MuPAD 2.0
Quick Reference

W. Oevel and J. Gerhard

MuPAD

The Open Computer Algebra System

MuPAD 2.0
Quick Reference

This paper gives an overview of the data structures, constants, functions and libraries in MuPAD version 2.0. More details are available in the tutorial

W. Oevel, F. Postel, S. Wehmeier, and J. Gerhard.
The MuPAD Tutorial. Springer, 2000.

This document is available online.

Contents

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.

	Default Value	
`DIGITS`	10	significant digits in floating point operations
`HISTORY`	20	number of results available via `last`
`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?
`READPATH`		pathname for files to be read via `read`
`SEED`	1	seed for generating random numbers via `random`
`TEXTWIDTH`	75	text width for output
`WRITEPATH`		pathname for files to be saved via `write`

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_LIST`	:	lists
`DOM_NIL`	:	the object `NIL`
`DOM_NULL`	:	the empty 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 procedure variables

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

`Factored`	:	objects in factored form
`O`	:	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::Puiseux`	:	series expansions

More data structures and their constructors are available in the library `Dom` (information via `info(Dom)`).

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 "empty object"(`DOM_NULL`)
`O`	:	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::Puiseux` or `Series::gseries`)
`slot`	:	defines or reads an attribute of a domain or function environment
`taylor`	:	Taylor expansion (generates `Series::Puiseux`)
`table`	:	generates a table (`DOM_TABLE`)

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

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
`::`	2000	`slot`	method access
`'`	1900	`D`	differential operator
`[]`	1800	`_index`	index operator
`.`	1700	`_concat`	concatenation
`@@`	1600	`_fnest`	iterated composition
`@`	1500	`_fconcat`	composition of functions
`!`	1300	`fact`	factorial function
`^`	1200	`_power`	power
`*`	1100	`_mult`	multiplication
`/`	1100	`_divide`	division
`+`	1000	`_plus`	addition
`-`	1000	`_subtract` and `_negate`	subtraction negation
`div`	900	`_div`	quotient "modulo"
`mod`	900	`_mod`	remainder "modulo"
`intersect`	800	`_intersect`	intersection of sets
`minus`	700	`_minus`	difference of sets
`union`	600	`_union`	union of sets
`..`	500	`_range`	range
`=`	400	`_equal`	equation
`<>`	400	`_unequal`	inequality
`<` and `>`	400	`_less`	comparisons
`<=` and `>=`	400	`_leequal`	comparisons
`in`	300	`_in`	containment
`$`	300	`_seqgen` and `_seqin`	sequence generator
`not`	300	`_not`	logical negation
`and`	200	`_and`	logical "and"
`or`	100	`_or`	logical "or"
`,`		`_exprseq`	separator between elements of sequences
`;` and `:`		`_stmtseq`	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
`_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)
`_if`	:	branch
`_index`	:	indexed expression
`_intersect`	:	intersection of sets
`_invert`	:	inversion
`_lazy_and`	:	lazily evaluated logical "and"
`_lazy_or`	:	lazily evaluated logical "or"
`_leequal`	:	inequality $\leq$
`_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

7 Symbols and Constants

`C_`	:	mathematical 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, only 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_`	:	mathematical set $\mathbb{Q}$ of rational numbers
`R_`	:	mathematical set $\mathbb{R}$ of real numbers
`TRUE`	:	Boolean constant
`undefined`	:	undefined value
`universe`	:	set-theoretic universe of all objects
`UNKNOWN`	:	Boolean constant
`Z_`	:	mathematical set $\mathbb{Z}$ of integers

8 Special Mathematical Functions

The following mathematical functions are implemented in MuPAD version 2.0. 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`
`arcsech`	:	inverse function of `sech`
`arcsin`	:	inverse function of `sin`
`arcsinh`	:	inverse function of `sinh`
`arctan`	:	inverse function of `tan`
`arctanh`	:	inverse function of `tanh`
`arg`	:	polar angle of a complex number
`bernoulli`	:	Bernoulli numbers and polynomials
`besselI`	:	modified Bessel function of the first kind
`besselJ`	:	Bessel function of the first kind
`besselK`	:	modified Bessel function of the second kind
`besselY`	:	Bessel function of the second kind
`beta`	:	β–function
`binomial`	:	binomial expression $\binom{m}{n}$
`ceil`	:	smallest integer $\geq x$

`Ci`	:	$\gamma + ln(x) + \int_0^x (\cos(t) - 1)/t \, dt$
`cos`	:	cosine function
`cosh`	:	hyperbolic cosine function
`cot`	:	cotangent function
`coth`	:	hyperbolic cotangent function
`csc`	:	$1/\sin(x)$
`csch`	:	$1/\sinh(x)$
`dilog`	:	dilogarithm function
`dirac`	:	Dirac's δ–function
`Ei`	:	$\int_x^\infty e^{-t}/t \, dt$
`erf`	:	$(2/\sqrt{\pi}) \ \int_0^x e^{-t^2} \, dt$
`erfc`	:	$1 - \mathrm{erf}(x)$
`exp`	:	exponential function
`fact`	:	factorial function
`floor`	:	Gaussian brackets: greatest integer $\leq x$
`frac`	:	fractional part of a number
`gamma`	:	Γ–function
`heaviside`	:	Heaviside function
`id`	:	identity map $x \to x$
`igamma`	:	incomplete Γ–function
`Im`	:	imaginary part
`lambertV`	:	lower real branch of the Lambert function
`lambertW`	:	upper real branch of the Lambert function
`ln`	:	natural logarithm
`log`	:	logarithm to an arbitrary base
`polylog`	:	polylogarithm function
`psi`	:	polygamma function
`Re`	:	real part
`round`	:	rounding to next number
`sec`	:	$1/\cos(x)$
`sech`	:	$1/\cosh(x)$
`Si`	:	$\int_0^x \sin(t)/t \, dt$
`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
`extsubsop`	:	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
`radsimp`	:	simplification of expressions with radicals
`rationalize`	:	conversion of arbitrary expressions into rational expressions
`rewrite`	:	rewriting expressions
`rhs`	:	right hand side of an expression
`select`	:	selection of operands according to properties
`simplify`	:	simplifier
`slot`	:	defines or reads an attribute of a domain or a function environment
`split`	:	splitting objects according to properties
`subs`	:	substitution
`subsex`	:	substitution of more complex expressions
`subsop`	:	substitution of operands
`sysorder`	:	information about the order of objects in the MuPAD kernel
`testtype`	:	type comparison

`traperror`	:	trapping errors
`type`	:	type of an object
`unalias`	:	deleting an abbreviation
`unassume`	:	deleting assumptions about properties of identifiers
`unexport`	:	unloading libraries
`unprotect`	:	remove write protection of identifiers
`zip`	:	merging of lists and matrices

11 Manipulation of Strings

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

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

12 Functions for Input, Output, and Graphics

`error`	:	output of an error
`fclose`	:	closing a file
`finput`	:	reading from a file
`fopen`	:	opening a file
`fprint`	:	writing into a file
`fread`	:	reading from a file
`ftextinput`	:	text input from a file
`input`	:	interactive assignment
`plot`	:	plotting graphical objects
`plot2d`	:	2–dimensional plots
`plot3d`	:	3–dimensional plots
`plotfunc2d`	:	plotting the graph of a unary function
`plotfunc3d`	:	plotting the graph of a binary function
`print`	:	screen output
`protocol`	:	opening and closing a session protocol
`read`	:	reading from a file
`setuserinfo`	:	setting the "information level"for `userinfo`
`textinput`	:	interactive input of strings

`userinfo`	:	output of information on algorithms
`warning`	:	output of a warning
`write`	:	saving values into a file

More functions for input, output, and graphics are available in the libraries `import`, `output`, and `plot`, respectively.

13 Controlling Evaluation

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

14 Functions in Procedure Definitions

`_procdef`	:	procedure definition
`args`	:	access to procedure arguments
`error`	:	output of error messages
`return`	:	returning of function results
`testargs`	:	controlling argument checking
`warning`	:	output of 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
`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
`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
`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 16).

16 Libraries and Modules

The following libraries and modules in MuPAD version 2.0 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
`intlib`	:	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)
`solvelib`	:	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

17 Debugging and Timing

`debug`	:	debugging a procedure
`rtime`	:	real-time usage
`time`	:	CPU-time usage

More functions for debugging are available in the library `prog`.

18 Technical Issues

`builtin`	:	functions of the MuPAD kernel
`bytes`	:	memory usage
`export`	:	loading libraries
`external`	:	module administration
`getpid`	:	process no. of MuPAD
`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