

MATLAB Basic Functions Reference

MATLAB Environment

<code>clc</code>	Clear command window
<code>help fun</code>	Display in-line help for <code>fun</code>
<code>doc fun</code>	Open documentation for <code>fun</code>
<code>load("filename","vars")</code>	Load variables from <code>.mat</code> file
<code>uiimport("filename")</code>	Open interactive import tool
<code>save("filename","vars")</code>	Save variables to file
<code>clear item</code>	Remove items from workspace
<code>examplescript</code>	Run the script file named <code>examplescript</code>
<code>format style</code>	Set output display format
<code>ver</code>	Get list of installed toolboxes
<code>tic, toc</code>	Start and stop timer
<code>Ctrl+C</code>	Abort the current calculation

Operators and Special Characters

<code>+, -, *, /</code>	Matrix math operations
<code>.*, ./</code>	Array multiplication and division (element-wise operations)
<code>^, .^</code>	Matrix and array power
<code>\</code>	Left division or linear optimization
<code>.', '</code>	Normal and complex conjugate transpose
<code>==, ~=, <, >, <=, >=</code>	Relational operators
<code>&&, , ~, xor</code>	Logical operations (AND, NOT, OR, XOR)
<code>;</code>	Suppress output display
<code>...</code>	Connect lines (with break)
<code>% Description</code>	Comment
<code>'Hello'</code>	Definition of a character vector
<code>"This is a string"</code>	Definition of a string
<code>str1 + str2</code>	Append strings

Special Variables and Constants

<code>ans</code>	Most recent answer
<code>pi</code>	$\pi=3.141592654\dots$
<code>i, j, 1i, 1j</code>	Imaginary unit
<code>NaN, nan</code>	Not a number (i.e., division by zero)
<code>Inf, inf</code>	Infinity
<code>eps</code>	Floating-point relative accuracy

Defining and Changing Array Variables

<code>a = 5</code>	Define variable <code>a</code> with value 5
<code>A = [1 2 3; 4 5 6]</code> <code>A = [1 2 3 4 5 6]</code>	Define <code>A</code> as a 2x3 matrix "space" separates columns ";" or new line separates rows
<code>[A,B]</code>	Concatenate arrays horizontally
<code>[A;B]</code>	Concatenate arrays vertically
<code>x(4) = 7</code>	Change 4th element of <code>x</code> to 7
<code>A(1,3) = 5</code>	Change <code>A(1,3)</code> to 5
<code>x(5:10)</code>	Get 5th to 10th elements of <code>x</code>
<code>x(1:2:end)</code>	Get every 2nd element of <code>x</code> (1st to last)
<code>x(x>6)</code>	List elements greater than 6
<code>x(x==10)=1</code>	Change elements using condition
<code>A(4,:)</code>	Get 4th row of <code>A</code>
<code>A(:,3)</code>	Get 3rd column of <code>A</code>
<code>A(6, 2:5)</code>	Get 2nd to 5th element in 6th row of <code>A</code>
<code>A(:,[1 7])=A(:,[7 1])</code>	Swap the 1st and 7th column
<code>a:b</code>	<code>[a, a+1, a+2, ..., a+n]</code> with $a+n \leq b$
<code>a:ds:b</code>	Create regularly spaced vector with spacing <code>ds</code>
<code>linspace(a,b,n)</code>	Create vector of <code>n</code> equally spaced values
<code>logspace(a,b,n)</code>	Create vector of <code>n</code> logarithmically spaced values
<code>zeros(m,n)</code>	Create <code>m</code> x <code>n</code> matrix of zeros
<code>ones(m,n)</code>	Create <code>m</code> x <code>n</code> matrix of ones
<code>eye(n)</code>	Create a <code>n</code> x <code>n</code> identity matrix
<code>A=diag(x)</code>	Create diagonal matrix from vector
<code>x=diag(A)</code>	Get diagonal elements of matrix
<code>meshgrid(x,y)</code>	Create 2D and 3D grids
<code>rand(m,n), randi</code>	Create uniformly distributed random numbers or integers
<code>randn(m,n)</code>	Create normally distributed random numbers

Complex Numbers

<code>i, j, 1i, 1j</code>	Imaginary unit
<code>real(z)</code>	Real part of complex number
<code>imag(z)</code>	Imaginary part of complex number
<code>angle(z)</code>	Phase angle in radians
<code>conj(z)</code>	Element-wise complex conjugate
<code>isreal(z)</code>	Determine whether array is real

Elementary Functions

<code>sin(x)</code> , <code>asin</code>	Sine and inverse (argument in radians)
<code>sind(x)</code> , <code>asind</code>	Sine and inverse (argument in degrees)
<code>sinh(x)</code> , <code>asinh</code>	Hyperbolic sine and inverse (arg. in radians)
Analogous for the other trigonometric functions: <code>cos</code> , <code>tan</code> , <code>csc</code> , <code>sec</code> , and <code>cot</code>	
<code>abs(x)</code>	Absolute value of x, complex magnitude
<code>exp(x)</code>	Exponential of x
<code>sqrt(x)</code> , <code>nthroot(x,n)</code>	Square root, real nth root of real numbers
<code>log(x)</code>	Natural logarithm of x
<code>log2(x)</code> , <code>log10</code>	Logarithm with base 2 and 10, respectively
<code>factorial(n)</code>	Factorial of n
<code>sign(x)</code>	Sign of x
<code>mod(x,d)</code>	Remainder after division (modulo)
<code>ceil(x)</code> , <code>fix</code> , <code>floor</code>	Round toward +inf, 0, -inf
<code>round(x)</code>	Round to nearest decimal or integer

Tables

<code>table(var1,...,varN)</code>	Create table from data in variables var1, ..., varN
<code>readtable("file")</code>	Create table from file
<code>array2table(A)</code>	Convert numeric array to table
<code>T.var</code>	Extract data from variable var
<code>T(rows,columns)</code> , <code>T(rows,["col1","coln"])</code>	Create a new table with specified rows and columns from T
<code>T.varname=data</code>	Assign data to (new) column in T
<code>T.Properties</code>	Access properties of T
<code>categorical(A)</code>	Create a categorical array
<code>summary(T)</code> , <code>groupsummary</code>	Print summary of table
<code>join(T1, T2)</code>	Join tables with common variables

Tasks (Live Editor)

Live Editor tasks are apps that can be added to a live script to interactively perform a specific set of operations. Tasks represent a series of MATLAB commands. To see the commands that the task runs, show the generated code.

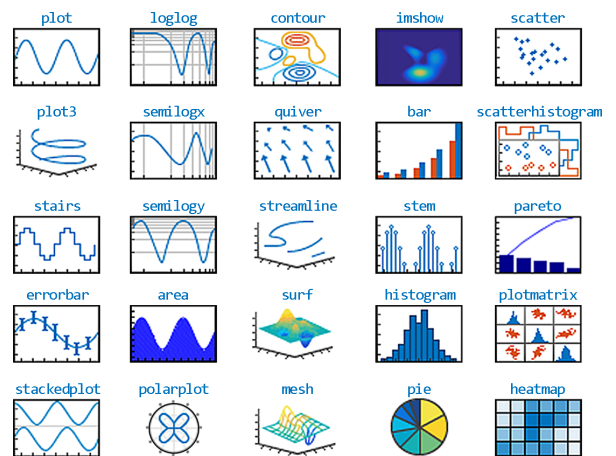
Common tasks available from the Live Editor tab on the desktop toolstrip:

- Clean Missing Data
- Clean Outlier
- Find Change Points
- Find Local Extrema
- Remove Trends
- Smooth Data

Plotting

<code>plot(x,y,LineStyle)</code>	Plot y vs. x (<code>LineStyle</code> is optional) <code>LineStyle</code> is a combination of <code>linestyle</code> , <code>marker</code> , and <code>color</code> as a string. Example: <code>"-r"</code> = red solid line without markers
Line styles: <code>-</code> , <code>--</code> , <code>:</code> , <code>-.</code>	
Markers: <code>+</code> , <code>o</code> , <code>*</code> , <code>.</code> , <code>x</code> , <code>s</code> , <code>d</code>	
Colors: <code>r</code> , <code>g</code> , <code>b</code> , <code>c</code> , <code>m</code> , <code>y</code> , <code>k</code> , <code>w</code>	
<code>title("Title")</code>	Add plot title
<code>legend("1st", "2nd")</code>	Add legend to axes
<code>x/y/zlabel("label")</code>	Add x/y/z axis label
<code>x/y/zticks(ticksvec)</code>	Get or set x/y/z axis ticks
<code>x/y/zticklabels(labels)</code>	Get or set x/y/z axis tick labels
<code>x/y/ztickangle(angle)</code>	Rotate x/y/z axis tick labels
<code>x/y/zlim</code>	Get or set x/y/z axis range
<code>axis(lim)</code> , <code>axis style</code>	Set axis limits and style
<code>text(x,y,"txt")</code>	Add text
<code>grid on/off</code>	Show axis grid
<code>hold on/off</code>	Retain the current plot when adding new plots
<code>subplot(m,n,p)</code> , <code>tiledlayout(m,n)</code>	Create axes in tiled positions
<code>yyaxis left/right</code>	Create second y-axis
<code>figure</code>	Create figure window
<code>gcf</code> , <code>gca</code>	Get current figure, get current axis
<code>clf</code>	Clear current figure
<code>close all</code>	Close open figures

Common Plot Types



Plot Gallery: mathworks.com/products/matlab/plot-gallery

Programming Methods

Functions

```
% 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
function cavg = cumavg(x) %multiple args. possible
    cavg=cumsum(x)./(1:length(x)) ;
end
```

Anonymous Functions

```
% defined via function handles
fun = @(x) cos(x.^2)./abs(3*x);
```

Control Structures

if, elseif Conditions

```
if n<10
    disp("n smaller 10")
elseif n<=20
    disp("n between 10 and 20")
else
    disp("n larger than 20")
```

Switch Case

```
n = input("Enter an integer: ");
switch n
    case -1
        disp("negative one")
    case {0,1,2,3} % check four cases together
        disp("integer between 0 and 3")
    otherwise
        disp("integer value outside interval [-1,3]")
end % control structures terminate with end
```

For-Loop

```
% loop a specific number of times, and keep
% track of each iteration with an incrementing
% index variable
for i = 1:3
    disp("cool");
end % control structures terminate with end
```

While-Loop

```
% loops as long as a condition remains true
n = 1;
nFactorial = 1;
while nFactorial < 1e100
    n = n + 1;
    nFactorial = nFactorial * n;
end % control structures terminate with end
```

Further programming/control commands

break	Terminate execution of for- or while-loop
continue	Pass control to the next iteration of a loop
try, catch	Execute statements and catch errors

Numerical Methods

fzero(fun,x0)	Root of nonlinear function
fminsearch(fun,x0)	Find minimum of function
fminbnd(fun,x1,x2)	Find minimum of fun in [x1, x2]
fft(x), ifft(x)	Fast Fourier transform and its inverse

Integration and Differentiation

integral(f,a,b)	Numerical integration (analogous functions for 2D and 3D)
trapz(x,y)	Trapezoidal numerical integration
diff(x)	Differences and approximate derivatives
gradient(X)	Numerical gradient
curl(X,Y,Z,U,V,W)	Curl and angular velocity
divergence(X,...,W)	Compute divergence of vector field
ode45(ode,tspan,y0)	Solve system of nonstiff ODEs
ode15s(ode,tspan,y0)	Solve system of stiff ODEs
deval(sol,x)	Evaluate solution of differential equation
pdepe(m,pde,ic,...,bc,xm,ts)	Solve 1D partial differential equation
pdeval(m,xmesh,...,usol,xq)	Interpolate numeric PDE solution

Interpolation and Polynomials

interp1(x,v,xq)	1D interpolation (analogous for 2D and 3D)
pchip(x,v,xq)	Piecewise cubic Hermite polynomial interpolation
spline(x,v,xq)	Cubic spline data interpolation
ppval(pp,xq)	Evaluate piecewise polynomial
mkpp(breaks,coeffs)	Make piecewise polynomial
unmkpp(pp)	Extract piecewise polynomial details
poly(x)	Polynomial with specified roots x
polyeig(A0,A1,...,Ap)	Eigenvalues for polynomial eigenvalue problem
polyfit(x,y,d)	Polynomial curve fitting
residue(b,a)	Partial fraction expansion/decomposition
roots(p)	Polynomial roots
polyval(p,x)	Evaluate poly p at points x
polyint(p,k)	Polynomial integration
polyder(p)	Polynomial differentiation

Matrices and Arrays

<code>length(A)</code>	Length of largest array dimension
<code>size(A)</code>	Array dimensions
<code>numel(A)</code>	Number of elements in array
<code>sort(A)</code>	Sort array elements
<code>sortrows(A)</code>	Sort rows of array or table
<code>flip(A)</code>	Flip order of elements in array
<code>squeeze(A)</code>	Remove dimensions of length 1
<code>reshape(A,sz)</code>	Reshape array
<code>repmat(A,n)</code>	Repeat copies of array
<code>any(A, all)</code>	Check if any/all elements are nonzero
<code>nnz(A)</code>	Number of nonzero array elements
<code>find(A)</code>	Indices and values of nonzero elements

Linear Algebra

<code>rank(A)</code>	Rank of matrix
<code>trace(A)</code>	Sum of diagonal elements of matrix
<code>det(A)</code>	Determinant of matrix
<code>poly(A)</code>	Characteristic polynomial of matrix
<code>eig(A, eigs)</code>	Eigenvalues and vectors of matrix (subset)
<code>inv(A), pinv</code>	Inverse and pseudo inverse of matrix
<code>norm(x)</code>	Norm of vector or matrix
<code>expm(A), logm</code>	Matrix exponential and logarithm
<code>cross(A,B)</code>	Cross product
<code>dot(A,B)</code>	Dot product
<code>kron(A,B)</code>	Kronecker tensor product
<code>null(A)</code>	Null space of matrix
<code>orth(A)</code>	Orthonormal basis for matrix range
<code>tril(A), triu</code>	Lower and upper triangular part of matrix
<code>linsolve(A,B)</code>	Solve linear system of the form $AX=B$
<code>lsqminnorm(A,B)</code>	Least-squares solution to linear equation
<code>qr(A), lu, chol</code>	Matrix decompositions
<code>svd(A)</code>	Singular value decomposition
<code>gsvd(A,B)</code>	Generalized SVD
<code>rref(A)</code>	Reduced row echelon form of matrix

Descriptive Statistics

<code>sum(A), prod</code>	Sum or product (along columns)
<code>max(A), min, bounds</code>	Largest and smallest element
<code>mean(A), median, mode</code>	Statistical operations
<code>std(A), var</code>	Standard deviation and variance
<code>movsum(A,n), movprod, movmax, movmin, movmean, movmedian, movstd, movvar</code>	Moving statistical functions n = length of moving window
<code>cumsum(A), cumprod, cummax, cummin</code>	Cumulative statistical functions
<code>smoothdata(A)</code>	Smooth noisy data
<code>histcounts(X)</code>	Calculate histogram bin counts
<code>corrcoef(A), cov</code>	Correlation coefficients, covariance
<code>xcorr(x,y), xcov</code>	Cross-correlation, cross-covariance
<code>normalize(A)</code>	Normalize data
<code>detrend(x)</code>	Remove polynomial trend
<code>isoutlier(A)</code>	Find outliers in data

Symbolic Math*

<code>sym x, syms x y z</code>	Declare symbolic variable
<code>eqn = y == 2*a + b</code>	Define a symbolic equation
<code>solve(eqns,vars)</code>	Solve symbolic expression for variable
<code>subs(expr,var,val)</code>	Substitute variable in expression
<code>expand(expr)</code>	Expand symbolic expression
<code>factor(expr)</code>	Factorize symbolic expression
<code>simplify(expr)</code>	Simplify symbolic expression
<code>assume(var,assumption)</code>	Make assumption for variable
<code>assumptions(z)</code>	Show assumptions for symbolic object
<code>fplot(expr), fcontour, fsurf, fmesh, fimplicit</code>	Plotting functions for symbolic expressions
<code>diff(expr,var,n)</code>	Differentiate symbolic expression
<code>dsolve(deqn,cond)</code>	Solve differential equation symbolically
<code>int(expr,var,[a, b])</code>	Integrate symbolic expression
<code>taylor(fun,var,z0)</code>	Taylor expansion of function

*requires Symbolic Math Toolbox