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 π

Input/Output Commands

disp	Displays contents of an array or string

Vector, Matrices and Arrays

Array Commands

find	Finds indices of nonzero elements.
	`ind = find(X)`
	`ind = find(X,k)`
	`[row,col] = find(X)`

length	Computers number of elements. `numberOfElements = length(array)`

linspace	Creates regularly spaced vector.
	`y = linspace(a,b)`
	`y = linspace(a,b,n)`

logspace	Creates log spaced vector.
	`y = logspace(a,b)`
	`y = logspace(a,b,n)`

max	Returns largest element.
	`C = max(A)`
	`[C,I] = max(A)`

min	Returns smallest element.
reshape	Change size `B = reshape(A,m,n)`
repmat	Replicate and tile array `B = repmat(A,m,n)`

size	Computes array size
	`d = size(X)`
	`[m,n] = size(X)`

sort	Sorts each column.
	`B = sort(A)`
	`B = sort(A,dim)`
	`[B,IX] = sort(A)`

sum	Sums each column.
	`B = sum(A)`
	`B = sum(A,dim)`

sub2ind	Convert subscripts to linear indices `ind = sub2ind(matrSize, rowSub, colSub)`
ind2sub	Subscripts from linear index `[I,J] = ind2sub(siz,IND)`
numel	Number of elements in array or subscripted array expression `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.
	`C = cross(A,B)`
	`C = cross(A,B,dim)`

dot	Computes dot products.
	`C = dot(A,B)`
	`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. `axis([xmin xmax ymin ymax])`
grid	Displays gridlines.

plot	Generates xy plot.
	`plot(Y)`
	`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.
	`bar(Y)`
	`bar(x,Y)`

polar	polar plot. `polar(theta,rho)`

hist	Create and plot histogram
	`hist(data)`
	`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 Plots

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

```
M = mean(A)
M = mean(A,dim)
```

median median.

std standard deviation

var variance

Random Numbers

rand uniformly distributed random numbers between 0 and 1.

```
r = rand(n)
r = rand(m,n)
```

randn normally distributed random numbers

```
r = randn(n)
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.

```
d = eig(A)
[V,D] = eig(A)
```

poly	Computes polynomial from roots
roots	Computes polynomial roots. `r = roots(c)`

Root Finding and Minimization

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

Numerical Integration

quad	Numerical integration with adaptive Simpson's rule. `q = quad(fun,a,b)`

trapz	Numerical integration with the trapezoidal rule.
	`Z = trapz(Y)`
	`Z = trapz(Y,dim)`

Numerical Differentiation

diff	the difference between adjacent elements
	`Y = diff(X)`
	`Y = diff(X,n)`
	`Y = diff(X,n,dim)`

List of muPad commands

General Purpose

:=	Assign variables
;	Statement sequences
delete	Delete the value of an identifier 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 `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. `Set:={…}`
""	Define a string. `S1:="…"`
.	Concatenate
$	Such that. `set:={f(i) $ i=a..b}`
map	Apply function to set/sequence/list. `Map(set,f)`
[]	Define a list List:=[a,b,…]
`sort`	Sort a list. sort(`list`)
select	Select from a list/set/sequence `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 body end_for

if	If-statement (conditional branch in a program)
	if condition$_1$
	then casetrue$_1$
	  elif condition$_2$ then
	casetrue$_2$
	  elif condition$_3$ then
	casetrue$_3$
	  ...
	  else casefalse
	end_if

while	"while" loop while condition do body end_while
return	Exit a procedure
proc	Define a procedure proc(x_1,x_2,...) begin body end_proc

Mathematics

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

Symbolic Solvers

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

Numeric Solvers

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

Properties and Assumptions

is	Check a mathematical property of an expression is(cond) is(ex, set)

Simplification

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

Calculus

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

Linear Algebra

`array`	Create an array
	`array(`m_1 `..` n_1`, <`m_2 `..` n_2`, …>)`
	`array(`m_1 `..` n_1`, <`m_2 `..` n_2`, …>, `$index_1$ `=` $entry_1$`,` $index_2$ `=` $entry_2$`, …)`
	`array(`m_1 `..` n_1`, <`m_2 `..` n_2`, …>, List)`
	`array(<`m_1 `..` n_1`,` m_2 `..` n_2`, …>, ListOfLists)`

`matrix`	Create a matrix or a vector
	`matrix(Array)`
	`matrix(List)`
	`matrix(ListOfRows)`
	`matrix(m, n)`
	`matrix(m, n, Array)`
	`matrix(m, n, List)`
	`matrix(m, n, ListOfRows)`
	`matrix(m, n, [(`i_1`,` j_1`) =` $value_1$`, (`i_2`,` j_2`) =` $value_2$`, …])`
	`matrix(m, n, f)`
	`matrix(m, n, List, Diagonal)`

Dom:: <ring>	Constructor Constructor:=`Dom::IntegerMod(7)`
`linalg::random Matrix`	Generate a random matrix `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 `poly(f, <[`x_1`,` x_2`, …]>, <ring>)`
`divide`	Divide polynomials `divide(p, q`
`coeff`	Coefficients of a polynomial `coeff(p, <x>, n)`

`degree`	Degree of a polynomial
	`degree(p)`
	`degree(p, x)`

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

`degree`	Degree of a polynomial
	`degree(p)`
	`degree(p, x)`

`numeric::polyroots`	Numerical roots of a univariate polynomial `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
	`frandom()`
	`frandom(seed)`

`random`	Generate random integer numbers
	`random(`n_1 `..` n_2`)`
	`random(n)`

`stats::normal Random`	Generate a random number generator for normal deviates `stats::normalRandom(m, v, <Seed = s>)`
`stats::uniform Random`	Generate a random number generator for uniformly continuous deviates `stats::uniformRandom(a, b, <Seed = s>)`

Discrete Mathematics

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

Set Operations

`contains`	Test if an entry exists in a container `contains(s, object)`
`in`	Membership `x in set`
`intersect`	Intersection of sets and/or intervals set_1 `intersect` set_2
`minus`	Difference of sets and/or intervals set_1 `minus` set_2
`union`	Union of sets and/or intervals set_1 `union` set_2

Graphics

`plot`	Display graphical objects on the screen
	`plot(object)`
	`plot(sin(x));`
	`plot(sin(x)/x,x=-1..1);`
	`plot([2*cos(t),sin(t)],t=0..2*PI)`
	Parametric representation
	`plot([2*cos(t),sin(t)],t=0..2*PI);`

`plot::PointList2d`	Plot a list of points. `plot(plot::PointList2d([[1,1],[2,2],[3,3]]));`
`plot::Polygon2d`	Plot a list of points connected by a line. `plot(plot::Polygon2d([[1,1],[2,4],[3,3]]));`
`plot::PointList2d`	3D version of `plot::PointList2d` `plot(plot::PointList3d([[1,1,1], [1,2,2], [1,3,2]])`
`plot::Polygon2d`	3D version of `plot::Polygon2d` `plot(plot::Polygon3d([[1,1,1],[2,4,2],[3,3,1]]));`
`plotfunc3d` `plot(…,#3D)`	Plot a 3D function `plotfunc3d(1/(x^2 + y^2), x = -1..1, y = -1..1):`
`plot::Implicit2d`	Plot implicit functions `plot(plot::Implicit2d(x^3+x+2=y^2,x=-5..5,y=-5..5));`
`plot::Implicit3d`	`plot(plot::Implicit3d(x^2+y^2+z^2=1,x=-2..2,y=-2..2,z=-2..2));`
`plot::Polar`	Polar representation `plot(plot::Polar([r(t),t],t=0..2*PI))`