Cheatography

MATLAB Cheat Sheet

by quantumrustler via cheatography.com/212850/cs/46338/

MATLAB Environment

`clc`	Clear command window
`clear`	Remove items from workspace. You can specify which variables or functions to clear
`clear all`	Fresh start without any leftover variables, functions, or compiled functions
`input( 'string with a message that is displayed in the Command Window' )`	Prompts the user for input and returns the entered value
`fprint f(' text', values)`	Displays formatted text output; supports format specifiers like %d (integer), %f (floating point), %s (string), %c (character), exponential notation (%e), and \n (newline)
`disp(v alue)` `disp(' text')`	Quickly displays text or variables

`function [output arguments] = functi on_ nam e(input arguments)`
`.....`
`end`
	Function definition line. Save your function in a function file or at the end of a script file. Function files must have the same name as the 1st function

`name = @ (arglist) expr`	Anonymous function; arglist is a list of independent variables separated by comma

Operators and Special Characters

`+, -, *, /, ^`	Matrix math operations (`X = DC^-1 = D / C`)
`\`	Left division (`a / b = b \ a`) (`X = A^-1B = A \ B`) or linear optimization
`.*, ./, .\, .^`	Element-wise operations
`'`	Transpose
`==, ~=, <, >, <=, >=`	Relational operators If an operand is an array, result is logical array
`&, |, ~`	Logical operations (AND, NOT, OR)
`;`	Suppress output display
`%`	Comment

Operators and Special Characters (cont)

`%{ ... %}`	Multiline Comment
`'Hello'`	Definition of a character vector
`"This is a string "`	Definition of a string
`str1 + str2`	Append strings

Defining and Changing Array Variables

`a = 5`	Define variable a with value 5
`x = y`	Define variable x with value assigned to y
`[m:q:n]` (The brackets are optional)	Create a vector with constant spacing by specifying the first term, the spacing, and the last term
`linspa ce( xi, xf,n)`	Creating a vector with constant spacing by specifying the first and last terms, and the number of terms

`A = [1 2 3; 4 5 6]`
`A = [1 2 3`
`4 5 6]`
	Define A as a 2x3 matrix
	“space” separates columns “;” or new line separates rows

`zeros(m,n)`	Create m x n matrix of zeros
`ones(m,n)`	Create m x n matrix of ones
`eye(n)`	Create an n x n identity matrix

`va(k)`
`va(x, y, z)`
`va(x, k:p, y, z)`
	Refers to the kth element of the vector ve

`va(k) = []`	Deletes kth element
`va(:)`	Refers to all the elements of the vector va
`va(m:n)`	Refers to elements m through n of the vector va
`A(k,p)`	Refers to the element at kth row and pth column of matrix A
`A(:,n)`	Refers to the elements in all the rows of column n of the matrix A
`A(n,:)`	Refers to the elements in all the columns of row n of the matrix A

Defining and Changing Array Variables (cont)

`A(:,m:n)`	Refers to the elements in all the rows between columns m and n of the matrix A
`A(m:n,:)`	Refers to the elements in all the columns between rows m and n of the matrix A
`A(m:n,p:q)`	Refers to the elements in rows m through n and columns p through q of the matrix A.

Plotting

`plot(x ,y, 'Li neS pec ifier', 'Prope rty Name', Proper - tyV alue)`
`plot(x ,y ,u, v,t,h))`
Line styles:
-, --, :, -
Markers:
+, o, *, ., x, s, d
Colors:
r, g, b, c, m, y, k, w
Property names:
LineWidth
MarkerSize
MarkerEdgeColor
MarkerFaceColor
	Plot y vs. x (`LineSp ecifier`, `Proper tyName`, and `Proper tyValue` are optional)
	`LineSp ecifer` is a combination of line style, marker, and color as a string

`subplo t(m ,n,p)`	Divides the Figure Window into m × n rectangular subplots; the command makes the subplot p current
`fplot( 'fu nct ion ',l imi ts, 'line specif iers')`	Plots a function with the form y = f(x) between specified limits (domain of x and, optionally, the limits of the y axis ([xmin, xmax, ymin, ymax]))
`line(x ,y, 'Pr ope rty Name', Proper tyV alue)`	Add additional graph (line) to a plot that already exists

Plotting (cont)

`hold on`
.....
`hold off`
	Retain the current plot when adding new plots

`title( " Tit le")`	Add plot title
`legend ("1s t", " 2nd ")`	Add legend to axes
`xlabel ("la bel ")`	Add x-axis label
`ylabel ("la bel ")`	Add y-axis label
`axis[xmin xmax ymin ymax]`	Sets the limits of the x-axis and y-axis on a plot to the specified values
`close all`	Closes all Figure Windows that are open

Control Structures

`if` conditional expression
.....
`elseif` conditional expression
.....
`else`
.....
`end`
	if-elseif-else-end structure

`switch` switch expression
`case` value1
.....
`case` value2
.....
`otherwise`
.....
`end`
	switch-case statement

`for k = f:s:t`
.....
`end`
	for-end loop, where k is the loop variable, f is the value of k in the first pass, s is the increment after each pass, and t is the value of k in the last pass

Control Structures (cont)

`while` conditional expression ..... `end`	while-end loop
`break`	Terminate execution of for- or while-loop
`continue`	Pass control to the next iteration of a loop

Special Variables and Constants

`ans`	Most recent answer
`pi`	The number π
`eps`	The smallest difference between two numbers. Equal to 2^(–52), which is approximately 2.2204e–016
`Inf`, `inf`	infinity
`NaN`, `nan`	Not a number (i.e., division by zero)
`real(x)`	Returns the real part of a complex number x
`imag(x)`	Returns the imaginary part of a complex number x

Elementary Math Functions

`sqrt(x)`, `nthroo t(x,n )`	Square root, Real nth root of a real number x. (If x is negative n must be an odd integer.)
`exp(x)`	Exponential of x
`abs(x)`	Absolute value of x
`log(x)`	Natural logarithm i.e. Base e logarithm (ln).
`log2(x)`, `log10(x)`	Logarithm with base 2 and 10, respectively
`factor ial(n)`	The factorial function n! (n must be a positive integer.)

Trigonometric Math Functions

`sin(x)`, `asin(x)`	Sine and inverse (argument in radians)
`sind(x)`, `asind(x)`	Sine and inverse (argument in degrees)
`cos(x)`, `acos(x)`	Cosine and inverse (argument in radians)
`cosd(x)`, `acosd(x)`	Cosine and inverse (argument in degrees)
`tan(x)`, `atan(x)`	Tan and inverse (argument in radians)

Trigonometric Math Functions (cont)

`tand(x)`, `atand(x)`	Tan and inverse (argument in degrees)

Analogous for the other trigonometric functions: `csc`, `sec`, and `cot`

Rounding Functions

`round(x, n)`	Round to the specified number of decimal places (n)
`fix(x)`	Round toward zero
`ceil(x)`	Round toward infinity
`floor(x)`	Round toward minus infinity
`rem(x,y)`	Returns the remainder after x is divided by y
`sign(x)`	Signum function. Returns 1 if x > 0, -1 if x < 0, and 0 if x = 0

Built-In Functions for Handling Arrays

`length(A)`	Returns the number of elements in the vector A
`size(A)`	Returns a row vector [m,n], where m and n are the size m x n of the array A
`reshape(A, m,n)`	Rearrange a matrix A that has r rows and s columns to have m rows and n columns. r times s must be equal to m times n
`diag(v)`	When v is a vector, creates a square matrix with the elements of v in the diagonal
`diag(A)`	When A is a matrix, creates a vector from the diagonal elements of A.

Built-in Functions for Analyzing Arrays

`mean(A)`	If A is a vector, returns the mean value of the elements of the vector
`C = max(A)`	If A is a vector, C is the largest element in A. If A is a matrix, C is a row vector of column max
`[d, n] = max(A)`	If A is a vector, d is the largest element in A, n is the position of the element (first if there are duplicates)
`min(A)`	Returns the smallest element of A
`[d, n] = min(A)`	d is the smallest element in A, n is the position (first if there are duplicates)
`sum(A)`	If A is a vector, returns the sum of the elements of the vector
`sort(A)`	If A is a vector, arranges the elements of the vector in ascending order
`median(A)`	If A is a vector, returns the median value of the elements of the vector
`det(A)`	Returns the determinant of a square matrix A.
`dot(a, b)`	Calculates the scalar (dot) product of two vectors a and b. The vectors can each be row or column vectors
`cross(a, b)`	Calculates the cross product of two vectors a and b, (a×b). The two vectors must have 3 elements.
`inv(A)`	Returns the inverse of a square matrix A

Built-in Logical Functions

`and(A,B)`	equivalent to A&B
`or(A,B)`	equivalent to A|B
`not(A)`	equivalent to ~A
`xor(a,b)`	Exclusive or. Returns true (1) if one operand is true and the other is false
`all(A)`	Returns 1 (true) if all elements in a vector A are true (nonzero). Returns 0 (false) if one or more elements are false (zero). If A is a matrix, treats columns of A as vectors, and returns a vector with 1s and 0s
`find(A)`	If A is a vector, returns the indices of the nonzero elements
`find(A >d )`	If A is a vector, returns the address of the elements that are larger than d (any relational operator can be used)

Polynomials and Interpolation

`polyva l(p,x)`	Calculates the value of a polynomial at a point x
`roots(p)`	Determines the root, or roots, of a polynomial
`p = poly(r)`	Determines the coefficients of the polynomial when the roots of a polynomial are known
`conv(a,b)`	Multiplies two polynomials
`[q,r] = deconv (u,v )`	Divides two polynomials and returns a vector with the coefficients of the quotient (q) as well as a vector with the coefficients of the remainder polynomial (r)

Polynomials and Interpolation (cont)

`k = polyder(p)`	Derivative of a single polynomial. p is a vector with the coefficients of the polynomial. k is a vector with the coefficients of the polynomial that is the derivative
`k = polyde r(a,b)`	Derivative of a product of two polynomials. a and b are vectors with the coefficients of the polynomials that are multiplied. k is a vector with the coefficients of the polynomial that is the derivative of the product
`[n d] = polyde r(u,v)`	Derivative of a quotient of two polynomials. u and v are vectors with the coefficients of the numerator and denominator polynomials. n and d are vectors with the coefficients of the numerator and denominator polynomials in the quotient that is the derivative

Polynomials and Interpolation (cont)

`p = polyfi t(x ,y,n)`	Polynomial curve fitting. p is the vector of the coefficients of the polynomial that fits the data. x is a vector with the horizontal coordinates of the data points (independent variable). y is a vector with the vertical coordinates of the data points (dependent variable). n is the degree of the polynomial.
`yi = interp 1(x ,y, xi, 'me t hod')` Methods: nearest (xi must be within the domain of x) linear (xi must be within the domain of x) spline (xi can have values outside the domain of x) pchip (xi can have values outside the domain of x)	One-dimensional interpolation (the last character is the number one) xi is domain