memory; returns pre-set variables (such as `pi`) to their original values; use without argument clears all workspace variables.
`who`	Lists all variable names currently in memory.
`editpath`	Starts MATLAB Path Browser.
`cd` *x*`:\`*dir*	Changes current working directory to directory *dir* on drive *x*.
`addpath` *pathstr*	Adds to the path the directory specified by *pathstr* (be sure to enclose the path in single quote marks, to make it a string).

General Commands and Operators

`%`	Tells MATLAB to ignore the rest of the line; used for commenting.
`...`	Tells MATLAB that the current command is continued on the next line; can be used anywhere in a command, except in the middle of a function or variable name.
`[ ]`	Used to create a vector or matrix; within the brackets, values within a row are separated by spaces or commas, and rows are separated by semicolons.
`;`	Suppresses the display of the output of a MATLAB operation; also starts a new row in a vector or matrix declaration (when using the square brackets).
`=`	Stores values to the workspace; saves the result of any MATLAB operation under the variable name preceding it.
`pi`	The constant $\pi \approx 3.1415926535897$.
a`:`*n*`:`*c*	Creates a vector of values beginning with *a* and counting by *n* up to *c*.
`linspace(`*a*`,`*b*`,`*pts*`)`	Creates a vector of values ranging from *a* to *b*, and containing *pts* number of values.
`logspace(`*a*`,`*b*`,`*pts*`)`	Creates a logarithmically scaled vector of values, ranging from 10^a to 10^b, and containing *pts* number of values.

Logic and Relational Operators

`==`	Equal to; used to compare scalars or identically-sized pairs of matrices or vectors; returns 1 if statement of equality is true, and 0 if false.
`~=`	Not equal to.
`<`	Less than.
`<=`	Less than or equal to.
`>`	Greater than.
`>=`	Greater than or equal to.
`&`	Logical AND; returns 1 if AND operation is true, and 0 if false.
`|`	Logical OR.
`~`	Logical NOT; returns the logical opposite of the elements of *x*.

Arithmetic Operators

`+`	Addition; used to add two scalars, two vectors, or a scalar to a vector.
`-`	Subtraction; follows rules of addition.
`*`	Matrix multiplication; can be used to multiply two scalars, or a scalar and a vector.
`.*`	Array or element-by-element multiplication; used to multiply the elements of a vector by the elements of another vector.
`^`	Matrix power; can be used to raise a scalar to a scalar exponent.
`.^`	Array or element-by-element power; used to raise the elements of a vector to a scalar exponent, or to a vector of exponents.
`/`	Matrix division; can be used to divide two scalars, or a vector by a scalar.
`./`	Array or element-by-element division; used to divide the elements of a vector by the elements of another vector, or to divide a scalar by a vector.

Common Loop Functions

`if` *x*	Used to execute successive statements based on the logical value of *x*.
`else`	Used to execute statements contrary to `if` condition.
`elseif` *x*	Same as `else`, but allows for an extra logical condition.
`for` *n*`=`*a*`:`*b*	Used to construct a definite loop; will repeat as *n* counts up from *a* to *b*.
`while(`*x*`)`	Used to construct an indefinite loop; will repeat as long as *x* remains true.
`end`	Delimits the scope of an `if`, `for`, or `while` construction.
`break`	Terminates execution of current loop.

Common Trigonometric Functions

`sin(`*x*`)`	Returns the sine of the elements of *x*.
`asin(`*x*`)`	Returns the arcsine (inverse sine) of the elements of *x*.
`cos(`*x*`)`	Returns the cosine of the elements of *x*.
`acos(`*x*`)`	Returns the arccosine (inverse cosine) of the elements of *x*.
`tan(`*x*`)`	Returns the tangent of the elements of *x*.
`atan(`*x*`)`	Returns the arctangent (inverse tangent) of the elements of *x*.

Common Exponential and Logarithmic Functions

`exp(`*x*`)`	Returns the exponential (e^x) of the elements of *x*.
`log(`*x*`)`	Returns the natural logarithm ($\ln x$) of the elements of *x*.
`log2(`*x*`)`	Returns the base-2 logarithm ($\log_2 x$) of the elements of *x*.
`log10(`*x*`)`	Returns the common logarithm ($\log_{10} x$) of the elements of *x*.
`sqrt(`*x*`)`	Returns the square root ($\sqrt{x}$) of the elements of *x*.

Common Complex Number Functions

`i`	Returns the basic imaginary unit ($\sqrt{-1}$); `j` may also be used.
`abs(`*x*`)`	Returns the absolute value or complex magnitude of the elements of *x*.
`angle(`*x*`)`	Returns the complex phase of the elements of *x*.
`real(`*x*`)`	Returns the real part of the of the elements of *x*.
`imag(`*x*`)`	Returns the imaginary part of the of the elements of *x*.
`conj(`*x*`)`	Returns the complex conjugate of the elements of *x*.

Common Rounding and Remainder Functions

`round(`*x*`)`	Rounds the elements of *x* towards the nearest integer.
`ceil(`*x*`)`	Rounds the elements of *x* towards ∞.
`floor(`*x*`)`	Rounds the elements of *x* towards $-\infty$.
`fix(`*x*`)`	Rounds the elements of *x* towards zero.
`rem(`*x*`,`*y*`)`	Returns the unsigned remainder of the division of the elements of *x* by the elements of *y*.
`signum(`*x*`)`	Returns 1, 0, and -1 for the corresponding positive, zero, and negative elements of *x*.