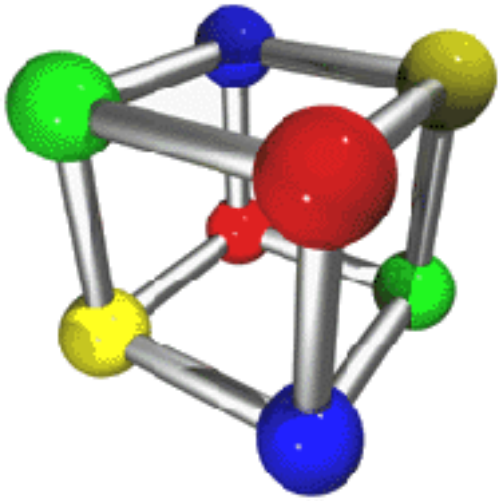
quickref —

MuPAD 3.1
Kurzübersicht

W. Oevel



MuPAD

The Open Computer Algebra System

Strukturen in MuPAD 3.1: eine Übersicht

Es wird eine Übersicht über die in MuPAD Version 3.1 installierten Datenstrukturen, Konstanten, Funktionen und Bibliotheken gegeben. Genauere Informationen sind im Tutorium

C. Creutzig, K. Gehrs und W. Oevel.

Das MuPAD-Tutorium. Dritte Auflage. Springer, 2004.

zu finden. Eine aktuelle Version steht im Hilfesystem zur Verfügung.

Inhaltsverzeichnis

1 Informationen und Hilfe

`help`, `?`	:	Anfordern von Hilfeseiten
`info`	:	Ausgabe von Kurzinformationen
`setuserinfo`	:	Informationen beim Ablauf von Algorithmen

2 Umgebungsvariablen

Umgebungsvariablen sind globale Variablen, die den Ablauf der in den MuPAD-Prozeduren implementierten Algorithmen beeinflussen. Die voreingestellten Werte können vom Nutzer im Laufe einer Sitzung abgeändert werden.

Variable	Voreinstellung	Bedeutung
`DIGITS`	10	signifikante Dezimalstellen in Gleitpunktrechnungen
`HISTORY`	20	Anzahl der mit `last` zugreifbaren Ergebnisse
`LEVEL`	100	Auswertungstiefe
`LIBPATH`		Pfadname des MuPAD-Programms und der Bibliotheken
`MAXDEPTH`	500	maximal zulässige Rekursionstiefe
`MAXLEVEL`	100	maximal zulässige Auswertungstiefe
`NOTEBOOKFILE`		Pfadname des aktuellen MuPADNotebook
`NOTEBOOKPATH`		Pfadname des aktuellen MuPADNotebook
`ORDER`	6	Standardanzahl Terme in Reihenentwicklungen
`PACKAGEPATH`		Pfadname für mit `package` zu ladende Pakete
`PRETTYPRINT`	`TRUE`	2-dimensionale Ausgabe?
`READPATH`		Pfadname für mit `read` einzulesende Dateien
`SEED`	1	Startwert für Generierung von Zufallszahlen mittels `random`
`TEXTWIDTH`	75	Textbreite der Ausgabe
`WRITEPATH`		Pfadname für mit `write` etc. abzuspeichernde Dateien

3 Vordefinierte Datentypen (Domains)

Datentypen des MuPAD-Kerns:

`DOM_ARRAY`	:	Felder
`DOM_BOOL`	:	die logischen Werte `TRUE`, `FALSE` und `UNKNOWN`
`DOM_COMPLEX`	:	komplexe Zahlen
`DOM_DOMAIN`	:	Datenstrukturen
`DOM_EXPR`	:	Ausdrücke
`DOM_FAIL`	:	das Objekt `FAIL`
`DOM_FLOAT`	:	reelle Gleitpunktzahlen
`DOM_FUNC_ENV`	:	Funktionsumgebungen

`DOM_IDENT`	:	Bezeichner
`DOM_INT`	:	ganze Zahlen
`DOM_INTERVAL`	:	Gleitpunktintervalle
`DOM_LIST`	:	Listen
`DOM_NIL`	:	das Objekt `NIL`
`DOM_NULL`	:	das leere Objekt `null()`
`DOM_POLY`	:	Polynome
`DOM_PROC`	:	Prozeduren
`DOM_RAT`	:	rationale Zahlen
`DOM_SET`	:	Mengen
`DOM_STRING`	:	Zeichenketten
`DOM_TABLE`	:	Tabellen
`DOM_VAR`	:	lokale Variablen einer Prozedur

In der MuPAD-Sprache definierte Datentypen der Standardbibliothek:

`Factored`	:	Objekte in faktorisierter Form
`matrix`	:	Matrizen und Vektoren
`O`	:	Ordnungsterm von Reihenentwicklungen
`ode`	:	gewöhnliche Differentialgleichungen
`piecewise`	:	fallweise definierte Objekte
`rec`	:	Rekurrenzgleichungen
`rectform`	:	Kartesische Darstellung komplexer Zahlen
`Series::gseries`	:	verallgemeinerte Reihenentwicklungen
`Series::Puiseux`	:	Reihenentwicklungen

Viele weitere Datenstrukturen sind mit ihren Erzeugern in der Bibliothek `Dom` installiert. Für weitere Informationen rufe man `?Dom` oder `info(Dom)` auf.

4 Erzeugung spezieller Datenstrukturen

`array`	:	erzeugt ein Feld (`DOM_ARRAY`)
`asympt`	:	asymptotische Entwicklung (erzeugt `Series::gseries`)
`contfrac`	:	Kettenbruchentwicklung (erzeugt `contfrac`-Objekte)
`factor`, `ifactor`	:	Zerlegung in irreduzible Faktoren (erzeugt `Factored`)
`funcenv`	:	Definition einer Funktionsumgebung (erzeugt `DOM_FUNC_ENV`)
`matrix`	:	erzeugt eine Matrix vom Domaintyp `Dom::Matrix()`
`new`	:	Erzeuger eines Domain-Elementes
`newDomain`	:	Erzeuger eines Domains
`null`	:	erzeugt das „leere Objekt“ (`DOM_NULL`)
`O`	:	erzeugt den Ordnungsterm von Reihenentwicklungen
`ode`	:	erzeugt eine gewöhnliche Differentialgleichung
`piecewise`	:	erzeugt ein durch Fallunterscheidung definiertes Objekt
`poly`	:	erzeugt ein Polynom (`DOM_POLY`)
`rec`	:	erzeugt eine Rekurrenzgleichung
`rectform`	:	Kartesische Darstellung komplexer Zahlen (erzeugt `rectform`)

`RootOf`	:	alle Nullstellen eines Polynoms (erzeugt `RootOf`-Objekte)
`series`	:	allgemeine Reihenentwicklung (erzeugt `Series::Puiseux` oder `Series::gseries`)
`slot`	:	Setzen oder Lesen eines Attributs eines Domains oder einer Funktionsumgebung
`taylor`	:	Taylor-Entwicklung (erzeugt `Series::Puiseux`)
`table`	:	erzeugt eine Tabelle (`DOM_TABLE`)

Viele Erzeuger weiterer Datenstrukturen sind in der Bibliothek `Dom` installiert, z. B.:

`Dom::IntegerMod(p)`	:	Erzeuger von ganzen Zahlen modulo p
`Dom::Matrix()`	:	Erzeuger von Matrizen
...	:	...

5 Operatoren

Mit folgenden Operatoren können die angegebenen Systemfunktionen in intuitiver Form aufgerufen werden, z. B. `a + b` statt `_plus(a, b)`, `x < 1` statt `_less(x, 1)`, usw.:

Operator	Bindungs-stärke	System-funktion	Bedeutung
`::`	2000	`slot`	Methodenzugriff
`'`	1900	`D`	Differentialoperator
`[]`	1800	`_index`	Indexoperator
`.`	1700	`_concat`	Konkatenation
`@@`	1600	`_fnest`	Iteration
`@`	1500	`_fconcat`	Komposition von Funktionen
`!`	1300	`fact`	Fakultät
`!!`	1300	`fact2`	Doppelfakultätsfunktion
`^`	1200	`_power`	Potenzieren
`*`	1100	`_mult`	Multiplikation
`/`	1100	`_divide`	Division
`-`	1050	`_negate`	Negation
`+`	1000	`_plus`	Addition
`-`	1000	`_subtract`	Subtraktion
`div`	900	`_div`	Quotient „modulo“
`mod`	900	`_mod`	Rest „modulo“
`...`	850	`hull`	Gleitpunktintervalle
`intersect`	800	`_intersect`	Schnitt von Mengen
`minus`	700	`_minus`	Differenz von Mengen
`union`	600	`_union`	Vereinigung von Mengen
`..`	500	`_range`	Bereich
`=`	400	`_equal`	Gleichung
`<>`	400	`_unequal`	Ungleichung
`<` und `>`	400	`_less`	Größenvergleiche

<= und >=	400	_leequal	Größenvergleiche
in	400	_in	Elementbeziehung
subset	400	_subset	Untermengenbeziehung
$	300	_seqgen und _seqin	Folgengenerator
not	300	_not	logische Verneinung
and	200	_and	logisches „und“
xor	150	_xor	logisches „exklusives oder“
or	100	_or	logisches „oder“
assuming	100	_assuming	Annahme
==>	75	_implies	logisches ”daraus folgt”
<=>	50	_equiv	logische Äquivalenz
|	35	evalAt	Auswertung an einem Punkt
,	20	_exprseq	Erzeugen von Folgen
; und :	10	_stmtseq	Trennung zwischen Befehlen

Benutzer können mittels `operator` ihre eigenen Operatoren definieren.

6 Arithmetikfunktionen

Die folgenden Funktionen können auch mittels Operatoren oder Schlüsselwörtern aufgerufen werden, z. B. `a+b` statt `_plus(a,b)`, `-a` statt `_negate(a)`, etc.:

_and	:	logisches „und“
_assign	:	Zuweisungsfunktion
_break	:	Unterbrechung von Schleifen etc.
_case	:	Verzweigung
_concat	:	Konkatenation
_delete	:	Löschen von Werten oder Elementen
_div	:	Quotient „modulo“
_divide	:	Division
_equal	:	Gleichung
_equiv	:	logische Äquivalenz
_exprseq	:	Folgengenerator
_fconcat	:	Hintereinanderschaltung von Funktionen
_fnest	:	Iteration von Funktionen
_for	:	Schleife (aufwärts)
_for_down	:	Schleife (abwärts)
_for_in	:	Schleife (durchläuft die Operanden eines Ausdrucks)
hull	:	Erzeugung eines Gleitpunktintervalls
_if	:	Verzweigung
_implies	:	logisches „daraus folgt“
_index	:	indizierter Aufruf
_intersect	:	Schnitt von Mengen
_invert	:	Invertierung
_lazy_and	:	„lazy“ ausgewertetes logisches „und“
_lazy_or	:	„lazy“ ausgewertetes logisches „oder“

`_leequal`	:	Ungleichung $\leq$
`_less`	:	Ungleichung $<$
`_minus`	:	Differenz von Mengen
`_mod`	:	Rest „modulo“
`_mult`	:	Multiplikation
`_negate`	:	Multiplikation mit -1
`_next`	:	Sprung in einer Schleife
`_not`	:	logische Verneinung
`_or`	:	logisches „oder“
`_plus`	:	Addition
`_power`	:	Potenzierung
`_procdef`	:	Prozedurdefinition
`_quit`	:	Ende einer MuPAD-Sitzung
`_range`	:	Bereich
`_repeat`	:	Schleife
`_seqgen`	:	Folgengenerator
`_seqin`	:	Folgengenerator
`_stmtseq`	:	Folge von Befehlen
`_subtract`	:	Differenz
`_unequal`	:	Ungleichung <>
`_union`	:	Vereinigung von Mengen
`_while`	:	Schleife
`_xor`	:	logisches „exklusives oder“

7 Symbole und Konstanten

`C_`	:	die Menge $\mathbb{C}$ der komplexen Zahlen
`CATALAN`	:	Catalan-Konstante: `float(CATALAN)=0.9159655941..`
`complexInfinity`	:	der unendlich ferne Punkt der komplexen Ebene
`E`	:	Eulersche Zahl $\exp(1) = 2.718281828..$
`EULER`	:	Eulersche Konstante $\gamma = 0.5772156649..$
`FAIL`	:	Fehlerobjekt, einziges Objekt vom Domaintyp `DOM_FAIL`
`FALSE`	:	Boolesche Konstante
`I`	:	imaginäre Einheit $\sqrt{-1}$
`infinity`	:	Unendlich
`NIL`	:	Nullobjekt, einziges Objekt vom Domaintyp `DOM_NIL`
`PI`	:	$\pi = 3.141592653..$
`Q_`	:	die Menge $\mathbb{Q}$ der rationalen Zahlen
`R_`	:	die Menge $\mathbb{R}$ der reellen Zahlen
`RD_INF`, `RD_NINF`	:	„gerundetes“ $\pm$ Unendlich für Intervallarithmetik
`TRUE`	:	Boolesche Konstante
`undefined`	:	nicht definierter Wert
`unit`	:	Domain physikalischer Einheiten
`universe`	:	mengentheoretisches Universum aller Objekte
`UNKNOWN`	:	Boolesche Konstante
`Z_`	:	die Menge $\mathbb{Z}$ der ganzen Zahlen

8 Spezielle mathematische Funktionen

Folgende mathematischen Funktionen sind in MuPAD Version 3.1 implementiert, d. h., sie sind Systemfunktionen wie `expand`, `float`, `simplify`, usw. bekannt und werden von ihnen verarbeitet:

`abs`	:	Absolutbetrag
`airyAi`	:	Airy-Funktion Ai
`airyBi`	:	Airy-Funktion Bi
`arccos`	:	Umkehrfunktion von `cos`
`arccosh`	:	Umkehrfunktion von `cosh`
`arccot`	:	Umkehrfunktion von `cot`
`arccoth`	:	Umkehrfunktion von `coth`
`arccsc`	:	Umkehrfunktion von `csc`
`arccsch`	:	Umkehrfunktion von `csch`
`arcsec`	:	Umkehrfunktion von `sec`
`arcsech`	:	Umkehrfunktion von `sech`
`arcsin`	:	Umkehrfunktion von `sin`
`arcsinh`	:	Umkehrfunktion von `sinh`
`arctan`	:	Umkehrfunktion von `tan`
`arctanh`	:	Umkehrfunktion von `tanh`
`arg`	:	Polarwinkel einer komplexen Zahl
`bernoulli`	:	Bernoulli-Zahlen und -Polynome
`besselI`	:	modifizierte Bessel-Funktion erster Art
`besselJ`	:	Bessel-Funktion erster Art
`besselK`	:	modifizierte Bessel-Funktion zweiter Art
`besselY`	:	Bessel-Funktion zweiter Art
`beta`	:	β-Funktion
`binomial`	:	Binomialausdruck $\binom{m}{n}$
`ceil`	:	kleinste ganze Zahl $\geq x$
`Ci`	:	$\gamma + \ln(x) + \int_0^x (\cos(t) - 1)/t \, dt$
`cos`	:	Cosinus-Funktion
`cosh`	:	Cosinus Hyperbolicus-Funktion
`cot`	:	Cotangens-Funktion
`coth`	:	Cotangens Hyperbolicus-Funktion
`csc`	:	$1/\sin(x)$
`csch`	:	$1/\sinh(x)$
`dilog`	:	Dilogarithmus-Funktion
`dirac`	:	Diracsche δ-Funktion
`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`	:	Exponentialfunktion
`fact`	:	Fakultätsfunktion
`fact2`	:	Doppelfakultätsfunktion
`floor`	:	Gaußklammer: größte ganze Zahl $\leq x$
`frac`	:	Nachkommastellen einer Zahl
`gamma`	:	Γ-Funktion

`heaviside`	:	Sprungfunktion
`hypergeom`	:	hypergeometrische Funktion
`id`	:	identische Abbildung $x \to x$
`igamma`	:	unvollständige Γ-Funktion
`Im`	:	Imaginärteil
`kummerU`	:	Kummers konfluente hypergeometrische U-Funktion
`lambertW`	:	Lamberts W-Funktion
`ln`	:	natürlicher Logarithmus
`log`	:	Logarithmus zu beliebiger Basis
`polylog`	:	Polylogarithmus-Funktion
`psi`	:	Polygamma-Funktion
`Re`	:	Realteil
`round`	:	Rundung
`sec`	:	$1/\cos(x)$
`sech`	:	$1/\cosh(x)$
`Si`	:	$\int_0^x \sin(t)/t\,dt$
`sign`	:	(komplexes) Vorzeichen
`signIm`	:	Vorzeichen des Imaginärteils
`sin`	:	Sinus-Funktion
`sinh`	:	Sinus Hyperbolicus-Funktion
`sqrt`	:	Wurzelfunktion
`surd`	:	n-te Wurzel
`tan`	:	Tangens-Funktion
`tanh`	:	Tangens Hyperbolicus-Funktion
`trunc`	:	Abschneiden von Nachkommastellen
`whittakerM`	:	Whittakers M-function
`whittakerW`	:	Whittakers W-function
`zeta`	:	Riemannsche ζ-Funktion

9 Funktionen zur Behandlung von Polynomen

Folgende Funktionen verarbeiten mittels `poly` erzeugte Polynome vom Domain-Typ `DOM_POLY` und teilweise auch polynomiale Ausdrücke vom Domain-Typ `DOM_EXPR`:

`coeff`	:	Koeffizienten
`collect`	:	Zusammenfassen von Koeffizienten
`content`	:	größter gemeinsamer Teiler aller Koeffizienten
`degree`	:	Polynomgrad
`degreevec`	:	Exponenten des führenden Terms
`diff`, `Ddivide`	:	Division mit Rest
`evalAt`	:	Auswertung an einem Punkt
`evalp`	:	Auswertung eines Polynoms an einem Punkt
`expr`	:	Konvertierung in einen Ausdruck
`factor`	:	Faktorisierung
`gcd`	:	größter gemeinsamer Teiler

`gcdex`	:	größter gemeinsamer Teiler (erweiterter Euklidischer Algorithmus)
`genpoly`	:	Polynomerzeugung mittels b-adischer Entwicklung
`ground`	:	konstanter Koeffizient eines Polynoms
`icontent`	:	größter gemeinsamer Teiler aller Koeffizienten
`irreducible`	:	Test auf Irreduzibilität
`iszero`	:	Test auf 0
`lcm`	:	kleinstes gemeinsames Vielfaches
`lcoeff`	:	führender Koeffizient
`ldegree`	:	niedrigster Grad innerhalb eines Polynoms
`lmonomial`	:	führendes Monom
`lterm`	:	führender Term
`mapcoeffs`	:	Anwendung einer Funktion auf die Koeffizienten
`monomials`	:	Monome
`multcoeffs`	:	Multiplikation mit einem Skalar
`norm`	:	Norm
`nterms`	:	Anzahl der Terme
`nthcoeff`	:	n-ter Koeffizient
`nthmonomial`	:	n-tes Monom
`nthterm`	:	n-ter Term
`pdivide`	:	Pseudodivision
`poly`	:	Erzeuger eines Polynoms
`poly2list`	:	Konvertierung in eine Liste
`tcoeff`	:	niedrigster Koeffizient

Weitere Funktionen, die auf Polynomen arbeiten, bietet die Bibliothek `polylib`.

10 Umgang mit MuPAD-Objekten

`alias`	:	Definition von Abkürzungen
`anames`	:	Auflistung aller Bezeichner, die einen Wert haben
`append`	:	Anfügen an eine Liste
`assign`	:	Zuweisungsfunktion
`assignElements`	:	Zuweisungsfunktion für Felder, Listen und Tabellen
`assume`	:	Setzen von Annahmen über Eigenschaften von Bezeichnern
`bool`	:	Auswertung zu `TRUE` oder `FALSE`
`coerce`	:	Typkonversion
`collect`	:	Zusammenfassen von Koeffizienten polynomialer Ausdrücke
`combine`	:	Zusammenfassen von Teilausdrücken
`contains`	:	Test auf Elemente in Listen, Mengen und Tabellen
`delete`	:	Löschen des Wertes eines Bezeichners
`denom`	:	Nenner eines rationalen Ausdrucks
`domain`	:	Definition eines neuen Domains
`domtype`	:	Domain-Typ
`expand`	:	Expansion von Ausdrücken
`export`	:	Einladen von Bibliotheken

`expose`	:	Ausgabe der Operanden von Funktionsumgebungen und Domains
`expr`	:	Konvertierung diverser Datenstrukturen in Ausdrücke, Felder, etc.
`extnops`	:	Anzahl der Operanden eines Domains
`extop`	:	die Operanden eines Domains
`extsubsop`	:	Substitution von Operanden eines Domains
`genident`	:	Erzeuger eines unbenutzten Bezeichners
`getprop`	:	Abfrage von Eigenschaften
`has`	:	Test auf Teilausdrücke
`hastype`	:	Test auf Teilausdrücke eines gegebenen Typs
`history`	:	die History-Tabelle einer MuPAD-Sitzung
`indets`	:	die Unbestimmten eines Ausdrucks
`is`	:	Abfrage von Eigenschaften eines Bezeichners
`lasterror`	:	erneutes Auslösen einer abgefangenen Fehlers
`length`	:	Komplexität eines Objekts
`lhs`	:	linke Seite eines Ausdrucks
`map`	:	Anwendung einer Funktion auf die Operanden eines Objekts
`maprat`	:	Anwendung einer Funktion auf einen „rationalisierten“ Ausdruck
`match`	:	Mustererkennung in Ausdrücken
`nops`	:	Anzahl der Operanden eines Objektes
`numer`	:	Zähler eines rationalen Ausdrucks
`op`	:	Operandenfunktion
`protect`	:	Setzen eines Schreibschutzes
`radsimp`	:	Vereinfachung von Wurzelausdrücken
`rationalize`	:	Konvertierung beliebiger Ausdrücke in rationale Ausdrücke
`rewrite`	:	Umschreiben von Ausdrücken
`rhs`	:	rechte Seite eines Ausdrucks
`select`	:	Auswählen von Operanden nach Eigenschaften
`simplify`	:	Vereinfacher
`Simplify`	:	konfigurierbarer Vereinfacher
`slot`	:	Setzen oder Lesen eines Attributs eines Domains oder einer Funktionsumgebung
`split`	:	Zerlegen eines Objektes nach Eigenschaften
`subs`	:	Substitution
`subsex`	:	Substitution komplexerer Ausdrücke
`subsop`	:	Substitution von Operanden
`sysorder`	:	Information über die Anordnung von Objekten im MuPAD-Kern
`testeq`	:	Test mathematischer Äquivalenz
`testtype`	:	Typenvergleich
`traperror`	:	Abfangen eines Fehlers
`type`	:	Typ eines Objektes
`unalias`	:	Löschen einer Abkürzung

`unassume`	:	Löschen von Annahmen über Eigenschaften
`unexport`	:	Ausladen von Bibliotheken
`unprotect`	:	Aufheben des Schreibschutzes für Bezeichner
`zip`	:	Verknüpfung von Listen und Matrizen

Weitere Funktionen zur Behandlung von Daten-Objekten sind in den diversen Bibliotheken (Abschnitt 17) zu finden.

11 Manipulation von Zeichenketten

`_concat`, `.`	:	Aneinanderfügen mehrerer Zeichenketten
`expr2text`	:	Konvertierung eines Ausdrucks in eine Zeichenkette
`ftextinput`	:	Texteingabe aus einer Datei
`int2text`	:	Konvertierung einer ganzen Zahl in eine Zeichenkette
`length`	:	Länge einer Zeichenkette
`revert`	:	Umkehrfunktion
`strmatch`	:	Mustererkennung
`substring`	:	Extrahieren von Teilen
`tbl2text`	:	Konvertierung einer Tabelle in eine Zeichenkette
`text2expr`	:	Konvertierung einer Zeichenkette in einen Ausdruck
`text2int`	:	Konvertierung einer Zeichenkette in eine ganze Zahl
`text2list`	:	Konvertierung einer Zeichenkette in eine Liste
`text2tbl`	:	Konvertierung einer Zeichenkette in eine Tabelle
`textinput`	:	Interaktive Eingabe von Zeichenketten

Weitere Funktionen sind in der Bibliothek `stringlib` installiert.

12 Funktionen für Ein- und Ausgabe, Graphik

`doprint`	:	Ausgabe großer Matrizen
`error`	:	Fehlererzeugung und Ausgabe einer Fehlermeldung
`fclose`	:	Schließen einer Datei
`finput`	:	Lesen einer Datei
`fopen`	:	Öffnen einer Datei
`fprint`	:	Beschreiben einer Datei
`fread`	:	Lesen einer Datei
`ftextinput`	:	Texteingabe aus einer Datei
`import::readdata`	:	Einlesen formatierter ASCII-Daten
`input`	:	Interaktive Zuweisung
`plot`	:	Zeichnen graphischer Objekte
`plotfunc2d`	:	Zeichnen des Graphen einer einstelligen Funktion
`plotfunc3d`	:	Zeichnen des Graphen einer zweistelligen Funktion
`print`	:	Bildschirmausgabe
`protocol`	:	Eröffnen und Schließen eines Sitzungsprotokolls
`read`	:	Einlesen einer Datei
`readbytes`	:	Lesen aus einer Binärdatei

`setuserinfo`	:	Setzen des „Informationslevels“ für `userinfo`
`textinput`	:	Interaktive Eingabe von Zeichenketten
`userinfo`	:	Ausgaben von Informationen beim Ablauf eines
	:	Algorithmus
`warning`	:	Ausgabe einer Warnung
`write`	:	Abspeichern von Werten in Dateien
`writebytes`	:	Beschreiben einer Binärdatei

Eine Übersicht über die Routinen zum Lesen/Beschreiben einer Datei kann mittels `?fileIO` angefordert werden. Zur Graphik beachte man auch die Dokumentation der `plot`–Bibliothek (`?plot`).

13 Steuerung der Auswertung

`bool`	:	Auswertung logischer Ausdrücke zu `TRUE` oder `FALSE`
`context`	:	Auswertung in einem Kontext
`eval`	:	erzwungene Auswertung
`evalassign`	:	Zuweisung mit Auswertung der linken Seite
`freeze`	:	Deaktivierung einer Funktion
`hold`	:	Verzögerung der Auswertung
`indexval`	:	indizierter Zugriff ohne Auswertung
`last`, `%`	:	Zugriff auf bereits berechnete Ergebnisse
`level`	:	Auswertung mit angegebener Tiefe
`unfreeze`	:	Reaktivierung einer Funktion
`val`	:	Wert eines Bezeichners (entspricht `level(.,1)`, aber ohne dass der interne Vereinfacher anspringt)

14 Funktionen in Prozedurdefinitionen

`_procdef`	:	Prozedurdefinition
`args`	:	Zugriff auf Prozedurargumente
`error`	:	Erzeugen eines Fehlers
`return`	:	Rückgabe von Werten
`testargs`	:	Kontrolle des Testens von Argumenten
`warning`	:	Ausgabe einer Warnung

15 Mathematische Funktionen und Algorithmen

`abs`	:	Absolutbetrag
`arg`	:	Polarwinkel einer komplexen Zahl
`asympt`	:	asymptotische Reihenentwicklung
`binomial`	:	Binomialausdruck $\binom{m}{n}$
`ceil`	:	kleinste ganze Zahl $\geq x$
`conjugate`	:	komplexe Konjugation
`contfrac`	:	Kettenbruchentwicklung

`D` und `'`	:	Differentialoperator
`diff`	:	Differentiation von Ausdrücken
`discont`	:	Unstetigkeitsstellen einer Funktion
`div`	:	Division „modulo“
`factor`	:	Faktorisierung von Polynomen, Ausdrücken und ganzen Zahlen
`float`	:	Konvertierung in Gleitpunktzahlen
`floor`	:	Gaußklammer: größte ganze Zahl $\leq x$
`frac`	:	Nachkommastellen einer Zahl
`frandom`	:	Zufallsgenerator für Gleitpunktzahlen
`gcd`	:	größter gemeinsamer Teiler
`gcdex`	:	größter gemeinsamer Teiler (erweiterter Euklidischer Algorithmus)
`ifactor`	:	Primfaktorzerlegung
`igcd`	:	größter gemeinsamer Teiler ganzer Zahlen
`igcdex`	:	größter gemeinsamer Teiler ganzer Zahlen (erweitert)
`ilcm`	:	kleinstes gemeinsames Vielfaches ganzer Zahlen
`Im`	:	Imaginärteil
`int`	:	Integration
`interpolate`	:	Polynominterpolation
`intersect`	:	Schnitt zweier Mengen
`interval`	:	convert to a floating point interval
`isprime`	:	Primzahltest
`isqrt`	:	ganzzahlige Approximation der Wurzel einer natürlichen Zahl
`iszero`	:	Test auf 0
`ithprime`	:	i-te Primzahl
`lcm`	:	kleinstes gemeinsames Vielfaches
`limit`	:	Grenzwertberechnung
`linsolve`	:	Lösung linearer Gleichungen
`lllint`	:	reduzierte Gitterbasis
`max`	:	Maximum
`min`	:	Minimum
`minus`	:	Differenz zweier Mengen
`mod`, `modp`, `mods`	:	Rest „modulo“
`nextprime`	:	nächste Primzahl $\geq x$
`norm`	:	Norm von Polynomen, Vektoren und Matrizen
`normal`	:	Normalisierung rationaler Ausdrücke
`not`	:	logische Verneinung
`or`	:	logisches „und“
`pade`	:	Padé-Approximation
`partfrac`	:	Partialbruchzerlegung
`powermod`	:	Potenz „modulo“
`product`	:	Erzeuger eines Produktes
`random`	:	Zufallszahlengenerator
`Re`	:	Realteil
`rectform`	:	Zerlegung nach Real- und Imaginärteil

`revert`	:	Umkehrfunktion (für Reihen, Zeichenketten und Listen)
`round`	:	Rundung
`series`	:	Reihenentwickler
`sign`	:	(komplexes) Vorzeichen
`signIm`	:	Vorzeichen des Imaginärteils
`solve`	:	Gleichungslöser
`sort`	:	Sortieren von Listen
`sum`	:	Erzeuger einer Summe
`taylor`	:	Taylor-Entwicklung
`trunc`	:	Abschneiden von Nachkommastellen
`union`	:	Vereinigung zweier Mengen

Zahlreiche weitere Funktionen (Algorithmen) zu speziellen mathematischen Themenkreisen sind in den Bibliotheken (Abschnitt 17) installiert.

16 Lösen von Gleichungen

Neben dem universellen `solve`-Kommando zum Lösen algebraischer Gleichungen, von Differentialgleichungen oder von Rekurrenzgleichungen stellt MuPAD eine Reihe spezialisierter Routinen zur Lösung spezieller Typen von Gleichungen zur Verfügung.

Gleichungstyp	verfügbare Löser

System linearer Gleichungen	`solve`
	`linsolve`
	`linalg::matlinsolve`
	`linalg::matlinsolveLU`
	`linalg::vandermondeSolve`
	`numeric::linsolve`
	`numeric::matlinsolve`
	`numeric::solve`

univariate polynomiale Gleichung	`solve`
	`polylib::realroots`
	`numeric::solve`
	`numeric::polyroots`

bivariate polynomiale Gleichung	`solve`

System polynomialer Gleichungen	`solve`
	`numeric::solve`
	`numeric::polysysroots`

beliebige univariate Gleichung	`solve`
	`numeric::solve`
	`numeric::realroot`
	`numeric::realroots`

System beliebiger Gleichungen	`solve`
	`numeric::solve`
	`numeric::fsolve`

Ungleichungen	`solve`

System gewöhnlicher Differentialgleichungen	`solve`
	`numeric::odesolve`
	`numeric::odesolve2`
	`numeric::odesolveGeometric`

Rekurrenzgleichung	`solve`
partielle Differentialgleichungen	`detools::pdesolve`

Kongruenz	`numlib::lincongruence`
	`numlib::mroots`
	`numlib::msqrts`

Ein Überblick über die zur Verfügung stehenden Löser kann mittels `?solvers` angefordert werden.

17 Die Bibliotheken

Folgende Bibliotheken und Module in MuPAD Version 3.1 enthalten eine große Anzahl weiterer Funktionen und Algorithmen zu speziellen mathematischen Themenkreisen. Die Bibliotheken befinden sich in ständiger Entwicklung: spätere MuPAD-Versionen stellen zusätzliche Bibliotheken und Funktionen bereit. Eine Übersicht über die aktuell installierten Funktionen erhält man jeweils mit `info` oder `?`. Hilfe zu den einzelnen Algorithmen ist jeweils in der Form `?Bibliotheksname::Funktionsname` anzufordern. Für Module erhält man mittels `modulename::doc()` Informationen.

`adt`	:	abstrakte Datentypen
`Ax`	:	Erzeuger von Axiomen
`Cat`	:	Erzeuger von Kategorien
`combinat`	:	Kombinatorik
`detools`	:	Differentialgleichungen
`Dom`	:	vorgefertigte Datenstrukturen: Körper, Ringe, Matrizen, etc.
`fp`	:	Funktionales Programmieren
`generate`	:	Erzeuger von C-, Fortran- und TeX-Code
`Graph`	:	Graphen
`groebner`	:	Berechnung mit Polynomidealen
`import`	:	Importieren externer Datenformate
`intlib`	:	Integration
`linalg`	:	lineare Algebra
`linopt`	:	lineare Optimierung
`listlib`	:	Manipulation von Listen
`matchlib`	:	Mustererkennung in Ausdrücken
`misc`	:	Diverses
`module`	:	Verwaltung von Modulen
`numeric`	:	numerische Algorithmen
`numlib`	:	Zahlentheorie
`orthpoly`	:	Orthogonalpolynome
`output`	:	Ausgabe von Objekten

`plot`	:	Zeichenroutinen
`polylib`	:	Algorithmen für Polynome
`Pref`	:	Systemvoreinstellungen
`prog`	:	Programmierung und Fehlersuche
`property`	:	Eigenschaften von Bezeichnern
`RGB`	:	Farbnamen (Rot-Grün-Blau-Werte)
`solvelib`	:	Lösen von Gleichungen
`stats`	:	Statistikfunktionen
`stdlib`	:	Standardbibliothek
`stdmod`	:	Erweitertes Modulmanagement
`stringlib`	:	Manipulation von Zeichenketten
`student`	:	einige elementare Algorithmen
`transform`	:	Integraltransformationen
`Type`	:	Typenbezeichner
`util`	:	Modul mit nützlichen Funktionen

18 Kontrolle eigener Prozeduren

`assert`	:	Überprüfen von Prozedurinvarianten
`debug`	:	Debuggen einer Prozedur
`prog::check`	:	Teste Prozeduren und Domains auf globale Variablen etc.
`prog::profile`	:	Statistik über Laufzeiten etc.
`prog::trace`	:	Aufrufverfolgung von Prozeduren
`rtime`	:	Zeitangabe
`time`	:	Zeitangabe

Weitere Programmierhilfen sind in der Bibliothek `prog` zu finden.

19 Technisches

`builtin`	:	Funktionen des MuPAD-Kerns
`bytes`	:	Speicherverbrauch
`export`	:	Laden von Bibliotheken
`external`	:	Modulverwaltung
`getpid`	:	Prozessnummer von MuPAD
`loadproc`	:	Einladen von Prozeduren
`module`	:	loading dynamic modules
`operator`	:	Definition neuer Operatoren
`package`	:	Einladen benutzerdefinierter Bibliotheken
`patchlevel`	:	Information über MuPAD-Patches
`pathname`	:	Information über Pfadnamen
`quit`	:	Beenden einer MuPAD-Sitzung
`register`	:	Registrieren einer MuPAD-Kopie
`reset`	:	Re-Initialisierung einer MuPAD-Sitzung
`share`	:	Erzeugung eindeutiger Datenhaltung
`sysname`	:	Name des Betriebssystems

`system`	:	Ausführen eines Kommandos des Betriebssystems
`unexport`	:	Ausladen von Bibliotheken
`version`	:	Information über die Versionsnummer MuPADs