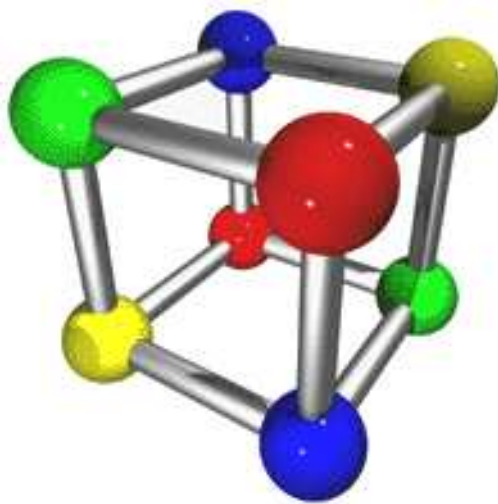


MuPAD 2.5

Quick Reference

W. Oevel and J. Gerhard



MuPAD
The Open Computer Algebra System

MuPAD 2.5

Quick Reference

This paper gives an overview of the data structures, constants, functions and libraries in MuPAD version 2.5. More details are available in the tutorial

W. OEVEL, F. POSTEL, S. WEHMEIER, AND J. GERHARD.

The MuPAD Tutorial. Springer, 2000.

An updated version of this book is available online.

Contents

1	Information and Help	2
2	Environment Variables	2
3	Predefined Data Types (Domains)	2
4	Generation of Data Structures	3
5	Operators	4
6	Arithmetical Functions	5
7	Symbols and Constants	6
8	Special Mathematical Functions	6
9	Functions for Polynomials	8
10	Handling of MuPAD-Objects	9
11	Manipulation of Strings	10
12	Functions for Input, Output, and Graphics	10
13	Controlling Evaluation	11
14	Functions in Procedure Definitions	11
15	Mathematical Functions and Algorithms	12
16	Solving Equations	13
17	Libraries and Modules	14
18	Debugging and Timing	15
19	Technical Issues	15

1 Information and Help

`help, ?` : calling help pages
`info` : print short information
`setuserinfo` : information on algorithms

2 Environment Variables

Environment variables are global variables which control the operation of algorithms implemented in MuPAD-procedures. The default values can be changed by the user.

variable	default value	meaning
DIGITS	10	significant digits in floating-point operations
HISTORY	20	number of results available via <code>last</code>
LEVEL	100	evaluation depth
LIBPATH		path name of the MuPAD program and libraries
MAXDEPTH	500	maximal recursion depth
MAXLEVEL	100	maximal evaluation depth
ORDER	6	default number of terms in series expansions
PRETTYPRINT	TRUE	2-dimensional output?
PACKAGEPATH		pathname for packages to be read via <code>package</code>
READPATH		pathname for files to be read via <code>read</code>
SEED	1	seed for generating random numbers via <code>random</code>
TEXTWIDTH	75	text width for output
WRITEPATH		pathname for files to be saved via <code>write</code>

3 Predefined Data Types (Domains)

Data types of the MuPAD kernel:

`DOM_ARRAY` : arrays
`DOM_BOOL` : the logical constants `TRUE`, `FALSE` and `UNKNOWN`
`DOM_COMPLEX` : complex numbers
`DOM_DOMAIN` : data structures
`DOM_EXPR` : expressions
`DOM_FAIL` : the object `FAIL`
`DOM_FLOAT` : real floating-point numbers
`DOM_FUNC_ENV` : function environments
`DOM_IDENT` : identifiers
`DOM_INT` : integer numbers
`DOM_INTERVAL` : floating point intervals
`DOM_LIST` : lists
`DOM_NIL` : the object `NIL`
`DOM_NULL` : the void object `null()`
`DOM_POINT` : points (as graphical primitives)
`DOM_POLY` : polynomials
`DOM_POLYGON` : polygons (as graphical primitives)

```

DOM_PROC      : procedures
DOM_RAT       : rational numbers
DOM_SET       : sets
DOM_STRING    : strings
DOM_TABLE     : tables
DOM_VAR       : local variables in procedures

```

Data types of the standard library defined in the MuPAD–language:

```

Factored      : objects in factored form
0             : order term of series expansions
ode           : ordinary differential equations
piecewise     : conditionally defined objects
rec           : recurrence equations
rectform      : cartesian representation of complex numbers
Series::gseries : generalized series expansions
Series::Puisseux : series expansions

```

Many more data structures and their constructors are available in the library `Dom`. Call `?Dom` or `info(Dom)` for more information.

4 Generation of Data Structures

```

array         : generates an array (DOM_ARRAY)
asympt        : asymptotic expansion (generates Series::gseries)
factor, ifactor : factor into irreducible elements (generates Factored)
funcenv       : defines a function environment (DOM_FUNC_ENV)
matrix        : generates a matrix of domain type Dom::Matrix()
new           : generates domain elements
newDomain     : generates domains (DOM_DOMAIN)
null          : generates the "void object" (DOM_NULL)
0             : generates the order term of series expansions
ode           : generates an ordinary differential equation
piecewise     : generates a conditionally defined object
point         : generates the data structure DOM_POINT for plotting points
poly          : generates a polynomial (DOM_POLY)
polygon       : generates the data structure DOM_POLYGON for
                plotting polygons
rec           : generates a recurrence equation
rectform      : cartesian representation of complex numbers
                (generates rectform)
series        : generalized series expansion (generates Series::Puisseux
                or Series::gseries)
slot          : defines or reads an attribute of a domain or
                function environment
sparsematrix  : generates a matrix of domain type Dom::SparseMatrix()
taylor        : Taylor expansion (generates Series::Puisseux)
table         : generates a table (DOM_TABLE)

```

Many more constructors of special data types are defined in the library `Dom`, e.g.:

```
Dom::IntegerMod(p) : generates integer numbers modulo p
Dom::Matrix()      : generates matrices
Dom::SparseMatrix() : generates sparse matrices
...                : ...
```

5 Operators

The following operators can be used to call system functions in a more intuitive way, e.g. `a+b` instead of `_plus(a,b)`, `x<1` instead of `_less(x,1)`, etc.:

operator	priority	system function	meaning
<code>::</code>	2000	<code>slot</code>	method access
<code>'</code>	1900	<code>D</code>	differential operator
<code>[]</code>	1800	<code>_index</code>	index operator
<code>.</code>	1700	<code>_concat</code>	concatenation
<code>@@</code>	1600	<code>_fnest</code>	iterated composition
<code>@</code>	1500	<code>_fconcat</code>	composition of functions
<code>!</code>	1300	<code>fact</code>	factorial function
<code>^</code>	1200	<code>_power</code>	power
<code>*</code>	1100	<code>_mult</code>	multiplication
<code>/</code>	1100	<code>_divide</code>	division
<code>-</code>	1050	<code>_negate</code>	Negation
<code>+</code>	1000	<code>_plus</code>	addition
<code>-</code>	1000	<code>_subtract</code>	subtraction
<code>div</code>	900	<code>_div</code>	quotient "modulo"
<code>mod</code>	900	<code>_mod</code>	remainder "modulo"
<code>...</code>	850	<code>hull</code>	floating point intervals
<code>intersect</code>	800	<code>_intersect</code>	intersection of sets
<code>minus</code>	700	<code>_minus</code>	difference of sets
<code>union</code>	600	<code>_union</code>	union of sets
<code>..</code>	500	<code>_range</code>	range
<code>=</code>	400	<code>_equal</code>	equation
<code><></code>	400	<code>_unequal</code>	inequality
<code>< and ></code>	400	<code>_less</code>	comparisons
<code><= and >=</code>	400	<code>_leequal</code>	comparisons
<code>in</code>	400	<code>_in</code>	containment
<code>\$</code>	300	<code>_seqgen</code> and <code>_seqin</code>	sequence generator
<code>not</code>	300	<code>_not</code>	logical negation
<code>and</code>	200	<code>_and</code>	logical "and"
<code>xor</code>	150	<code>_xor</code>	logical "exclusive or"
<code>or</code>	100	<code>_or</code>	logical "or"
<code>=></code>	75	<code>_implies</code>	logical "implies"

<code><=></code>		50		<code>_equiv</code>		logical equivalence
<code>,</code>		20		<code>_exprseq</code>		separator between elements of sequences
<code>; and :</code>		10		<code>_stmtseq</code>		command separator

Users can define their own operators via `operator`.

6 Arithmetical Functions

The following functions can be used via operators or keywords, e.g. `a+b` instead of `_plus(a,b)`, `-a` instead of `_negate(a)`, etc.:

```

_and      : logical "and"
_assign   : assignment function
_break    : interruption of loops etc.
_case     : branching
_concat   : concatenation
_delete   : delete values or elements
_div      : quotient "modulo"
_divide   : division
_equal    : equation
_equiv    : logical equivalence
_exprseq  : sequence generator
_fconcat  : concatenation of functions
_fnest    : iteration of functions
_for      : loop (upwards)
_for_down : loop (downwards)
_for_in   : loop (over the operands of an expression)
hull      : generate a floating point interval
_if       : branch
_implies  : logical "implies"
_index    : indexed expression
_intersect : intersection of sets
_invert   : inversion
_lazy_and : lazily evaluated logical "and"
_lazy_or  : lazily evaluated logical "or"
_leequal  : inequality ≤
_less     : inequality <
_minus    : difference of sets
_mod      : remainder "modulo"
_mult     : multiplication
_negate   : multiplication with -1
_next     : jump in a loop
_not      : logical negation
_or       : logical "or"
_plus     : addition
_power    : power
_procdef  : procedure definition

```

```

_quit      : quitting a MuPAD-session
_range     : range
_repeat    : loop
_seqgen    : sequence generator
_seqin     : sequence generator
_stmtseq   : sequence of commands
_subtract  : difference
_unequal   : inequality <>
_union     : union of sets
_while     : loop
_xor       : logical "exclusive or"

```

7 Symbols and Constants

```

C_         : the set  $\mathbb{C}$  of complex numbers
CATALAN    : Catalan constant: float(CATALAN)=0.9159655941..
complexInfinity : the infinite point of the complex plane
E          : Euler number  $\exp(1) = 2.718281828..$ 
EULER      : Euler constant  $\gamma = 0.5772156649..$ 
FAIL       : failure object (of domain type DOM_FAIL)
FALSE      : Boolean constant
I          : imaginary unit  $\sqrt{-1}$ 
infinity   : infinity
NIL        : null object, only object of domain type DOM_NIL
PI         :  $\pi = 3.141592653..$ 
Q_         : the set  $\mathbb{Q}$  of rational numbers
R_         : the set  $\mathbb{R}$  of real numbers
RD_INF, RD_NINF : rounded  $\pm$  infinity for interval arithmetic
TRUE       : Boolean constant
undefined  : undefined value
universe   : set-theoretic universe of all objects
UNKNOWN    : Boolean constant
Z_         : the set  $\mathbb{Z}$  of integers

```

8 Special Mathematical Functions

The following mathematical functions are implemented in MuPAD version 2.5. They are handled by system functions such as `expand`, `float`, `simplify`, etc.:

```

abs        : absolute value
arccos     : inverse function of cos
arccosh    : inverse function of cosh
arccot     : inverse function of cot
arccoth    : inverse function of coth
arccsc     : inverse function of csc
arccsch    : inverse function of csch
arcsec     : inverse function of sec

```

<code>arcsech</code>	: inverse function of <code>sech</code>
<code>arcsin</code>	: inverse function of <code>sin</code>
<code>arcsinh</code>	: inverse function of <code>sinh</code>
<code>arctan</code>	: inverse function of <code>tan</code>
<code>arctanh</code>	: inverse function of <code>tanh</code>
<code>arg</code>	: polar angle of a complex number
<code>bernoulli</code>	: Bernoulli numbers and polynomials
<code>besselI</code>	: modified Bessel function of the first kind
<code>besselJ</code>	: Bessel function of the first kind
<code>besselK</code>	: modified Bessel function of the second kind
<code>besselY</code>	: Bessel function of the second kind
<code>beta</code>	: β -function
<code>binomial</code>	: binomial expression $\binom{m}{n}$
<code>ceil</code>	: smallest integer $\geq x$
<code>Ci</code>	: $\gamma + \ln(x) + \int_0^x (\cos(t) - 1)/t dt$
<code>cos</code>	: cosine function
<code>cosh</code>	: hyperbolic cosine function
<code>cot</code>	: cotangent function
<code>coth</code>	: hyperbolic cotangent function
<code>csc</code>	: $1/\sin(x)$
<code>csch</code>	: $1/\sinh(x)$
<code>dilog</code>	: dilogarithm function
<code>dirac</code>	: Dirac's δ -function
<code>Ei</code>	: $\int_x^\infty e^{-t}/t dt$
<code>erf</code>	: $(2/\sqrt{\pi}) \int_0^x e^{-t^2} dt$
<code>erfc</code>	: $1 - \operatorname{erf}(x)$
<code>exp</code>	: exponential function
<code>fact</code>	: factorial function
<code>floor</code>	: Gaussian brackets: greatest integer $\leq x$
<code>frac</code>	: fractional part of a number
<code>gamma</code>	: Γ -function
<code>heaviside</code>	: Heaviside function
<code>hypergeom</code>	: hypergeometric function
<code>id</code>	: identity map $x \rightarrow x$
<code>igamma</code>	: incomplete Γ -function
<code>Im</code>	: imaginary part
<code>lambertV</code>	: lower real branch of the Lambert function
<code>lambertW</code>	: upper real branch of the Lambert function
<code>ln</code>	: natural logarithm
<code>log</code>	: logarithm to an arbitrary base
<code>polylog</code>	: polylogarithm function
<code>psi</code>	: polygamma function
<code>Re</code>	: real part
<code>round</code>	: rounding to next number
<code>sec</code>	: $1/\cos(x)$
<code>sech</code>	: $1/\cosh(x)$
<code>Si</code>	: $\int_0^x \sin(t)/t dt$

sign : (complex) sign
signIm : sign of the imaginary part
sin : sine function
sinh : hyperbolic sine function
sqrt : square root function
tan : tangent function
tanh : hyperbolic tangent function
trunc : truncates a number
zeta : Riemann ζ -function

9 Functions for Polynomials

The following functions operate on polynomials of domain type `DOM_POLY` created by `poly`. Some also operate on polynomial expressions of domain type `DOM_EXPR`:

coeff : coefficients
collect : collect coefficients
content : greatest common divisor of all coefficients
degree : degree of a polynomial
degreevec : exponent(s) of the leading term
diff, D : differentiation
divide : division with remainder
evalp : evaluation at a point
expr : conversion to an expression
factor : factorization
gcd : greatest common divisor
gcdex : extended Euclidean algorithm
genpoly : generates a polynomial from a b -adic expansion
ground : constant coefficient of a polynomial
icontent : greatest common divisor of all coefficients
irreducible : test for irreducibility
iszero : test for 0-polynomial
lcm : least common multiple
lcoeff : leading coefficient
ldegree : lowest degree in a polynomial
lmonomial : leading monomial
lterm : leading term
mapcoeffs : applying a function to the coefficients
multcoeffs : multiplication with a scalar
norm : norm
nterms : number of terms
nthcoeff : n -th coefficient
nthmonomial : n -th monomial
nthterm : n -th term
pdivide : pseudo division
poly : generator of a polynomial
poly2list : conversion to a list

`tcoeff` : lowest coefficient

More functions operating on polynomials are available in the library `polylib`.

10 Handling of MuPAD-Objects

`alias` : definition of abbreviations
`anames` : list all identifiers which have a value
`append` : append to a list
`assign` : assignment function
`assignElements` : assignment function for arrays, lists and tables
`assume` : assumptions about properties of identifiers
`bool` : boolean evaluation to `TRUE` or `FALSE`
`coerce` : type conversion
`collect` : collect coefficients of polynomial expressions
`combine` : combine subexpressions
`contains` : test for elements in lists, sets and tables
`delete` : deleting the value of an identifier
`denom` : denominator of a rational expression
`domain` : defining a new domain
`domtype` : domain type
`expand` : expansion of expressions
`export` : loading libraries
`expose` : output of the operands of function environments and domains

`expr` : conversion of various data structures to expressions, arrays, etc.

`extnops` : number of operands of a domain element
`extop` : the operands of a domain element
`extsubop` : substitution of operands of a domain element
`genident` : generator of unused identifiers
`getprop` : query of properties
`has` : test for subexpressions
`hastype` : test for subexpressions of a given type
`history` : the history table of a MuPAD session
`indets` : the indeterminates of an expression
`is` : query of properties of an identifier
`lasterror` : re-initiate a trapped error message
`length` : size of an object
`lhs` : left hand side of an expression
`map` : mapping a function onto the operands of an object
`maprat` : mapping a function onto a "rationalized" expression
`match` : pattern matching in expressions
`nops` : number of operands of an object
`numer` : numerator of a rational expression
`op` : operands of an object
`protect` : write protection

<code>radsimp</code>	:	simplification of expressions with radicals
<code>rationalize</code>	:	conversion of arbitrary expressions into rational expressions
<code>rewrite</code>	:	rewriting expressions
<code>rhs</code>	:	right hand side of an expression
<code>select</code>	:	selection of operands according to properties
<code>simplify</code>	:	simplifier
<code>slot</code>	:	defines or reads an attribute of a domain or a function environment
<code>split</code>	:	splitting objects according to properties
<code>subs</code>	:	substitution
<code>subsex</code>	:	substitution of more complex expressions
<code>subsop</code>	:	substitution of operands
<code>sysorder</code>	:	information about the order of objects in the MuPAD kernel
<code>testtype</code>	:	type comparison
<code>traperror</code>	:	trapping errors
<code>type</code>	:	type of an object
<code>unalias</code>	:	deleting an abbreviation
<code>unassume</code>	:	deleting assumptions about properties of identifiers
<code>unexport</code>	:	unloading libraries
<code>unprotect</code>	:	remove write protection of identifiers
<code>zip</code>	:	merging of lists and matrices

More functions for handling data objects are available in various libraries (section 17).

11 Manipulation of Strings

<code>_concat, .</code>	:	concatenation of strings
<code>expr2text</code>	:	conversion of an expression to a string
<code>ftextinput</code>	:	text input from a file
<code>int2text</code>	:	conversion of an integer to a string
<code>length</code>	:	length of a string
<code>revert</code>	:	reverting a string
<code>strmatch</code>	:	string matching
<code>substring</code>	:	selecting substrings
<code>tbl2text</code>	:	conversion of a table to a string
<code>text2expr</code>	:	conversion of a string to an expression
<code>text2int</code>	:	conversion of a string to an integer
<code>text2list</code>	:	conversion of a string to a list
<code>text2tbl</code>	:	conversion of a table to a string
<code>textinput</code>	:	interactive input of strings

More functions manipulating strings are available in the library `stringlib`.

12 Functions for Input, Output, and Graphics

<code>error</code>	: raising an error and printing an error message
<code>fclose</code>	: closing a file
<code>fininput</code>	: reading from a file
<code>fopen</code>	: opening a file
<code>fprint</code>	: writing into a file
<code>fread</code>	: reading from a file
<code>ftextinput</code>	: text input from a file
<code>import::readdata</code>	: reading formatted ASCII data
<code>input</code>	: interactive assignment
<code>plot</code>	: plotting graphical objects
<code>plot2d</code>	: 2-dimensional plots
<code>plot3d</code>	: 3-dimensional plots
<code>plotfunc2d</code>	: plotting the graph of a unary function
<code>plotfunc3d</code>	: plotting the graph of a binary function
<code>print</code>	: screen output
<code>protocol</code>	: opening and closing a session protocol
<code>read</code>	: reading from a file
<code>readbytes</code>	: reading from a binary file
<code>setuserinfo</code>	: setting the "information level" for <code>userinfo</code>
<code>textinput</code>	: interactive input of strings
<code>userinfo</code>	: print runtime information inside algorithms
<code>warning</code>	: print a warning
<code>write</code>	: saving values into a file
<code>writebytes</code>	: writing into a binary file

For input/output from/to a file, an overview can be requested by `?fileIO`. For graphics, also see the documentation of the `plot` library (`?plot`).

13 Controlling Evaluation

<code>bool</code>	: evaluation of boolean expressions to <code>TRUE</code> or <code>FALSE</code>
<code>context</code>	: evaluation in the context
<code>eval</code>	: forcing evaluation
<code>evalassign</code>	: assignment with evaluation of the left hand side
<code>freeze</code>	: make a function inactive
<code>hold</code>	: prevents evaluation
<code>indexval</code>	: indexed access without evaluation
<code>last, %</code>	: access to previous results
<code>level</code>	: evaluation with a given depth
<code>unfreeze</code>	: re-activate a function
<code>val</code>	: value of an identifier (equivalent to <code>level(.,1)</code> , but without internal simplification)

14 Functions in Procedure Definitions

`_procdef` : procedure definition

args : access to procedure arguments
error : raise an error
return : returning of function results
testargs : control argument checking
warning : print a warning

15 Mathematical Functions and Algorithms

abs : absolute value
arg : polar angle of a complex number
asympt : asymptotic expansion
binomial : binomial expression $\binom{m}{n}$
ceil : smallest integer $\geq x$
conjugate : complex conjugation
contfrac : continued fraction expansion
D and ' : differential operator
diff : differentiation of expressions
discont : discontinuities of a function
div : division "modulo"
factor : factorization of polynomials, expressions and integers
float : conversion into floating-point numbers
floor : Gaussian brackets: greatest integer $\leq x$
frac : fractional part of a number
frandom : random number generator for floating point numbers
gcd : greatest common divisor
gcdex : greatest common divisor (extended Euclidean algorithm)
ifactor : prime factorization
igcd : greatest common divisor of integers
igcdex : greatest common divisor of integers (extended)
ilcm : least common multiple of integers
Im : imaginary part
int : integration
interpolate : polynomial interpolation
intersect : intersection of sets
isprime : prime number test
isqrt : integer approximation of the square root of an integer number
iszero : test for 0
ithprime : i -th prime number
lcm : least common multiple
limit : limit computation
linsolve : solution of linear equations
lllint : reduced basis of a lattice
max : maximum
min : minimum
minus : difference of sets

```

mod, modp, mods : remainder "modulo"
nextprime      : next prime number  $\geq x$ 
norm          : norm of polynomials, vectors and matrices
normal        : normal form of a rational expression
not           : logical negation
or            : logical "and"
pade          : Padé approximation
partfrac      : partial fraction decomposition
powermod      : power "modulo"
product       : generator of products
random        : random number generator
Re            : real part
rectform      : decomposition into real and imaginary part
revert        : reverse function (for series, strings and lists)
round         : rounding to next number
series        : series expansion
sign          : (complex) sign
signIm        : sign of the imaginary part
solve         : equation solver
sort          : sorting lists
sum           : computing sums
taylor        : Taylor expansion
trunc         : truncates a number
union         : union of sets

```

More functions for special mathematical topics are available in the libraries (Section 17).

16 Solving Equations

Apart from the general `solve` command for solving algebraic equations, differential equations and recursion equations, there is a variety of specialized solvers for special types of equations.

type of equation	available solvers
system of linear equations	<code>solve</code> <code>linsolve</code> <code>linalg::matlinsolve</code> <code>linalg::matlinsolveLU</code> <code>linalg::vandermondeSolve</code> <code>numeric::linsolve</code> <code>numeric::matlinsolve</code> <code>numeric::solve</code>
univariate polynomial equation	<code>solve</code> <code>polylib::realroots</code> <code>numeric::solve</code> <code>numeric::polyroots</code>

bivariate polynomial equation	<code>solve</code> <code>series</code>
system of polynomial equations	<code>solve</code> <code>numeric::solve</code> <code>numeric::polysysroots</code>
arbitrary univariate equation	<code>solve</code> <code>numeric::solve</code> <code>numeric::realroot</code> <code>numeric::realroots</code>
system of arbitrary equations	<code>solve</code> <code>numeric::solve</code> <code>numeric::fsolve</code>
inequalities	<code>solve</code>
system of ordinary differential equations	<code>solve</code> <code>numeric::odesolve</code> <code>numeric::odesolve2</code> <code>numeric::odesolveGeometric</code>
recurrence equation	<code>solve</code>
partial differential equations	<code>detools::pdesolve</code>
congruence	<code>numlib::lincongruence</code> <code>numlib::mroots</code> <code>numlib::msqrts</code>

An overview of the available solvers can be requested by `?solvers`.

17 Libraries and Modules

The following libraries and modules in MuPAD version 2.5 contain many more functions and algorithms for special mathematical topics. The libraries are in a permanent development: later MuPAD versions will provide additional libraries and functions. An overview over the current installed functions is provided by `info` or `?`. Help on single functions can be obtained by `?libraryname::functionname`. Help on modules is available via `modulename::doc()`.

```

adt      : abstract data types
Ax       : generator of axioms
Cat      : generator of categories
combinat : combinatorics
detools  : differential equations
Dom      : predefined data structures: fields, rings, matrices, etc.
fp       : functional programming
generate : generation of C, Fortran and TeX Code
groebner : computation of polynomial ideals
import   : import external data formats
intl    : integration
linalg   : linear algebra
linopt   : linear optimization

```

`listlib` : manipulation of lists
`matchlib` : pattern matching in expressions
`misc` : miscellaneous
`module` : module administration
`Network` : graphs
`numeric` : numerical algorithms
`numlib` : number theory
`orthpoly` : orthogonal polynomials
`output` : output of objects
`plot` : plotting routines
`polylib` : algorithms for polynomials
`Pref` : user preferences
`prog` : programming and debugging
`property` : properties of identifiers
`RGB` : color names (red–green–blue values)
`solverlib` : solving equations
`stats` : statistical functions
`stdlib` : standard library
`stdmod` : extended module management
`stringlib` : manipulation of strings
`student` : some elementary algorithms
`transform` : integral transformations
`Type` : type specifiers
`util` : module with utility functions

18 Debugging and Timing

`debug` : debugging a procedure
`prog::check` : check procedures and domains for global variables etc.
`prog::profile` : runtime statistics
`prog::trace` : tracing procedure calls
`rtime` : real-time usage
`time` : CPU-time usage

More programming utilities are available in the library `prog`.

19 Technical Issues

`builtin` : functions of the MuPAD kernel
`bytes` : memory usage
`export` : loading libraries
`external` : module administration
`getpid` : process number of the current MuPAD session
`loadlib` : loading libraries
`loadmod` : loading modules
`loadproc` : loading procedures

`operator` : define a new operator
`package` : loading user-defined libraries
`patchlevel` : information on MuPAD patches
`pathname` : setting system dependent path names
`quit` : quit a MuPAD session
`register` : registering MuPAD
`reset` : resetting a MuPAD session
`share` : create unique data representation
`sysname` : name of the operating system
`system` : execution of commands by the operating system
`unexport` : unloading libraries
`unloadmod` : removing loaded modules
`version` : information on the MuPAD version number