

# 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>==, ~=, &lt;, &gt;, &lt;=, &gt;=</code>	Relational operators
<code>&amp;&amp;,   , ~, 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>	$\pi=3.141592654...$
<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&gt;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.

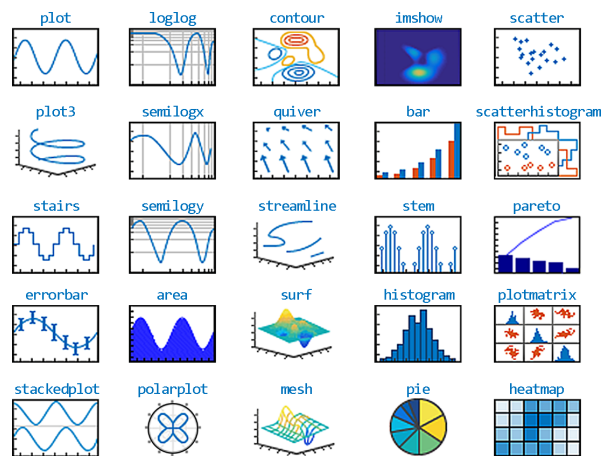
Common tasks available from the Live Editor tab on the desktop toolstrip:

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

## Plotting

<code>plot(x,y,LineSpec)</code> 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>	Plot y vs. x ( <b>LineSpec</b> is optional) <b>LineSpec</b> is a combination of <b>linestyle</b> , <b>marker</b> , and <b>color</b> as a string. Example: <code>"-r"</code> = red solid line without markers
<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](https://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(vec)./(1:length(vec));
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

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

## Numerical Methods

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

## Integration and Differentiation

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

## Interpolation and Polynomials

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

## Matrices and Arrays

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

## Linear Algebra

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

## Descriptive Statistics

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

## Symbolic Math\*

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

\*requires Symbolic Math Toolbox