MATLAB Cheat Sheet

Prepared by Ali  
Email: [AliGhanbariCs@gmail.com](mailto:AliGhanbariCs@gmail.com)

GitHub: <https://github.com/AliBinary>

Created: November 2023

note: MATLAB® code is sensitive to casing, and insensitive to blank spaces except when defining arrays.

clc # Clear screen

clear x # Clear x variable

clear # Clear all variables

help elfun # Show elementary functions

sin(), cos(), tan(), cot()

exp(), abs()

factorial()

A = [1 2 3 4 5] # Create a row matrix

B = [10; 20; 30] # Create a column matrix

X = [1 5 8 0 0; 9 6 2 0 7; 2 4 5 0 1]

N = [9 3 5; 7 2 4; 1 4 2]

A(end) # Shows last element of matrix (array)

A(:) # Shows all elements in a column

A(9) = 1 # Change 9th element in A(:)

N = 2:7 # Range with specified distance

**K = 10:7:70**

A(1:2:end) # Shows odd elements

A(2:2:end) # Shows even elements

a = 5, b = 7

a \* b # Matrix multiply ( for array multiply use .\* )

a / b, a \ b # Matrix divide ( for array divide use ./ or .\ )

a + b, a - b

a ^ b

sum(1:10)

prod(1:10)

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

M(2,:) # Get second column

M(:,2) # Get second row

M(2,3) # Get element in 2th row and in 3th column

M(1,:) = [] # Delete first column

M(:,3) = [] # Delete third row

M(:,[2,5]) = [] # Delete a few rows

A = [1 2 3]

B = [4 5 6]

C = [A B] # Row concatenation of two matrices

D = [A; B] # Column concatenation of two matrices

length(X) # Length of vector (it is equivalent to MAX(SIZE(X)) )

height(X) # Number of rows in an array (is equivalent to SIZE(X, 1) )

size(X) # Size of array

ndims(X) # Number of dimensions

nnz(X) # Number of non zeros elements

numel(X) # Number of elements

sum(sum(X)) # Sum of all elements

X’ # Transform a matrix

sort(X)

sort(X,1), sort(X,2) # Sort for columns or rows

sort(X, ’descend’), sort(X, ‘ascend’)

sortrows(A,2) # Sort second row of a matrix

min(X), max(X)

mean(X) # Shows Mean of each column as one row

max(max(X)) # Max of elements

tril(X) # Lower triangular matrix

triu(X) # Upper triangular matrix

diag(X) # Main diameter

sum(diag(X)) # Sum of the main diameter reservoirs

fliplr(X) # Flip array in left/right direction

flipud(X) # Flip array in up/down direction

rot90(X) # Rotate array 90 degrees

rot90(X,K) # Rotate array K\*90 degrees

ones(m,n)

zeros(m,n)

eye(m,n)

det(B) # Determinant of a square matrix

inv(B) # Inverse of a square matrix

X = inv(A) \* B # AX = B --> X = A^-1 \* B

magic(n)

pascal(n)

eig(N) # Eigenvalues and eigenvectors

<, >, <=, >= # Less than, Greater than, Less than or equal, Greater than or equal

==, ~= # Equal, Not equal

&, | # Logical or, Logical and

r = X>0

X(r) # Shows elements greater than 0 in X

r = X>5 & X<20

X(r)

rem(-13,2) # Remainder after division

mod(-13,2) # Modulus after division

r = rem(X,2) ~= 0 # Find elements by condition

r = rem(X,2) == 0 & X >5

r = rem(X,2) ~= 0 | X>8

X(r) = X(r) + 100 # Change the specified elements

find(X>0) # Index of the elements that apply to the condition

all(X>0) # columns whose all elements are valid in the condition

any(X>0) # columns where at least one element applies to the condition

rand(m,n) # Uniformly distributed pseudorandom numbers

randn(m,n) # Normally distributed pseudorandom numbers

r = a + (b-a) .\* rand(); # Random number in range (a,b)

randi(100, 1, 5) # 5 Random integers in 1:100

linspace(1,5,100) # Linearly spaced vector

M = 0:0.25:1 # Both included

repmat(X, 2, 5) # Replicate and tile an array

reshape(1:20,4,5) # Reshape array by rearranging existing elements

T = 0 : .01 : 2\*pi

plot(T, Sin(T))

plot(x, y, ‘color style marker’)

grid on # Grid lines

grid off

axis on # Control axis scaling and appearance

axis off

axis([XMIN XMAX YMIN YMAX])

figure # Create figure window

hold on # Hold current graph for next plot

hold off

subplot(m, n, k) # Create axes in tiled positions (m rows, n columns, k=current window)

clf # Clears Figure

close # close all figures

xlabel(“text”)

ylabel(“text”)

zlabel(“text”)  
title(“text”)

text(x, y, “text”) # Add text descriptions to data points

gtext(“text”) # Place text with mouse

legend(‘Label 1’, ‘Label 2’) # Create legend

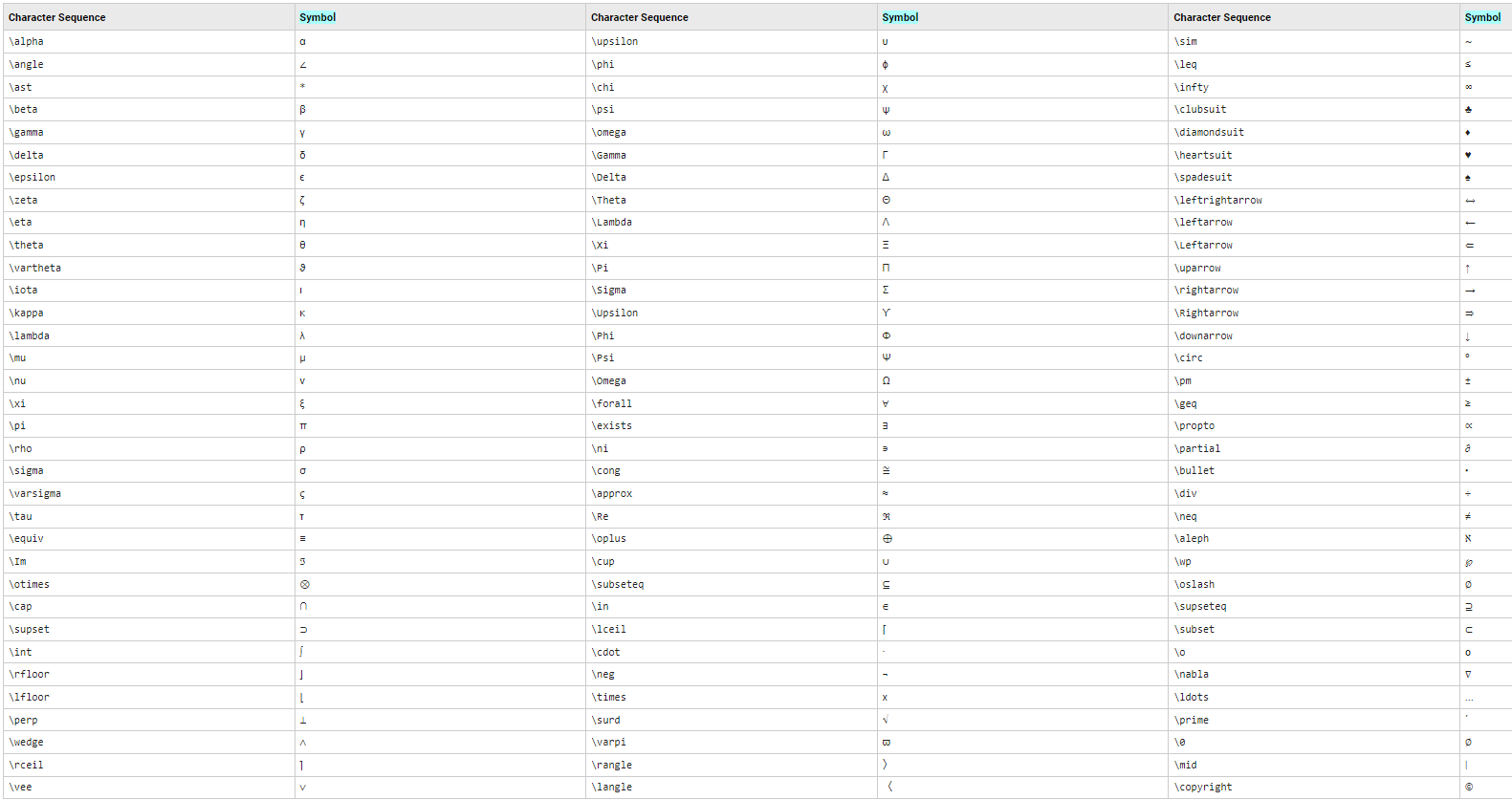
gtext(‘\int\_0^5 x.^2 dx’) # Mathematical symbols

\sum\_{n=0}^{\infty}

\infty

\x^{y+1}

\sin



Line(x, y, z) # Create line

rectangle(‘position’,[1 2 5 6], ‘curvature’,1) # Create rectangle, rounded-rectangle, or ellipse

triplot(TRI, x, y) # Plots a 2D triangulation

r = rectangle(‘position’, [0 0 1 1]) # Create and customize a rectangle

r.FaceColor = [0 .5 .5];

r.EdgeColor = ‘b’;

r.LineWidth = 3;

x = [1 2 3]; y = [2 3 2]; TRI = [1 2 3];

triplot(TRI, x, y)

axis([0 5 0 5])

plot3(x, y, z, ‘color style marker’)

box on # Adds a box to the current axes

box off # Removes the box from the current axes

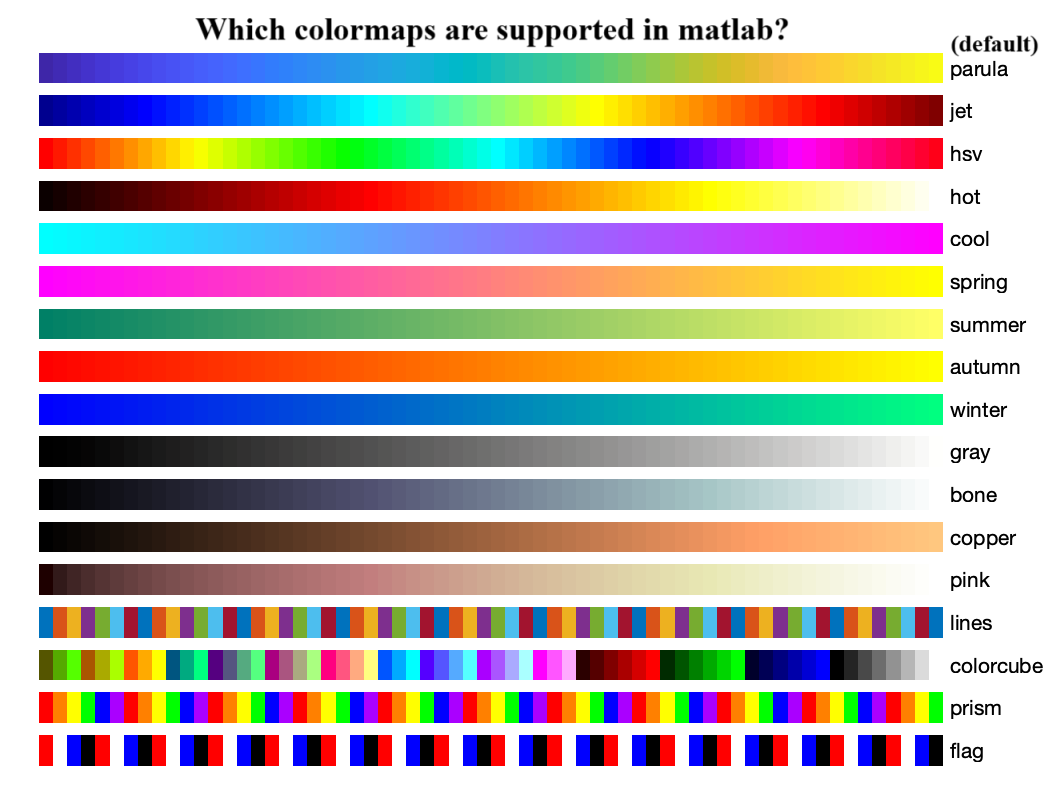
comet3(x, y, z) # 3-D comet-like trajectories

[X, Y] = meshgrid(x, y) # Cartesian rectangular grid in 2-D or 3-D

surf(X, Y, Z) # 3-D colored surface

mesh(X, Y, Z) # 3-D mesh surface

colormap(hot) # Sets the Colormap property of a figure



colormap(‘default’)

colormap([1 1 1]) # RGB (between 0 and 1)

shading flat # Sets the shading of the current graph to flat

shading interp # Sets the shading to interpolated

shading faceted # Sets the shading to faceted, which is the default

sphere(N) # Graph the sphere as a SURFACE and do not return anything

[X,Y,Z] = sphere(N) # Generates three (N+1)by(N+1) matrices

surf(X,Y,Z) # Produces a unit sphere

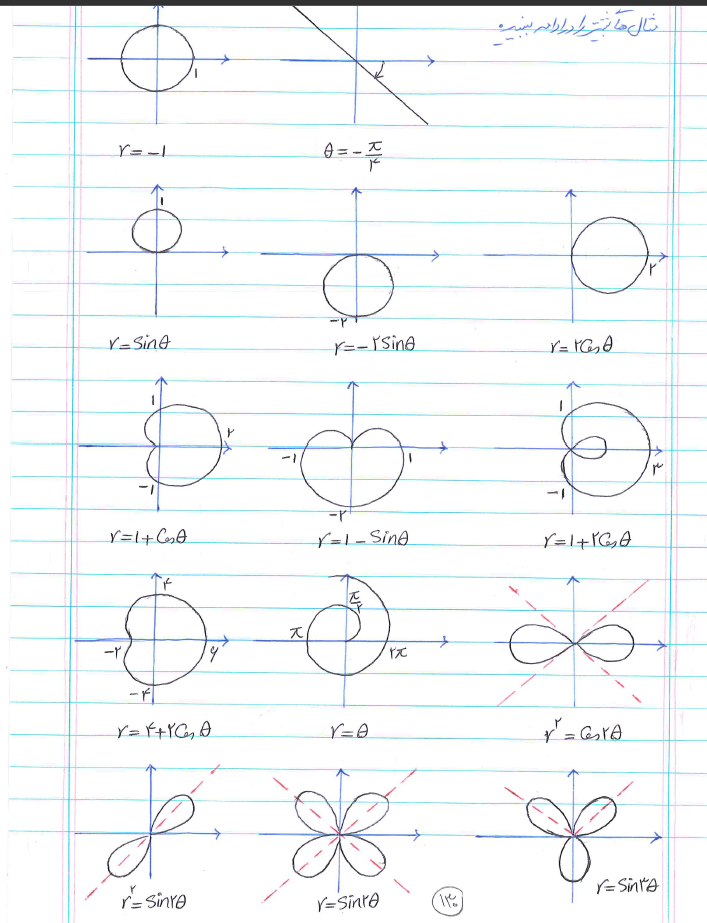
axis equal # equal tick mark increments on the x-,y- and z-axis are equal in size

axis square # makes the current axis box square in size

cylinder(R,N) # forms the unit cylinder based on the generator curve in the vector R. The cylinder has N points around the circumference.

[X,Y,Z] = cylinder(R) # Default to N = 20, you can graph it with MESH or SURF

polar(THETA, RHO) # makes a plot using polar coordinates of the angle THETA, in radians, versus the radius RHO.



t = 0 : .01 : 2\*pi ; r = 5

x = R \* cos(t)

y = R \* sin(t)

plot(x, y) # Draws a circle with radius R

polar(t, R + zeros(size(t)) # Draws a circle with radius R

pos = [0 0 R R];

rectangle(‘position’, pos, ‘curvature’, [1 1]) # Draws a circle with radius R

axis equal

ezplot(‘sin(x)’) # (NOT RECOMMENDED) Easy to use function plotter

ezplot3(‘sin(x)’, ‘cos(x)’, ‘x’, [0, 2.\* pi], ‘animate’)

# (NOT RECOMMENDED) Easy to use 3-d parametric curve plotter

ezmesh(‘tan(x)’) # (NOT RECOMMENDED) Easy to use 3-D mesh plotter

ezsurf(‘tanh(x)’) # (NOT RECOMMENDED) Easy to use 3-D colored surface plotter

ezpolar(‘1+cos(t)’) # Easy to use polar coordinate plotter

fill(x, y, color) # Filled 2-D polygons

x = 0 : .01 : 2\*pi;

y = sin(x)

t = x(end : -1 : 1) # The same as t = fliplr(x)

u = cos(t)

a = [x t]; b = [y u]

fill(a, b, ‘g’ ) # Coloring the area between sine and cosine

p=[3 0 -5] # p(x)=3x^2-5

x=roots(p) # Find polynomial roots

poly(x) # Convert roots to polynomial

conv(A, B) # Convolution and polynomial multiplication

[X, R] = deconv(Y, H) # Least-squares deconvolution and polynomial division

Y = polyval(P, X) # Evaluate polynomial

polyder(P) # Differentiate polynomial

polyint(P) # Integrate polynomial analytically

F = polyfit(X, Y, N) # Fit polynomial to data

hold on

plot(X, polyval(F, X), ‘b’)

%----------------------------------------

if # Conditionally execute statements

switch # Switch among several cases based on expression

for # Repeat statements a specific number of times

while # Repeat statements an indefinite number of times

# To see their structure, refer to their help

sprintf(‘this car cost %d dollars!’, variable)

# %d for integers, %15.5f for floats, %s for strings

[d, id] = sort(diag(a))

a = a(id, :) # Moves the rows of the matrix so that the main diameter of the matrix is ​​arranged

x = 0:0.2:2\*pi; # Star motion on sine graph

for j = 1:2

for i=1:numel(x)

plot(x,sin(x),’b’)

hold on

plot(x(i),sin(x(i)),’r\*’)

pause(0.3)

hold off

end

x = fliplr(x)

end

some shortcuts:{

ctrl+C # Stop the program

ctrl+R # Comment

ctrl+T # Uncomment

ctrl+I # Beautify the code

}

nargin # Number of function input arguments

nargout # Number of function output arguments

help symbolic math toolbox

f = ‘x^2 + 2\*sin(x) + 7\*y’ # The first way to define an equation

syms x y # The second way

f = x^2 + 2\*sin(x) + 7\*y

symvar(f) # Determine the symbolic variables in an expression

diff(f, x, n) # Difference and approximate derivative

limit(f, x, a, ‘right’) # Limit of an expression

syms x

f = -2\*x/(1+x^2)^2

int(f, x) # Calculation of indefinite integral

int(f, x, 1, 5) # Calculate the definite integral

syms n

f = 1/n^2

symsum(f, n, 1, inf) # Symbolic summation

pretty(y) # Pretty print a symbolic expression

simplify(y) # Algebraic simplification

syms x y z

f = x+y\*a+7

subs(f, [x z], [10 5]) # Symbolic substitution

eqn1 = x+2\*y-z==5;

eqn2= 3\*x-2\*y+2\*z==7;

eqn3= x+y+z==3;

s = solve([eqn1, eqn2, eqn3], [x, y, z]); # Symbolic solution of algebraic equations