# List of Matlab commands

## General Purpose

| | |
|---|---|
| **Operators and Special Characters** | |
| **+ ,- , *, .*, ^, .^, / , .\, ./, : , ( ), [ ], ., ..., ,,; , %, ', =** | |
| **Managing a Session** | |
| *clc* | Clears Command window |
| *clear* | Removes variables from memory |
| **Special Variables and Constants** | |
| *ans* | Most recent answer |
| *eps* | Accuracy of floating-point precision |
| *i,j* | The imaginary unit $\sqrt{-1}$ |
| *pi* | The number $\pi$ |
| **Input/Output Commands** | |
| *disp* | Displays contents of an array or string |

## Vector, Matrices and Arrays

| | |
|---|---|
| **Array Commands** | |
| *find* | Finds indices of nonzero elements.<br>`ind = find(X)`<br>`ind = find(X,k)`<br>`[row,col] = find(X)` |
| *length* | Computers number of elements.<br>`numberOfElements = length(array)` |
| *linspace* | Creates regularly spaced vector.<br>`y = linspace(a,b)`<br>`y = linspace(a,b,n)` |
| *logspace* | Creates log spaced vector.<br>`y = logspace(a,b)`<br>`y = logspace(a,b,n)` |
| *max* | Returns largest element.<br>`C = max(A)`<br>`[C,I] = max(A)` |
| *min* | Returns smallest element. |
| *reshape* | Change size<br>`B = reshape(A,m,n)` |
| *repmat* | Replicate and tile array<br>`B = repmat(A,m,n)` |
| *size* | Computes array size<br>`d = size(X)`<br>`[m,n] = size(X)` |
| *sort* | Sorts each column.<br>`B = sort(A)`<br>`B = sort(A,dim)`<br>`[B,IX] = sort(A)` |
| *sum* | Sums each column.<br>`B = sum(A)`<br>`B = sum(A,dim)` |
| *sub2ind* | Convert subscripts to linear indices<br>`ind = sub2ind(matrSize, rowSub, colSub)` |
| *ind2sub* | Subscripts from linear index<br>`[I,J] = ind2sub(siz,IND)` |
| *numel* | Number of elements in array or subscripted array expression<br>`n = numel(A)` |

| | |
|---|---|
| **Special Matrices** | |
| *eye* | Creates an identity matrix. |
| *ones* | Creates an array of ones. |
| *zeros* | Creates an array of zeros. |
| *diag* | Diagonal matrices |
| **Matrix Arithmetic** | |
| *cross* | Computes cross products.<br>`C = cross(A,B)`<br>`C = cross(A,B,dim)` |
| *dot* | Computes dot products.<br>`C = dot(A,B)`<br>`C = dot(A,B,dim)` |
| **Solving Linear Equations** | |
| *det* | Computes determinant of an array. |
| *inv* | Computes inverse of a matrix. |
| *pinv* | Computes pseudoinverse of a matrix. Solve linear equations in the least-squares sense. |
| *rank* | Computes rank of a matrix. |
| *trace* | Sum of diagonal elements |
| *norm* | Vector and matrix norms. |

## Plotting Commands

| | |
|---|---|
| **Basic xy Plotting Commands** | |
| *axis* | Sets axis limits.<br>`axis([xmin xmax ymin ymax])` |
| *grid* | Displays gridlines. |
| *plot* | Generates xy plot.<br>`plot(Y)`<br>`plot(X1,Y1,...,Xn,Yn)` |
| *title* | Puts text at top of plot. |
| *xlabel* | Adds text label to x-axis. |
| *ylabel* | Adds text label to y-axis. |
| *figure* | Opens a new figure window. |
| *Hold on/off* | Freezes/unfreezes current plot. |
| *text* | Places string in figure |
| **Specialized Plot Commands** | |
| *bar* | bar chart.<br>`bar(Y)`<br>`bar(x,Y)` |
| *polar* | polar plot.<br>`polar(theta,rho)` |
| *hist* | Create and plot histogram<br>`hist(data)`<br>`hist(data,nbins)` |

```
               hist(data,xcenters)
```

**Color Symbol Line**

| | | |
|---|---|---|
| *y* yellow | . point | - solid |
| *m* magenta | *o* circle | *:* dotted |
| *c* cyan | *x* x-mark | -. dash dotted |
| *r* red | + plus | -- dashed |
| *g* green | * star | |
| *b* blue | *d* diamond | |
| *w* white | *v* triangle (down) | |
| *k* black | ^ triangle (up) | |

**Three-Dimensional Plot**s

| | |
|---|---|
| *contour* | Creates contour plot |
| *mesh* | mesh surface plot |
| *plot3* | lines and points |
| *surf* | shaded mesh surface plot |
| *surfc* | surf with contour plot underneath |
| *meshgrid* | Creates rectangular grid |
| *zlabel* | Adds text label to z-axis |

## Programming

**Logical and Relation Operators**

== , ~=, <, <=, >, >=, &, |, ~, *xor*

**Flow Control**

| | |
|---|---|
| *break* | Terminates execution of a loop |
| *error* | Display error messages |

```
error('msgString')
```

*for*

```
for var = drange
        statements
end
```

*if*

```
if expression
    statements
elseif expression
    statements
else
    statements
end
```

| | |
|---|---|
| *return* | Return to the invoking function |
| *switch* | comparing with case expressions |

```
switch switch_expression
    case case_expression
        statements
    case case_expression
        statements
      :
    otherwise
        statements
end
```

| | |
|---|---|
| *warning* | Display a warning message. |

*while*

```
while expression
    statements
end
```

**Logical Functions**

| | |
|---|---|
| *any* | True if any elements are nonzero |
| *all* | True if all elements are nonzero |
| *find* | Finds indices of nonzero elements |
| *logical* | Convert numeric values to logical |

**M-Files**

| | |
|---|---|
| *function* | Creates a function M-file. |
| *global* | Define global variables |

**Timing**

| | |
|---|---|
| *cputime* | CPU time in seconds. |
| *clock* | Current date and time |
| *tic, toc* | Start, stop a stopwatch timer. |

## Mathematical Functions

**Exponential and Logarithms**

*Exp*, *log,* ln, *log10, sqrt*

**Trigonometric**

*cos, cot, csc, sec, sin, tan*

**Inverse trig**

*acos* , *acot, acsc*, *asec, asin, atan*

**Complex Functions**

| | |
|---|---|
| *abs* | Absolute value; \|x\|. |
| *angle* | Angle of a complex number x. |
| *conj* | Complex conjugate of x. |
| *imag* | Imaginary part |
| *real* | Real part |

**Statistical Functions**

| | |
|---|---|
| *mean* | Average<br>`M = mean(A)`<br>`M = mean(A,dim)` |
| *median* | median. |
| *std* | standard deviation |
| *var* | variance |

**Random Numbers**

| | |
|---|---|
| *rand* | uniformly distributed random numbers between 0 and 1.<br>`r = rand(n)`<br>`r = rand(m,n)` |
| *randn* | normally distributed random numbers<br>`r = randn(n)`<br>`r = randn(m,n)` |

**Numeric Functions**

| | |
|---|---|
| *ceil* | Round up |
| *floor* | Round down |
| *round* | Round to nearest integer |
| *sign* | Signum |
| *rem* | Remainder after division |
| *mod* | Modulus after division |

## Numerical Methods

**Polynomial**

| | |
|---|---|
| *eig* | eigenvalues of a matrix.<br>`d = eig(A)`<br>`[V,D] = eig(A)` |

| | |
|---|---|
| *poly* | Computes polynomial from roots |
| *roots* | Computes polynomial roots.<br>`r = roots(c)` |

**Root Finding and Minimization**

| | |
|---|---|
| *fminbnd* | Find minimum of single-variable function on fixed interval<br>`x = fminbnd(fun,x1,x2)` |
| *fminsearch* | Find minimum of unconstrained multivariable<br>`x = fminsearch(fun,x0)` |
| *fzero* | Finds zero of single-variable function.<br>`x = fzero(fun,x0)` |

**Numerical Integration**

| | |
|---|---|
| *quad* | Numerical integration with adaptive Simpson's rule.<br>`q = quad(fun,a,b)` |
| *trapz* | Numerical integration with the trapezoidal rule.<br>`Z = trapz(Y)`<br>`Z = trapz(Y,dim)` |

**Numerical Differentiation**

| | |
|---|---|
| *diff* | the difference between adjacent elements<br>`Y = diff(X)`<br>`Y = diff(X,n)`<br>`Y = diff(X,n,dim)` |

# List of muPad commands

## General Purpose

| | |
|---|---|
| := | Assign variables |
| ; | Statement sequences |
| delete | Delete the value of an identifier<br>delete $x_1$, $x_2$, … |
| reset | Re-initialize a session |
| /% %/ | comment |

**Special Values**

| | |
|---|---|
| TRUE | Boolean constant TRUE |
| FALSE | Boolean constant FALSE |
| UNKNOWN | Boolean constant UNKNOWN |
| infinity | Real positive infinity |

**Common Operations**

| | |
|---|---|
| .. | Range operator |
| nops | Number of operands |
| op | Operands of an object<br>`op(object, [i1, i2, …])` |
| domtype | Data type of an object |
| prog::exprtree | Visualize an expression as tree |
| Print | Print command |

**Operations on Lists, sets, Stering, etc …**

| | |
|---|---|
| `{}` | Define a set.<br>`Set:={…}` |
| "" | Define a string.<br>`S1:="…"` |
| . | Concatenate |
| `$` | Such that.<br>`set:={f(i) $ i=a..b}` |
| map | Apply function to set/sequence/list.<br>`Map(set,f)` |
| [] | Define a list<br>List:=[a,b,…] |
| sort | Sort a list.<br>sort(list) |
| select | Select from a list/set/sequence<br>`Select(list,boolFunc)` |

## Programming Basics

**Flow control**

`switch` Switch statement

```
case x
  of match1 do
    statements1
  of match2 do
    statements2
  ...
  otherwise
    otherstatements
end_case

case x
  of match1 do
    statements1
  of match2 do
    statements2
  ...
end_case
```

`for` For loop

```
for i from start to stop do
  body
end_for

for i from start to stop
step stepwidth do
  body
end_for

_for(i, start, stop,
stepwidth, body)
for i from start downto stop
do
  body
end_for
```

| | |
|---|---|
| | for i from start downto stop step stepwidth do<br>body<br>end_for |
| if | If-statement (conditional branch in a program)<br>if $condition_1$<br>then $casetrue_1$<br>elif $condition_2$ then $casetrue_2$<br>elif $condition_3$ then $casetrue_3$<br>...<br>else casefalse<br>end_if |
| while | "while" loop<br>while condition do<br>body<br>end_while |
| return | Exit a procedure |
| proc | Define a procedure<br>proc($x_1$,$x_2$,...)<br>begin<br>body<br>end_proc |

# Mathematics

| | |
|---|---|
| **->** | Define a function/procedure inline<br>( $x_1$, $x_2$, … ) -> body |
| **-->** | Turn an expression into a procedure.<br>f:=x^2: g:=x-->f |
| @ | Compose functions<br>f @ g @ ... |

### Symbolic Solvers

| | |
|---|---|
| linsolve | Solve a system of linear equations<br>linsolve(eqs, vars, options) |
| RootOf | Set of roots of a polynomial<br>RootOf(f, x) |
| solve | Solve equations and inequalities<br>solve(eq, x, options)<br>solve(eq, x = a .. b, options) |

### Numeric Solvers

| | |
|---|---|
| numeric::fsolve | Search for a numerical root of a system of equations<br>numeric::fsolve(eq, x, options)<br>numeric::fsolve(eq, x = a, options)<br>numeric::fsolve(eq, x = a .. b, options) |
| numeric::leastSquares | Least squares solution of linear equations<br>numeric::leastSquares(A, B, <mode>, <method>, options) |
| numeric::linsolve | Solve a system of linear equations<br>numeric::linsolve(eqs, <vars>, options) |
| numeric::solve | Numerical solution of equations (the float attribute of solve). Find all roots.<br>numeric::solve(eqs, <vars>, options) |

### Properties and Assumptions

| | |
|---|---|
| is | Check a mathematical property of an expression<br>is(cond)<br>is(ex, set) |

### Simplification

| | |
|---|---|
| factorout | Factor out a given expression<br>factorout(x, f, <list>) |
| simplify | Simplify an expression<br>Simplify(f) |
| expand | Expand an expression<br>expand(f, options) |
| subs | Substitute into an object<br>subs(f, old = new) |

### Calculus

| | |
|---|---|
| D | Differential operator for functions<br>D(f) |
| diff | Differentiate an expression or a polynomial<br>diff(f)<br>diff(f, x)<br>diff(f, x1, x2, …)<br>diff(f, x $ 3) |
| int | Definite and indefinite integrals<br>int(f, x)<br>int(f, x = a .. b, options) |
| numeric::quadrature | Numerical integration ( Quadrature )<br>numeric::quadrature(f(x), x = a .. b) |
| taylor | Compute a Taylor series expansion<br>taylor(f, x = x0, <order>) |
| sum | Definite and indefinite summation<br>sum(f, i)<br>sum(f, i = a .. b) |
| numeric::sum | Numerical approximation of sums (the Float attribute of Sum )<br>numeric::sum(f(x), x = a .. b)<br>numeric::sum(f(x), x in {$x_1$, $x_2$, …}) |
| limit | Compute a limit<br>limit(f, x = $x_0$, <Left \| Right \| Real>, <Intervals>) |

## Linear Algebra

| | |
|---|---|
| array | Create an array<br>array($m_1$ .. $n_1$, <$m_2$ .. $n_2$, …>)<br>array($m_1$ .. $n_1$, <$m_2$ .. $n_2$, …>, index$_1$ = entry$_1$, index$_2$ = entry$_2$, …)<br>array($m_1$ .. $n_1$, <$m_2$ .. $n_2$, …>, List)<br>array(<$m_1$ .. $n_1$, $m_2$ .. $n_2$, …>, ListOfLists) |
| matrix | Create a matrix or a vector<br>matrix(Array)<br>matrix(List)<br>matrix(ListOfRows)<br>matrix(m, n)<br>matrix(m, n, Array)<br>matrix(m, n, List)<br>matrix(m, n, ListOfRows)<br>matrix(m, n, [($i_1$, $j_1$) = value$_1$, ($i_2$, $j_2$) = value$_2$, …])<br>matrix(m, n, f)<br>matrix(m, n, List, Diagonal) |
| Dom::<ring> | Constructor<br>Constructor:=Dom::IntegerMod(7) |
| linalg::randomMatrix | Generate a random matrix<br>linalg::randomMatrix(m, n, <R>, <bound>) |

## Matrix Operations and Transformations

| | |
|---|---|
| linalg::addCol | Add a colums |
| linalg::addRow | Add a row |
| linalg::col | Extract columns of a matrix |
| linalg::delCol | Delete matrix columns |
| linalg::delRow | Delete matrix rows |
| linalg::row | Extract rows of a matrix |
| inverse | Inverse of a matrix |
| transpose | Transpose of a matrix |
| linalg::pseudoInverse | Moore-Penrose inverse of a matrix |
| numeric::inverse | Numerical inverse of a matrix |
| norm | norm of a matrix or vector |
| linalg::normalize | Normalize a vector |
| det | Determinant |
| numeric::det | Numerical determinant |
| linalg::angle | Angle between two vectors |
| linalg::ncols | Number of columns |
| linalg::nrows | Number of rows |
| linalg::sqrtMatrix | Square root of a matrix |
| linalg::tr | Trace |
| linalg::matdim | Dimension of a matrix |
| linalg::nullspace | Basis for the null space |
| linalg::orthog | Orthogonalization of vectors |
| linalg::rank | Rank of a matrix |
| numeric::rank | Numerical estimate of the rank of a matrix |
| linalg::eigenvalues | Eigenvalues |
| linalg::eigenvectors | Eigenvectors |
| numeric::eigenvalues | Numerical eigenvalues |
| numeric::eigenvectors | Numerical eigenvalues |

## Polynomial Algebra

| | |
|---|---|
| poly | Create a polynomial<br>poly(f, <[$x_1$, $x_2$, …]>, <ring>) |
| divide | Divide polynomials<br>divide(p, q |
| coeff | Coefficients of a polynomial<br>coeff(p, <x>, n) |
| degree | Degree of a polynomial<br>degree(p)<br>degree(p, x) |
| numeric::polyroots | Numerical roots of a univariate polynomial<br>numeric::polyroots(eqs) |
| numeric::realroot | Numerical search for a real root of a real univariate function |

| | |
|---|---|
| poly | Create a polynomial<br>poly(f, <[$x_1$, $x_2$, …]>, <ring>) |
| divide | Divide polynomials<br>divide(p, q |
| coeff | Coefficients of a polynomial<br>coeff(p, <x>, n) |
| degree | Degree of a polynomial<br>degree(p)<br>degree(p, x) |
| numeric::polyroots | Numerical roots of a univariate polynomial<br>numeric::polyroots(eqs) |
| numeric::realroot | Numerical search for a real root of a real univariate function |

## Mathematical functions

### Complex Numbers

`abs, arg, Re, Im`

### Exponents and Logarithms

`exp, ln, log, log10, log2, ^, sqrt`

### Trigonometric Functions

`arcsin, arccos, arctan, arccsc, arcsec, arccot, sin, cos, tan, csc, sec, cot`

### Numbers and Precision

| | |
|---|---|
| `float` | Convert to a floating-point number |

### Operations on Numbers

| | |
|---|---|
| `ceil` | Rounding up to the next integer |
| `floor` | Rounding down to the next integer |
| `conjugate` | Complex conjugation |
| `max` | Maximum of numbers |
| `min` | Minimum of numbers |
| `round` | Rounding to the nearest integer |

## Random Numbers

| | |
|---|---|
| `frandom` | Generate random floating-point numbers<br>`frandom()`<br>`frandom(seed)` |
| `random` | Generate random integer numbers<br>`random(`n$_1$ `..` n$_2$`)`<br>`random(n)`<br>`die := random(1..6):`<br>`die() $ i = 1..20` |
| `stats::normalRandom` | Generate a random number generator for normal deviates<br>`stats::normalRandom(m, v, <Seed = s>)` |
| `stats::uniformRandom` | Generate a random number generator for uniformly continuous deviates<br>`stats::uniformRandom(a, b, <Seed = s>)` |

## Discrete Mathematics

| | |
|---|---|
| `gcd` | Greatest common divisor of polynomials<br>`gcd(p, q)` |
| `fact, !` | Factorial function |
| `div` | Integer part of a quotient<br>`m div n` |
| `mod` | Modulo operator<br>`x mod m` |
| `bool` | Boolean evaluation<br>`bool(b)` |
| isprime | |

## Set Operations

| | |
|---|---|
| `contains` | Test if an entry exists in a container<br>`contains(s, object)` |
| `in` | Membership<br>`x in set` |
| `intersect` | Intersection of sets and/or intervals<br>set$_1$ `intersect` set$_2$ |
| `minus` | Difference of sets and/or intervals<br>set$_1$ `minus` set$_2$ |
| `union` | Union of sets and/or intervals<br>set$_1$ `union` set$_2$ |

## Graphics

| | |
|---|---|
| `plot` | Display graphical objects on the screen<br>`plot(object)`<br>`plot(sin(x));`<br>`plot(sin(x)/x,x=-1..1);`<br>`plot([2*cos(t),sin(t)],t=0..2*PI)`<br>Parametric representation<br>`plot([2*cos(t),sin(t)],t=0..2*PI);` |
| `plot::PointList2d` | Plot a list of points.<br>`plot(plot::PointList2d([[1,1],[2,2],[3,3]]));` |
| `plot::Polygon2d` | Plot a list of points connected by a line.<br>`plot(plot::Polygon2d([[1,1],[2,4],[3,3]]));` |
| `plot::PointList2d` | 3D version of `plot::PointList2d`<br>`plot(plot::PointList3d([[1,1,1], [1,2,2], [1,3,2]`]) |
| `plot::Polygon2d` | 3D version of `plot::Polygon2d`<br>`plot(plot::Polygon3d([[1,1,1],[2,4,2],[3,3,1]]));` |
| `plotfunc3d`<br>`plot(…,#3D)` | Plot a 3D function<br>`plotfunc3d(1/(x^2 + y^2), x = -1..1, y = -1..1):` |
| `plot::Implicit2d`<br>`plot::Implicit3d` | Plot implicit functions<br>`plot(plot::Implicit2d(x^3+x+2=y^2,x=-5..5,y=-5..5));`<br>`plot(plot::Implicit3d(x^2+y^2+z^2=1,x=-2..2,y=-2..2,z=-2..2));` |
| `plot::Polar` | Polar representation<br>`plot(plot::Polar([r(t),t],t=0..2*PI))` |