**# MATLAB Cheat Sheet**

Based off of [Learn X in Y Minutes](http://learnxinyminutes.com/docs/matlab/).

MATLAB stands for MATrix LABoratory. It is a powerful numerical computing language commonly used in engineering and mathematics.

**## Quick Access Links**

1. [Basics](#basics)

    1. [Comments and Code Sections](#comments)

    2. [Helper Commands](#commands)

    3. [Variables and Expressions](#vars)

2. [Matrices and Vectors](#matrices)

    1. [Declarations](#declarations)

    2. [Splicing and Dicing](#splice)

    3. [Arithmetic and Operations](#arith)

3. [Plots](#plots)

4. [Functions and Scripts](#functions)

5. [Programming Logic](#logic)

6. [Math/Engineering](#math)

    1. [Common Math Functions](#common)

    2. [Transfer Functions](#transfer)

7. [Vectorization](#vectorization)

8. [Optimization](#optimization)

9. [Machine Learning](#ML)

10. [Simulink](#simulink)

<a name="basics"></a>

## 1. Basics

<a name="comments"></a>

### i. Comments and Code Sections

```matlab

%% Code sections start with two percent signs. Section titles go on the same line.

% Comments start with a percent sign.

%{

Multi line comments look

something

like

this

%}

% Two percent signs denote the start of a new code section

% Individual code sections can be run by moving the cursor to the section followed by

% either clicking the "Run Section" button

% or     using Ctrl+Shift+Enter (Windows) or Cmd+Shift+Return (OS X)

%% This is the start of a code section

%  One way of using sections is to separate expensive but unchanging start-up code like loading data

load myFile.mat y

%% This is another code section

%  This section can be edited and run repeatedly on its own, and is helpful for exploratory programming and demos

A = A \* 2;

plot(A);

% commands can span multiple lines, using '...':

 a = 1 + 2 + ...

 + 4

```

<a name="commands"></a>

### ii. Helper Commands

```matlab

% commands can be passed to the operating system

!ping google.com

who % Displays all variables in memory

whos % Displays all variables in memory, with their types

clear % Erases all your variables from memory

clear('A') % Erases a particular variable

openvar('A') % Open variable in variable editor

clc % Erases the writing on your Command Window

diary % Toggle writing Command Window text to file

ctrl-c % Abort current computation

close all % Closes all figures

edit('myfunction.m') % Open function/script in editor

type('myfunction.m') % Print the source of function/script to Command Window

profile on  % turns on the code profiler

profile off     % turns off the code profiler

profile viewer  % Open profiler

help command    % Displays documentation for command in Command Window

doc command     % Displays documentation for command in Help Window

lookfor command % Searches for command in the first commented line of all functions

lookfor command -all % searches for command in all functions

% Output formatting

format short    % 4 decimals in a floating number

format long     % 15 decimals

format bank     % only two digits after decimal point - for financial calculations

fprintf('text') % print "text" to the screen

disp('text')    % print "text" to the screen

% pressing the up key shows you a history of previous commands

```

<a name="vars"></a>

### iii. Variables and Expressions

```matlab

% Variables & Expressions

myVariable = 4  % Notice Workspace pane shows newly created variable

myVariable = 4; % Semi colon suppresses output to the Command Window

4 + 6       % ans = 10

8 \* myVariable  % ans = 32

2 ^ 3       % ans = 8

a = 2; b = 3;

c = exp(a)\*sin(pi/2) % c = 7.3891

% Logicals

1 > 5 % ans = 0

10 >= 10 % ans = 1

3 ~= 4 % Not equal to -> ans = 1

3 == 3 % equal to -> ans = 1

3 > 1 && 4 > 1 % AND -> ans = 1

3 > 1 || 4 > 1 % OR -> ans = 1

~1 % NOT -> ans = 0

% Logicals can be applied to matrices:

A > 5

% for each element, if condition is true, that element is 1 in returned matrix

A( A > 5 )

% returns a vector containing the elements in A for which condition is true

% Strings

a = 'MyString'

length(a) % ans = 8

a(2) % ans = y

[a,a] % ans = MyStringMyString

% Cells

a = {'one', 'two', 'three'}

a(1) % ans = 'one' - returns a cell

char(a(1)) % ans = one - returns a string

% Structures

A.b = {'one','two'};

A.c = [1 2];

A.d.e = false;

% Variables can be saved to .mat files

save('myFileName.mat') % Save the variables in your Workspace

load('myFileName.mat') % Load saved variables into Workspace

```

<a name="matrices"></a>

## 2. Matrices and Vectors

\*\*IMPORTANT: Indices in Matlab start at 1, not 0\*\*

<a name="declarations"></a>

### i. Declarations

```matlab

% Vectors

x = [4 32 53 7 1]

x(2) % ans = 32

x(2:3) % ans = 32 53

x(2:end) % ans = 32 53 7 1

x = [4; 32; 53; 7; 1] % Column vector

x = [1:10] % x = 1 2 3 4 5 6 7 8 9 10

x = [1:2:10] % Increment by 2, i.e. x = 1 3 5 7 9

% Matrices

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

% Rows are separated by a semicolon; elements are separated with space or comma

% A =

%     1     2     3

%     4     5     6

%     7     8     9

A(2,3) % ans = 6, A(row, column)

A(6) % ans = 8

% (implicitly concatenates columns into vector, then indexes into that)

A(2,3) = 42 % Update row 2 col 3 with 42

% A =

%     1     2     3

%     4     5     42

%     7     8     9

```

<a name="splice"></a>

### ii. Splicing and Dicing

```matlab

A(2:3,2:3) % Creates a new matrix from the old one

%ans =

%     5     42

%     8     9

A(:,1) % All rows in column 1

%ans =

%     1

%     4

%     7

A(1,:) % All columns in row 1

%ans =

%     1     2     3

[A ; A] % Concatenation of matrices (vertically)

%ans =

%     1     2     3

%     4     5    42

%     7     8     9

%     1     2     3

%     4     5    42

%     7     8     9

% this is the same as

vertcat(A,A);

[A , A] % Concatenation of matrices (horizontally)

%ans =

%     1     2     3     1     2     3

%     4     5    42     4     5    42

%     7     8     9     7     8     9

% this is the same as

horzcat(A,A);

A(:, [3 1 2]) % Rearrange the columns of original matrix

%ans =

%     3     1     2

%    42     4     5

%     9     7     8

A(1, :) =[] % Delete the first row of the matrix

A(:, 1) =[] % Delete the first column of the matrix

squeeze(A); % Removes singular dimensions ie. 2x1x3 -> 2x3

```

<a name="arith"></a>

### iii. Arithmetic and Operations

```matlab

transpose(A) % Transpose the matrix, which is the same as:

A one

A' % Concise version of complex transpose

A.' % Concise version of transpose (without taking complex conjugate)

size(A) % ans = 3 3

% Element by Element Arithmetic vs. Matrix Arithmetic

% On their own, the arithmetic operators act on whole matrices. When preceded

% by a period, they act on each element instead. For example:

A \* B % Matrix multiplication

A .\* B % Multiple each element in A by its corresponding element in B

% There are several pairs of functions, where one acts on each element, and

% the other (whose name ends in m) acts on the whole matrix.

exp(A) % exponentiate each element

expm(A) % calculate the matrix exponential

sqrt(A) % take the square root of each element

sqrtm(A) %  find the matrix whose square is A

% Solving matrix equations (if no solution, returns a least squares solution)

% The \ and / operators are equivalent to the functions mldivide and mrdivide

x=A\b % Solves Ax=b. Faster and more numerically accurate than using inv(A)\*b.

x=b/A % Solves xA=b

inv(A) % calculate the inverse matrix

pinv(A) % calculate the pseudo-inverse

% Common matrix functions

zeros(m,n) % m x n matrix of 0's

ones(m,n) % m x n matrix of 1's

diag(A) % Extracts the diagonal elements of a matrix A

diag(x) % Construct a matrix with diagonal elements listed in x, and zeroes elsewhere

eye(m,n) % Identity matrix

linspace(x1, x2, n) % Return n equally spaced points, with min x1 and max x2

inv(A) % Inverse of matrix A

det(A) % Determinant of A

eig(A) % Eigenvalues and eigenvectors of A

trace(A) % Trace of matrix - equivalent to sum(diag(A))

isempty(A) % Tests if array is empty

all(A) % Tests if all elements are nonzero or true

any(A) % Tests if any elements are nonzero or true

isequal(A, B) % Tests equality of two arrays

numel(A) % Number of elements in matrix

triu(x) % Returns the upper triangular part of x

tril(x) % Returns the lower triangular part of x

cross(A,B) %  Returns the cross product of the vectors A and B

dot(A,B) % Returns scalar product of two vectors (must have the same length)

transpose(A) % Returns the transpose of A

fliplr(A) % Flip matrix left to right

flipud(A) % Flip matrix up to down

% Matrix Factorisations

[L, U, P] = lu(A) % LU decomposition: PA = LU,L is lower triangular, U is upper triangular, P is permutation matrix

[P, D] = eig(A) % eigen-decomposition: AP = PD, P's columns are eigenvectors and D's diagonals are eigenvalues

[U,S,V] = svd(X) % SVD: XV = US, U and V are unitary matrices, S has non-negative diagonal elements in decreasing order

[Q, R] = qr(A) % if A is mxn, Q is mxm and R is mxn upper triangular

% Common vector functions

max     % largest component

min     % smallest component

length  % length of a vector

sort    % sort in ascending order

sum     % sum of elements

prod    % product of elements

mode    % modal value

median  % median value

mean    % mean value

std     % standard deviation

perms(x) % list all permutations of elements of x

find(x) % Finds all non-zero elements of x and returns their indexes, can use comparison operators,

        % i.e. find( x == 3 ) returns indexes of elements that are equal to 3

        % i.e. find( x >= 3 ) returns indexes of elements greater than or equal to 3

```

<a name="plots"></a>

## 3. Plots

```matlab

% Plotting

x = 0:.10:2\*pi; % Creates a vector that starts at 0 and ends at 2\*pi with increments of .1

y = sin(x);

plot(x,y)

xlabel('x axis')

ylabel('y axis')

title('Plot of y = sin(x)')

axis([0 2\*pi -1 1]) % x range from 0 to 2\*pi, y range from -1 to 1

plot(x,y1,'-',x,y2,'--',x,y3,':') % For multiple functions on one plot

legend('Line 1 label', 'Line 2 label') % Label curves with a legend

% Alternative method to plot multiple functions in one plot.

% while 'hold' is on, commands add to existing graph rather than replacing it

plot(x, y)

hold on

plot(x, z)

hold off

loglog(x, y) % A log-log plot

semilogx(x, y) % A plot with logarithmic x-axis

semilogy(x, y) % A plot with logarithmic y-axis

fplot (@(x) x^2, [2,5]) % plot the function x^2 from x=2 to x=5

% Creates a meshgrid (2D grid) to calculate a function for every point in the grid

[X, Y] = meshgrid(x\_min:step:x\_max, y\_min:step:y\_max)

grid on % Show grid; turn off with 'grid off'

axis square % Makes the current axes region square

axis equal % Set aspect ratio so data units are the same in every direction

scatter(x, y); % Scatter-plot

hist(x); % Histogram

stem(x); % Plot values as stems, useful for displaying discrete data

bar(x); % Plot bar graph

z = sin(x);

plot3(x,y,z); % 3D line plot

pcolor(A) % Heat-map of matrix: plot as grid of rectangles, coloured by value

contour(A) % Contour plot of matrix

contourf(A) % Filled contour plot of matrix

mesh(A) % Plot as a mesh surface

h = figure % Create new figure object, with handle h

figure(h) % Makes the figure corresponding to handle h the current figure

close(h) % close figure with handle h

close all % close all open figure windows

close % close current figure window

shg % bring an existing graphics window forward, or create new one if needed

clf clear % clear current figure window, and reset most figure properties

% Properties can be set and changed through a figure handle.

% You can save a handle to a figure when you create it.

% The function get returns a handle to the current figure

h = plot(x, y); % you can save a handle to a figure when you create it

set(h, 'Color', 'r')

% 'y' yellow; 'm' magenta, 'c' cyan, 'r' red, 'g' green, 'b' blue, 'w' white, 'k' black

set(h, 'LineStyle', '--')

 % '--' is solid line, '---' dashed, ':' dotted, '-.' dash-dot, 'none' is no line

get(h, 'LineStyle')

% The function gca returns a handle to the axes for the current figure

set(gca, 'XDir', 'reverse'); % reverse the direction of the x-axis

% To create a figure that contains several axes in tiled positions, use subplot

subplot(2,3,1); % select the first position in a 2-by-3 grid of subplots

plot(x1); title('First Plot') % plot something in this position

subplot(2,3,2); % select second position in the grid

plot(x2); title('Second Plot') % plot something there

% Given

x1 = [-3:0.5:3];

x2 = x1;

y = randi(500, length(x1), length(x1));

% Show a 3-D plot

figure

subplot(2,1,1);

surf(x1,x2,y);

xlabel(’x\_1’);

ylabel(’x\_2’);

% Show contours

subplot(2,1,2);

contour(x1,x2,y);

xlabel(’x\_{1}’);

ylabel(’x\_{2}’);

axis equal

% Show a colour map

figure

imagesc(x1,x2,y)

xlabel(’x\_{1}’);

ylabel(’x\_{2}’);

```

<a name="functions"></a>

## 4. Functions and Scripts

```matlab

% Calling Functions

% Standard function syntax:

load('myFile.mat', 'y')

% Command syntax:

load myFile.mat y   % no parentheses, and spaces instead of commas

% Calling a function from a script

% [arguments out] = function\_name(arguments in)

[V,D] = eig(A);

[~,D] = eig(A);  % if you only want D and not V

% To use functions or scripts, they must be on your path or current directory

path % displays current path

addpath /path/to/dir % add to path

rmpath /path/to/dir % remove from path

cd /path/to/move/into % change directory

% M-file Scripts

% A script file is an external file that contains a sequence of statements.

% They let you avoid repeatedly typing the same code in the Command Window

% Have .m extensions

% M-file Functions

% Like scripts, and have the same .m extension

% But can accept input arguments and return an output

% Also, they have their own workspace (ie. different variable scope).

% Function name should match file name (so save this example as double\_input.m).

% 'help double\_input.m' returns the comments under line beginning function

function output = double\_input(x)

    %double\_input(x) returns twice the value of x

    output = 2\*x;

end

double\_input(6) % ans = 12

% If you want to create a function without creating a new file you can use an

% anonymous function.

% Example that returns the square of it's input, assigned to the handle sqr:

sqr = @(x) x.^2;

sqr(10) % ans = 100

doc function\_handle % find out more

```

<a name="logic"></a>

## 5. Programming Logic

```matlab

% User input

a = input('Enter the value: ')

% Stops execution of file and gives control to the keyboard: user can examine

% or change variables. Type 'return' to continue execution, or 'dbquit' to exit

keyboard

% Reading in data (also xlsread/importdata/imread for excel/CSV/image files)

fopen(filename)

% Output

disp(a) % Print out the value of variable a

disp('Hello World') % Print out a string

fprintf % Print to Command Window with more control

% Conditional statements (the parentheses are optional, but good style)

if (a > 15)

    disp('Greater than 15')

elseif (a == 23)

    disp('a is 23')

else

    disp('neither condition met')

end

% Looping

% NB. looping over elements of a vector/matrix is slow!

% Where possible, use functions that act on whole vector/matrix at once

for k = 1:5

    disp(k)

end

k = 0;

while (k < 5)

    k = k + 1;

end

% Timing code execution: 'toc' prints the time since 'tic' was called

tic

A = rand(1000);

A*\*A\**A*\*A\**A*\*A\**A;

toc

```

<a name="math"></a>

## 6. Math/Engineering

<a name="common"></a>

### i. Common Math Functions

```matlab

sin(x)

cos(x)

tan(x)

asin(x)

acos(x)

atan(x)

exp(x)

sqrt(x)

log(x)

log10(x)

abs(x) %If x is complex, returns magnitude

min(x)

max(x)

ceil(x)

floor(x)

round(x)

rem(x)

rand % Uniformly distributed pseudorandom numbers

randi % Uniformly distributed pseudorandom integers

randn % Normally distributed pseudorandom numbers

%Complex math operations

abs(x)   % Magnitude of complex variable x

phase(x) % Phase (or angle) of complex variable x

real(x)  % Returns the real part of x (i.e returns a if x = a +jb)

imag(x)  % Returns the imaginary part of x (i.e returns b if x = a+jb)

conj(x)  % Returns the complex conjugate

% Common constants

pi

NaN

inf

% Given a meshgrid X,Y and a function defined on the meshgrid like Gauss, interpolates the value of the function at the point u1,u2

interp2(X,Y,Gauss,u1,u2)

```

<a name="transfer"></a>

### ii. Transfer Functions

```matlab

% Transfer functions

s = tf('s');

G = s^2/(s^3 + 100\*s^2 + 30\*s + 50);

pole(G); % Returns the location(s) of the pole(s) in rad/s

zero(G); % Returns the location(s) of the zero(s) in rad/s

pzmap(G); % Plots the locations of both the pole(s) and zero(s)

bandwidth(closed\_loop\_system); % Returns bandwidth of a closed loop transfer function in rad/s

bode(closed\_loop\_system) % Creates bode plot of system

rlocus(closed\_loop\_system) % Plots a root locus of the specified system

margin(open\_loop\_system); % Creates a bode plot, displaying the gain and phase margins of an open loop transfer function

```

<a name="vectorization"></a>

## 7. Vectorization

<a href="https://www.mathworks.com/help/matlab/matlab\_prog/vectorization.html">Tips to vectorize your code to get rid of loops and make it run more efficiently.</a>

```matlab

```

<a name="optimization"></a>

## 8. Optimization

```matlab

% fmincon

```

<a name="ML"></a>