Maxima

無料数式処理ソフト

文：乙部厳己

平成 20 年 6 月 19 日

★★★ダウンロード★★★

`http://maxima.sourceforge.net/`
左側のメニュー［download］から［Sourceforge download page］へ移動。
`http://sourceforge.net/project/showfiles.php?group_id=4933`

★★★起動法★★★

端末版：`maxima`
X ウィンドウ版：`xmaxima &`
Windows では xmaxima または wxMaxima（使いやすい）のアイコンをクリックして起動。

★★★ヘルプ★★★

`describe(string)`	`string` に関する文書（ヘルプ）を表示。
`? string`	`describe(string)` に同じ。
`apropos(string)`	`string` を含むコマンドを表示。
`example(topic)`	`topic` の例を表示

★★★文★★★

すべての文は; または $ で終わる。

`......;`	結果表示
`......$`	結果非表示
`%`	直前の出力。`%o1` など。`%%` は `block` 等での直前の式の値。
`%th(i)`	i 個前の出力。
`/* ... */`	コメント
`_`	直前の入力。`%i1` など。`__` は現在評価中の式

★★★定義★★★

`:`	代入（`a:3` は `a` を 3 にする）
`:=`	関数定義（`f(x):=sin(x)`）
`::`	ポインタ代入（左辺の値に右辺を代入する）
`fundef(func)`	`func` の定義表示
`kill(v)`	`v` の定義を消去する
`remfunction(f1,...,fn)`	関数定義を消去する。`remfunction(all)` ですべて消去する。
`remvalue(v1,...,vn)`	変数定義を消去する。`remvalue(all)` ですべて消去する。
`assume(p)`	仮定をおく。`assume(m>0)`

★★★演算★★★

`+・-・*・/`	足し算・引き算・かけ算・割り算
`**` または `^`	冪乗（`2^3` $= 2^3$）
`.`	（行列の）非可換な積。冪乗は `^^`。
`=`	方程式定義（`x^2+2*x+1=0`）
`!・!!`	階乗・二重階乗
`'`	式を評価しない（`'(F(x))`）

★★★数学定数★★★

`inf`	$+\infty$
`infinity`	複素無限大
`minf`	$-\infty$
`%pi`	$\pi = 3.141592\ldots$
`%e`	$e = \lim_{n\to\infty}(1+1/n)^n = 2.71828\ldots$
`%i`	$\sqrt{-1}$
`%gamma`	$\lim_{n\to\infty}(\sum_{k=1}^{n}\frac{1}{k} - \log n) = 0.5772\ldots$
`%phi`	$(1+\sqrt{5})/2 = 1.6180\ldots$

★★★関数★★★

`sin(x)・cos(x)・tan(x)`	sin, cos, tan
`sec(x)・csc(x)・cot(x)`	sec, csc, cot
`asin(x)`	逆三角関数 ($\sin^{-1} x$)。`acos`, `acsc` なども同様。
`sinh(x)`	双曲線関数。`coth`, `asinh` なども同様。
`atan2(y,x)`	$-\pi < \mathrm{atan}\,\frac{y}{x} < \pi$
`exp(x)`	e^x
`log(x)`	$\log_e x$
`plog(x)`	log の主分枝（$-\pi <$ `CARG(x)` $\leq \pi$）
`abs(x)`	絶対値
`cabs(x)`	複素絶対値
`ceiling(x)`	x 以上の最小の整数
`floor(x)`	x 以下の最大の整数
`round(x)`	x に最も近い整数
`max(x1,x2,...)`	最大値
`min(x1,x2,...)`	最小値
`signum(x)`	$\mathrm{sgn}(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases}$

sqrt(x)	$\sqrt{x}$
factorial(n)	$n!$
binomial(x,y)	二項関数。x と y が整数なら $\frac{x\cdot(x-1)\cdots(x-y+1)}{y!}$
bern(n)・burn(n)	n 番目のベルヌイ数（n が 105 より大きいと burn の方がいい）
bernpoly(x,n)	第 n ベルヌイ多項式
zeta(n)	n が負整数, 0, 1, 正偶数ならリーマンゼータ
bfzeta(s,n)	n 桁の bigfloat で s でのリーマンゼータ関数の値を返す。
bfhzeta(s,h,n)	n 桁のフルビツゼータ関数の値を返す。
fib(n)	第 n フィボナッチ数
airy_*(x)	*=ai・bi・dai・dbi でそれぞれエアリー関数 $\mathrm{Ai}(x)$・$\mathrm{Bi}(x)$ およびそれらの微分
bessel_?(n,z)	?=j・y・i・k で第一種・第二種・修正第一種・修正第二種ベッセル関数。
beta(x)	ベータ関数 $\frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$
gamma(x)	ガンマ関数。x が整数なら $\Gamma(x)=(x-1)!$
euler(n)	n が自然数なら第 n オイラー数
jacobi_??(u,m)	ヤコビの楕円関数 sn・cn・dn・ns・sc・sd・nc・cs・cd・nd・ds・dc
inverse_jacobi_??(u,m)	ヤコビの楕円関数 sn・cn・dn・ns・sc・sd・nc・cs・cd・nd・ds・dc の逆
elliptic_??(*)	f(phi,m)=$\int_0^\phi 1/\sqrt{(1-m\sin^2 x)}\,dx$（第一種不完全楕円積分）, e(phi,m)=$\int_0^\phi \sqrt{1-m\sin^2 x}\,dx$（第二種不完全楕円積分）, eu(u,m)=$\int_0^{\mathrm{sn(u,m)}} \sqrt{1-mt^2}/\sqrt{(1-t^2)}\,dt$, pi(n,phi,m)=$\int_0^\phi 1/((1-n\sin^2 x)\sqrt{1-m\sin^2 x})\,dx$, k(m)=$\int_0^{\pi/2} 1/\sqrt{1-m\sin^2 x}\,dx$, ec(m)=$\int_0^{\pi/2} \sqrt{1-m\sin^2 x}\,dx$
erf(x)	$\frac{2}{\sqrt{\pi}}\int_{-\infty}^{x}\exp\{-y^2\}\,dy$
carg(z)	$\arg z$
polarform(z)	$Re^{i\theta}$, polarform$(1+\sqrt{-1})=\sqrt{2}e^{\sqrt{-1}\pi/4}$
random(n)	0 から $n-1$ までの乱数。random() は -2^{29} ～$2^{29}-1$。random(false) でリスタート。
inv_mode(n,m)	$n \bmod m$ の逆を計算する。
jacobi(p,q)	ヤコビ記号 $\left(\frac{p}{q}\right)$ を計算する。
prev_prime(n)	n より小さな最大の素数。
next_prime(n)	n より大きい最小の素数。
power_mod(a,n,m)	$a^n \bmod m$ を求める。$a,n\in\mathbb{Z}, m\in\mathbb{N}$
primep(n)	n が素数かどうかを判定する。$n<34155071728321$ では決定論的 Miller–Rabin 判定を行う。それより大きければ、Miller–Rabin の疑似素数判定と Luvas の疑似素数判定を行う。その確率は標準で 10^{-15} よりも小さい。
qunit(n)	実二次体 $\mathbb{Q}(n)$ で基本単数を Pell 方程式を解いて求める。
totient(n)	n 以下の n と互いに素な数の個数

★★★式の評価★★★

ev(exp, arg1, arg2, ...)
exp を arg1, arg2, ... に基づいて評価する。arg は exp 内の未評価状態の関数（integrate や sum などが多い）を指定する（nouns 指定でもよい）ほか、次の通り。

simp	exp を簡約化
noeval	ev の評価を抑制
nouns	評価されていない「名詞形」の関数を評価
expand	展開
expand(m,n)	maxposex と maxnegex を m と n にして展開
detout	逆行列計算で行列式で各係数を割らず、行列の前に置く
diff	すべての微分を実行
derivlist(x,y,...)	指定された変数に関してのみ微分
float	分数を浮動小数に変換
numer	いくつかの数学関数を浮動小数で評価する

★★★展開★★★

expand(exp)	展開する（多項式には ratexpand がよい）
ratcoef(exp,x,n)	exp の x^n の係数を表示する。
expandwrt(exp,x1,x2,...)	変数 x1, x2, ... に関して展開する
distrib(exp)	分配法則を一度だけ適用
partfrac(exp,var)	exp を変数 var に関する部分分数に展開
ratexpand(exp)	和や積、約分などを考慮しながら展開する。
trigexpand(exp)	三角関数を展開

★★★簡約★★★

radcan(exp)	指数・対数・根号などの含まれる exp を簡約する。
ratsimp(exp)	有理式を簡約化する。
fullratsimp(exp)	ratsimp を式が変わらなくなるまで繰り返す。
scsimp(exp)	逐次比較簡約化を行う。
trigreduce(exp)	三角（双曲）関数の「積を和に直す」
trigsimp(exp)	三角（双曲）関数の 1 になる部分を消去する。
factcomb(exp)	$(n+1)n!=(n+1)!$
makegamma(exp)	binomial, factorial, beta などを Γ にする。
xthru(exp)	通分して整理する。
factor(exp)	整数上既約にする（因数分解）
gfactor(exp)	ガウス整数上規約にする
ifactors(n)	正の整数 n を素因数分解する。9973 までの素数で割り、Pollard の ρ 法、楕円関数法を適用する。
factorout(exp,x,...)	変数を含まない項を括り出す
subst(a,s,exp)	exp の s に a を代入する。

★★★多項式★★★

divide(p1,p2,x)	多項式 p_1 を変数 x に関して多項式 p_2 で割った商と余りを求める。
mod(p,m)	多項式 p の m を法としたモジュラー表現を求める。
polydecomp(p,x)	多項式 p を変数 x に関する多項式の合成に分解する。
gcd(p1,p2,x)	p_1 と p_2 の x に関する最大公約数を求める。
lcm(p1,p2,x)	p_1 と p_2 の x に関する最小公倍数を求める。load(functs) が必要。

★★★グラフ★★★

contour_plot(exp,x_range,y_range,options,...)	2 変数関数 exp の等高線を x_range, y_range の範囲でかく。各範囲は [x,-1,1] のように指定する。options は plot3d と同じ。
plot2d(exp,x_range,options,...)	2 次元で exp のグラフをかく。x_range は [x,-1,1] のように指定する。exp は式であるか、[discrete,[x1,...,xn],[y1,...,yn]] または [discrete,[[x1,y1],...,[xn,yn]]] の形で離散の点を指定するか、[parametric,x_exp,y_exp,t_range] の形で $\{(x(t),y(t))\}$, $t \in$ t_range と 2 次元平面上の点を指定する。t_range は [t,0,%pi] のように指定する。複数のグラフを同時にかくときには exp を [exp1,...,expn] の形で指定する。
plot3d(exp,x_range,y_range,options,...)	3 次元で exp のグラフをかく。指定は plot2d と同様。
make_transform(v,fx,fy,fz)	transform_xy で使用される変換関数を定義する。たとえば make_transform([r,th,z],r*cos(th),r*sin(th),z) とすれば極座標表示によるグラフ描画を可能にする。

●共通オプション

オプションは、引数が必要なときには [ylabel,xxx] のように指定する。そうでないときには [logx] のように指定する。全体に適用するオプションは plot_option に保存し、set_plot_option で変更できる。

plot_format	グラフのインターフェース。標準は gnuplot（Windows）または gnuplot_pipes（それ以外）。その他は mgnuplot（Tk ベースの gnuplot ラッパ）と openmath（Xmaxima に同梱）
y	縦方向の範囲。[y,-3,3] のように指定。
discrete	離散のグラフ
plot_realpart	true のときには複素数値関数の実部のみ描画

● plot2d のオプション

xlabel・ylabel	x, y 軸のラベル
logx・logy	各軸を対数スケールにする
legend	グラフの題名
ntics	グラフをかくときのサンプル点の個数（plot2d）。標準は 10
style	lines, points, linespoints, dots, impulses。
lines(style)	線の属性を指定するには [style,[lines,2,3]] のように線の太さと色（1 から 7）を指定する。Windows の場合には 1: 青, 2: 赤, 3: マジェンタ, 4: 青緑, 5: 茶, 6: 黄緑, 7: 紺。色は gnuplot_term ごとに異なる。
points(style)	点の属性を指定するには [style,[points,2,3,4]] のように点の半径、色（lines と同じ）、形状を指定する。形状は 1: ●, 2: ○, 3:+, 4:×, 5:∗, 6: ■, 7: □, 8: ▲, 9: △, 10: ▼, 11: ▽, 12: ◆, 13: ◇

● plot3d のオプション

grid	x および y 方向のグリッドの数。[grid,50,50] のように指定する。標準は [grid,30,30]
transform_xy	make_transform() の出力とすると、それで定義された座標変換を使用可能にする。make_transform([r,th,z], r*cos(th), r*sin(th), z)$ は polar_to_xy で最初から定義されており、[transform_xy,polar_to_xy] で利用可能になる。

● gnuplot のオプション

gnuplot で表示するときには専用オプションを指定できる。

gnuplot_term	dumb（アスキー文字で出力）, ps（gnuplot_out_file または maxplot.ps ファイルに PS ファイルを出力）, その他 [gnuplot_term,"png size 1000,1000"] のように指定
gnuplot_pm3d	true とすると gnuplot 3.7 以降であれば PM3D（きれいに 3D グラフを表示）モードを使用。
gnuplot_preamble	グラフを描画する前に gnuplot が実行するコマンドを指定する。例：[gnuplot_preamble,"set log y"]
gnuplot_curve_titles	[gnuplot_curve_titles,["title, 'first'", "title, 'second'"]] の用にしてタイトルを指定
gnuplot_curve_styles	[gnuplot_curve_styles,["with lines 5", "with lines 2"]] のようにスタイルを指定
gnuplot_xxx_term_command	xxx は default, dumb, ps, [gnuplot_ps_term_command,"set term postscript eps enhanced color solid 24"] のように使用。

○例

- plot2d (sin(x), [x, -5, 5])$

- plot2d (sec(x), [x, -2, 2], [y, -20, 20], [nticks , 200])$
- F(x) := x^2 $
  G(x) := if x < 0 then x^4 - 1 else 1 - x^5 $
  plot2d ([F, G], [u, -1, 1], [y, -1.5, 1.5])$
- plot2d ([parametric, cos(t), sin(t), [t,-%pi,%pi] , [nticks,80]], [x, -4/3, 4/3])$
- plot2d ([x^3+2, [parametric, cos(t), sin(t), [t, -5, 5], [nticks, 80]]], [x, -3, 3])$
- plot3d (2^(-u^2 + v^2), [u, -3, 3], [v, -2, 2], [plot_format, openmath]);
- メビウスの帯
  plot3d ([cos(x)*(3 + y*cos(x/2)), sin(x)*(3 + y*cos (x/2)), y*sin(x/2)], [x, -%pi, %pi], [y, -1, 1], [' grid, 50, 15]);
- $z^{1/3}$ のリーマン面（実部）
  plot3d (r^(1/3)*cos(th/3), [r, 0, 1], [th, 0, 6*%pi], ['grid, 12, 80], ['transform_xy, polar_to_xy]);
- クラインの壺
  exp1: 5*cos(x)*(cos(x/2)*cos(y) + sin(x/2)*sin(2*y) + 3.0) - 10.0$
  exp2: -5*sin(x)*(cos(x/2)*cos(y) + sin(x/2)*sin(2*y) + 3.0)$
  exp3: 5*(-sin(x/2)*cos(y) + cos(x/2)*sin(2*y))$
  plot3d ([exp1, exp2, exp3], [x, -%pi, %pi], [y, -%p i, %pi], ['grid, 40, 40]);
- トーラス
  exp1: cos(y)*(10.0+6*cos(x))$
  exp2: sin(y)*(10.0+6*cos(x))$
  exp3: -6*sin(x)$
  plot3d ([exp1, exp2, exp3], [x, 0, 2*%pi], [y, 0, 2 *%pi], ['grid, 40, 40]);

★★★級数★★★

sum(exp,ind,low, high)	$\sum_{\mathrm{ind}=\mathrm{low}}^{\mathrm{high}} \mathrm{exp}$
nusum(exp,var,lo w,high)	「不定和」 $\sum_{j=\mathrm{low}}^{\mathrm{high(var)}} \mathrm{exp}(j)$
unsum(f,i)	$f(i) - f(i-1)$
product(exp,ind, low,high)	$\prod_{\mathrm{ind}=\mathrm{low}}^{\mathrm{high}} \mathrm{exp}$
powerseries(exp, var,p)	exp の点 p（inf でもよい）での変数 var による冪級数展開（できなければ taylor でできることもある）。
taylor(exp,var,p ,pow)	exp の点 p での変数 var に関する pow 次までの Taylor（または Laurent）展開。
taylor(exp,[v1,p 1,o1],[v2,p2,o2] ,...)	v_i に関する p_i での o_i 次展開
pade(ts,nd,dd)	与えられた Taylor/Laurent 級数 ts を持つ分子・分母の次元が nd・dd の有理関数をリストで返す（パデ近似）。
cf(exp)	exp を（正則）連分数にする。
cfdisrep(l)	リスト l を連分数表記にする。

○例

- taylor (sqrt (sin(x) + a*x + 1), x, 0, 3);
- taylor (1/(cos(x) - sec(x))^3, x, 0, 5);
- g(p) := p*4^n/binomial(2*n,n);
  g(n^4);
  nusum (%, n, 0, n);
  unsum (%, n);

★★★線形代数★★★

●多重配列

array(name,d1,.. .,dn)	n 重（$n \leq 5$）配列を作る。第 i 次元の添え字は 0 から d_i まで。
arrayinfo(A)	配列 A の情報を表示。
listarray(A)	配列 A の各要素を表示する。

array(a, 5) は a[0] から a[5] までの 6 次元ベクトルを作る。array(a, 2, 2) は a[0,0] から a[2,2] の 9 次元からなる行列を作る。a[0,0] : 1 のように値を設定できる。a[i,j] := 1/(i+ j+1) のような定義も可能。

●行列生成

matrix(r1,r2,... ,rn)	行列。各行 r_i は [1,2,3,4] のようなリスト表現。
genmatrix(A,i2,j 2,i1,j1)	配列 A から行列を生成する。A[i1,j1] が行列の左上、A[i2,j2] は右下の要素。$j_1 = i_1$ のときには j_1 は省略可能。$i_1 = j_1 = 1$ のときには i_1 も省略可能。
entermatrix(m,n)	$m \times n$ 行列を対話的に作る。
copymatrix(m)	行列 m のコピーを作る。
diagmatrix(n,exp)	$n \times n$ 行列で対角成分がすべて exp
ematrix(m,n,exp, i,j)	$m \times n$ 行列で (i,j) 成分が exp のほかはすべて 0
zeromatrix(m,n)	成分がすべて 0 の $m \times n$ 行列

genmatrix においては array(A,2,2); で 2 重配列 A を作り、A[i ,j] := 1/(i+j+1); と各項を定義し、genmatrix(A,2,2); とすると（省略すると A[1,1] から始まるので） $\begin{pmatrix} \mathrm{A[1,1]} & \mathrm{A[1,2]} \\ \mathrm{A[2,1]} & \mathrm{A[2,2]} \end{pmatrix}$ という行列、つまり $\begin{pmatrix} 1/3 & 1/4 \\ 1/4 & 1/5 \end{pmatrix}$ ができる。

●特性量

rank(m)	m のランク（階数）
mattrace(m)	m のトレース（跡）
determinant(m)	m の行列式
permanent(m)	m のパーマネント
charpoly(m,x)	m の特性多項式（変数 x）
eigenvalues(m)	m の固有値のリストをつくる。第一リストは固有値、第二リストは重複度。
eigenvectors(m)	m の固有ベクトルのリスト。第一リストが固有値、第二リストが対応固有ベクトル。

●行列操作

setelmx(exp,i,j,m)	行列 m の (i,j) 成分を exp にする。
row(m,i)	行列 m の第 i 行を行列として返す
col(m,i)	行列 m の第 j 列を行列として返す
addrow(m,l1,l2,...)	m にリストまたは行列 $l1, \ldots$ を行として追加
addcol(m,l1,l2,...)	m にリストまたは行列 $l1, \ldots$ を列として追加
submatrix(i1,i2,...,m,j1,j2,...)	行列 m から $i_1, i_2, \ldots$ 行と $j_1, j_2, \ldots$ 列を取り除いた行列
adjoint(m)	m の共役転置行列
invert(m)	m の逆行列
transpose(m)	m の転置行列
triangularize(m)	m を三角化（m は正方でなくてよい）
matrixmap(f,m)	m の各成分に関数 f を適用

●固有値操作パッケージ

load(eigen) とすると、より使いやすい固有値操作ができる。

conjugate(m)	複素共役
columnvector(l)	リスト l から縦ベクトルを作る。（関数出力をベクトル化するときに使う）
gramschmidt(m)	m（リストのリスト）をグラム・シュミット直交化する。
innerproduct(l1,l2)	リスト l_1 と l_2 の内積をとる
similaritytransform(m)	m の左行列と右行列を生成する（これらを掛けると対角行列になる）
uniteigenvectors(m)	単位固有ベクトルを生成する。
unitvector(l)	リスト l を単位ベクトルにする。

●線形方程式系等パッケージ

load(affine) とすると以下の機能が利用できる。

fast_linsolve([exp1,...,expn],[v1,...,vn])	linsolve より高速に線形方程式系を解くことができる。
grobner_basis([exp1,...,expm])	方程式 exp1,...,expm に対するグレブナー基底を求める。
set_up_dot_simplifications(eqns)	非可換変数の多項式方程式を指定する。
dotsimp(f)	f が方程式で生成されるイデアルに含まれていれば 0
mono([x1,...,xn],n)	現在の dotsimp に関する独立な単項式のリストを返す。

★★★微積分★★★

limit(exp,var,val,dir)	var→val の dir 方向の極限を求める。dir は plus または minus または省略。
diff(exp,v1,n1,v2,n2,...)	exp を変数 v_1 で n_1 回、v_2 で n_2 回...微分する。
gradef(f(v1,v2,...),exp1,exp2,...)	$\partial f/\partial v_1$ = exp1, $\partial f/\partial v_2$ = exp2 となる f を定義する。
depends(fl1,vl1,fl2,vl2,...)	diff で使われる依存関係を指示する。depends([f,g],[x,y],[r,s],[u,v,w],p,t) は $f(x,y), g(x,y), r(u,v,w), s(u,v,w), p(t)$ という関数であることを表す。diff(f,x) = 0 でも depends(f,x) の後では $\partial f/\partial x$ となる。
integrate(exp,var)	exp の var に関する原始関数
integrate(exp,var,low,high)	$\int_{\mathrm{low}}^{\mathrm{high}} \mathrm{exp}(\mathrm{var}) d\mathrm{var}$
ldefint(exp,var,ll,ul)	ll, ul は limit で求める。
changevar(exp,f(x,y),y,x)	exp 内の x に関する積分(和)を $f(x,y)=0$ によって y に変換する。
atvalue(exp,x=a,c)	exp の点 $x = a$ での値を c とする。atvalue('diff(f(x,y),x),x=0,1+y) など。点を指定する式が複数あるときは [x1=a1,x2=a2,...] と指定する。
at(exp,eq)	exp をその変数が方程式 eq に従うとして評価する。複数指定するときは [eq1,eq2,...] とする。
laplace(exp,ov,lv)	ov 変数に関する exp を lv にラプラス変換する。exp は exp, log, 三角関数, erf を含んでいてよい。
specint(exp(-s*t)*expr,t)	t に関して expr のラプラス変換を求める。expr は特殊関数を含んでいてよい。
ilt(exp,lv,ov)	lv に関する多項式の比の exp を ov 変数の式に逆ラプラス変換する。
residue(exp,z,z0)	exp の z に関する、z_0 での留数。

●数値積分

load(romberg) で Romberg 法による数値積分ができる。

romberg(exp,x,a,b)	exp の変数 x に関して a から b まで積分する。

●漸化式

load(solve_req) で多項式係数の線形漸化式を解くことができる。

solve_rec(eq,v)	多項式係数を持つ項 v に関する漸化式 eq を解く。
solve_rec(eq,v,ini1,...)	多項式係数を持つ項 v に関する漸化式 eq を解く。ini1,... で初期値を設定することもできる。

- rec: 2*n*(n+1)*x[n] - (n^2+3*n-2)*x[n+1] + (n-1)*x[n+2];
  solve_rec(rec, x[n], x[1]=1, x[3]=3);

★★★フーリエ解析★★★

fft(real_array,imaginary_array)	高速離散フーリエ変換。load("fft") が必要。
ift(real_array,imaginary_array)	高速逆離散フーリエ変換。load("fft") が必要。
fourier(f,x,p)	$f(x)$ の $[-p,p]$ でのフーリエ係数を求める。load("fourie") が必要。
foursimp(l)	$\sin n\pi = 0$ および $\cos n\pi = (-1)^n$ を用いて簡約化。
fourexpand(l,x,p,limit)	リスト l に与えられた limit（inf 可）までのフーリエ係数から $[-p,p]$ でフーリエ級数を作る。
fourcos(f,x,p)・foursin(f,x,p)	$[0,p]$ 上の f のフーリエ余弦・正弦展開を返す。
totalfourier(f,x,p)	fourexpand(foursimp(fourier(f,x,p)),x,p,'inf) を返す。
fourint(f,x)	$f(x)$ の $(-\infty,\infty)$ 上のフーリエ変換を返す。
fourintcos(f,x)・fourintsin(f,x)	$f(x)$ の $[0,\infty)$ 上でのフーリエ余弦・正弦変換を返す。

★★★方程式★★★

solve(exp,var)	exp を var で解いた解のリストを返す。exp が等式でないなら $=0$ と仮定。var は f(x) などでもよく、exp が 1 変数なら省略可。
solve([eq1,eq2,...],[v1,v2,...])	n 連立方程式を解く。未知数 v_i の個数が方程式数に等しければ省略できる。
funcsolve(eq,f(x))	方程式 eq を満たす有理関数 $f(x)$ が存在すればそれを返す。f について線形でなければならない（$(n+1)f(n)+(n+3)f(n+1)/(n+1)=(n-1)/(n+2)$ などはよい）。
allroots(pol)	pol のすべての実根・複素根を求める。
nroots(pol,low,high)	(low,high] に含まれる実根の数を求める。

○例

- solve (asin (cos (3*x))*(f(x) - 1), x);
- ev (solve (5^f(x) = 125, f(x)), solveradcan);
- solve (1 + a*x + x^3, x);

●多変数ニュートン法

load("mnewton") とするとニュートン法を用いることができる。

mnewton([eq1,...],[v1,v2,...],[x1,x2,...])	変数 $v_1, v_2, \ldots$ に関する方程式を $(x_1, x_2, \ldots)$ から出発してニュートン法で解く。

○例

- mnewton([2*a^a-5],[a],[1]);
- mnewton([x1+3*log(x1)-x2^2, 2*x1^2-x1*x2-5*x1+1], [x1, x2], [5, 5]);

★★★常微分方程式★★★

ode2(eq,dvar,ivar)	eqn で定められた従属変数（$y = f(x)$ としたときの y）dvar で独立変数（$y = f(x)$ としたときの x）ivar の 1 階・2 階の微分方程式を解析的に解く。1 階方程式では %c が、2 階方程式では %k1 と %k2 が積分定数。
ic1(sol,xval,yval)	1 階の微分方程式の初期値問題を解く。sol は ode2 などで求められた一般解。xval, yval はそれぞれ独立変数 x=x0 と従属変数 y=y0 で指定する。
ic2(sol,xval,yval,dval)	2 階微分方程式の初期値問題を解く。sol は一般解。xval, yval はそれぞれ独立変数 x=x0 と従属変数 y=y0 をこの形で指定する。dval は従属変数の（独立変数による）一階微分を diff(y,x)=dy0 の形で指定する。
bc2(sol,xv1,yv1,xv2,yv2)	2 階微分方程式の境界値問題を解く。sol は一般解。xv1 と xv2 は独立変数を x=x0 の形で指定する。yv1 と yv2 は従属変数の値を y=y0 の形で指定する。
desolve([eq1,eq2,...],[y1,y2,...])	線形常微分方程式系をラプラス変換の方法で解く。変数への依存はすべて明示する必要がある。

○例

- x^2*'diff(y,x) + 3*y*x = sin(x)/x;
  ode2(%,y,x);
  ic1(%,x=%pi,y=0);
- 'diff(y,x,2) + y*'diff(y,x)^3 = 0;
  sol2: ode2(%,y,x);
  ratsimp(ic2(sol2,x=0,y=0,'diff(y,x)=2));
  bc2(sol2,x=0,y=1,x=1,y=3);
- eqn1: 'diff(f(x),x,2) = sin(x) + 'diff(g(x),x);
  eqn2: 'diff(f(x),x) + x^2 - f(x) = 2*'diff(g(x),x,2);
  desolve([eqn1, eqn2], [f(x),g(x)]);

●常微分方程式拡張

load("contrib_ode") とすると、常微分方程式の拡張機能を用いることができる。（将来的には標準機能に組み込まれる予定。）

contrib_ode(eq,y,x)	微分方程式 eq の解を求める。
odelin(eq,y,x)	1 階および 2 階の線形斉次方程式の解を求める。

●ベクトル場表示

load("plotdf") とすると一階常微分方程式の定めるベクトル場を表示できる。積分曲線も表示できる。方程式は $dy/dx = F(x,y)$ の形か、自励系なら $dx/dt = G(x,y)$, $dy/dt = F(x,y)$ の形でなければならない。この F を plotdf に渡す。

plotdf(dydx,options...)	dydx を元にグラフを書く。dydx は y と x の関数。
plotdf(dvdu,[u,v],options...)	微分方程式が x, y 変数でない場合
plotdf([dxdt,dydt],options...)	自励系のグラフ
plotdf([dudt,dvdt],[u,v],options...)	従属変数が x, y でない場合

オプションは次の通り。

tstep	t（や x）を進める幅。初期値は 0.1
parameters	方程式中のパラメータを設定できる
sliders	パラメータの値をマウスで変化させられる。
nstep	進める回数。初期値は 100
direction	独立変数を進める方向。forward, backward, both
tinitial	t の初期値。初期値は 0
versus_t	2 つめのウィンドウを作り、x, y の t に関するグラフを出力
trajectory_at	積分曲線の xinitial と yinitial を定義（定義しなければ、クリックした点を通る積分曲線を引く）
x・y	表示する軸の範囲。初期値は -10 から 10

○例

- plotdf(exp(-x)+y, [trajectory_at,2,-0.1])
- plotdf(x-y^2, [xfun,"sqrt(x);-sqrt(x)"], [trajectory_at,-1,3], [direction,forward], [y,-5,5], [x,-4,16])
- plotdf([v,-k*z/m], [z,v], [parameters,"m=2,k=2"], [sliders,"m=1:5"], [trajectory_at,6,0])
- plotdf([y,-(k*x + c*y + b*x^3)/m], [parameters,"k=-1,m=1.0,c=0,b=1"], [sliders,"k=-2:2,m=-1:1"],[tstep,0.1])
- plotdf([w,-g*sin(a)/l - b*w/m/l], [a,w], [parameters,"g=9.8,l=0.5,m=0.3,b=0.05"], [trajectory_at,1.05,-9],[tstep,0.01], [a,-10,2], [w,-14,14], [direction,forward], [nsteps,300], [sliders,"m=0.1:1"], [versus_t,1])

●力学系

load("dynamics") とすると力学系の様子を図示することができる。

chaosgame([[x1,y1],...,[xm,ym]],[x0,y0],b,n,options,...)	初期点を (x_0, y_0) に印をつけ、そこからランダムに点 $(x_1, y_1), \ldots, (x_m, y_m)$ から 1 つ選び、そこへの線分上の点を選んだ点から距離 b と線分の長さを掛けただけの距離に選ぶということを n 回繰り返す（カオスゲーム）。
evolution(F,y0,n,options,...)	x 軸上の点を $0, 1, 2, \ldots, n$ として、縦軸に $y_n = F(y(n))$ で与えられる点をプロットする。
evolution2d([F,G],[u,v],[u0,v0],n,options,...)	$u_{n+1} = F(u_n, v_n)$, $v_{n+1} = G(u_n, v_n)$ で与えられる 2 次元力学系の軌道をプロットする。
orbits(F,y0,n1,n2,[x,x0,xf,xstep],options)	1 次元離散力学系の軌道の族をかく。時間発展は $F(y)$ で与えられるが、軌道族を表すパラメータ x も使用してよい。その場合は $[x_0, x_f]$ の範囲を xstep ずつ変化する。x の値が横軸に用いられ、縦軸には $y_{n_1+1}, \ldots, y_{n_1+n_2+1}$ の値が表示される。
ifs([r1,...,rm],[A1,...,Am],[[x1,y1],...,[xm,ym]],[x0,y0],n,options,...)	関数の反復。chaosgame と同じだが、線分を縮めるときに b 倍ではなく、選んだ点 (x_i, y_i) に応じて 2×2 行列 A_i を用いる。また、点の選び方は同確率ではなく、重みを r_i で変更できる。
staircase(F,y0,n,options...)	$y_{n+1} = F(y_n)$ で定義される 1 次元離散力学系の「階段図」をかく。
rk(ODE,var,ini,dom)	1 階の常微分方程式を 4 次の Runge–Kutta 法で解く。方程式系を解きたい場合には [ODE1,...,ODEm], [v1,...vm], [ini1,...,inim] とそれぞれリストで与える。独立変数の領域 domain は [t,0,10,0.1] のように変数、初期値、終了値、増加量をリストで指定する。

図を描画するときの options は plot2d と同じ。

○例：

- evolution(cos(y), 2, 11);
- staircase(cos(y), 1, 11, [y, 0, 1.2]);
- ロジスティック力学系の分岐図

```
orbits(x^2+a, 0, 50, 200, [a, -2, 0.25], [style, dots]);
```

- フラクタル図形の例

```
f: 0.6*x*(1+2*x)+0.8*y*(x-1)-y^2-0.9$
g: 0.1*x*(1-6*x+4*y)+0.1*y*(1+9*y)-0.4$
evolution2d([f,g], [x,y], [-0.5,0], 50000, [style,dots]);
evolution2d([f,g], [x,y], [-0.5,0], 300000, [x,-0.8,-0.6], [y,-0.4,-0.2], [style,dots]);
```

- シェルピンスキー・ガスケット

```
chaosgame([[0, 0], [1, 0], [0.5, sqrt(3)/2]], [0.1, 0.1], 1/2, 30000, [style, dots]);
```

- バーンスレイのシダ（Barnsley's fern）

```
a1: matrix([0.85,0.04],[-0.04,0.85])$
a2: matrix([0.2,-0.26],[0.23,0.22])$
a3: matrix([-0.15,0.28],[0.26,0.24])$
a4: matrix([0,0],[0,0.16])$
p1: [0,1.6]$
p2: [0,1.6]$
p3: [0,0.44]$
p4: [0,0]$
w: [85,92,99,100]$
ifs(w, [a1,a2,a3,a4], [p1,p2,p3,p4], [5,0], 50000, [style,dots]);
```

- rk(t-x^2, x, 1, [t,0,8,0.1]);
- rk([4-x^2-4*y^2,y^2-x^2+1], [x,y], [-1.25,0.75], [t,0,4,0.02]);

★★★確率分布★★★

load("distrib") が必要。確率変数 X の密度関数（density function）$f(x)$ に対して、分布関数（distribution function）$F(x) := \int_{-\infty}^{x} f(u)\,du = P(X \le x)$, 平均値（mean）$E[X] = \int_{-\infty}^{\infty} x f(x)\,dx$, 分散（variance）$V[X] = \int_{-\infty}^{\infty} (x - E[X])^2 f(x)\,dx$, 歪度（skewness coefficient）$SK[X] = \frac{1}{D[X]^3} \int_{-\infty}^{\infty} (x - E[X])^3 f(x)\,dx$, $D[X] = \sqrt{V[X]}$, 尖度（kurtosis coefficient）$KU[X] = \frac{1}{D[X]^4} \int_{-\infty}^{\infty} (x - E[X])^4 f(x)\,dx - 3$ 等を求めることができる。
それぞれの関数は pdf_* 等と *normal 等をあわせて pdf_normal という関数を表す。また、(x,...) の... は分布の引数を表す。たとえば pdf_normal(x,m,s) など。

●関数タイプ

pdf_*(x,..)	密度関数
cdf_*(x,..)	分布関数
quantile_*(q,..)	分布関数の逆関数
mean_*(...)	平均・期待値
var_*(...)	分散
std_*(...)	標準偏差
skewness_*(...)	歪度（わいど）
kurtosis_*(...)	尖度（せんど）
random_*(...)	乱数

●接尾辞（分布）

*normal(m,s)	平均 m で標準偏差 s の正規分布
*student_t(n)	パラメータ $n > 0$ のスチューデント分布
*chi2(n)	パラメータ $n > 0$ の χ 二乗分布
*f(m,n)	（少なくとも）$m, n > 0$ の F 分布
*exp(m)	パラメータ $m > 0$ の指数分布 (Weibull$(1, 1/m)$)
*lognormal(m,s)	平均 m で標準偏差 s の対数正規分布
*gamma(a,b)	パラメータ $a, b > 0$ のガンマ分布
*beta(a,b)	パラメータ $a, b > 0$ のベータ分布
*continuous_uniform(a,b)	(a, b) の連続一様分布
*logistic(a,b)	$b > 0$ のロジスティック分布
*pareto(a,b)	$a, b > 0$ のパレート分布
*weibull(a,b)	$a, b > 0$ のワイブル分布
*reyleigh(b)	$b > 0$ のレイリー分布（Weibull$(2, 1/b)$）
*laplace(a,b)	$b > 0$ のラプラス分布
*cauchy(a,b)	$b > 0$ のコーシー分布
*gumbel(a,b)	$b > 0$ のグンベル分布
*binomial(n,p)	$n \in \mathbb{N}$, $0 < p < 1$ の二項分布
*poisson(m)	$m > 0$ のポアソン分布
*bernoulli(p)	$0 < p < 1$ のベルヌイ分布
*geometric(p)	$0 < p < 1$ の幾何分布
*discrete_uniform(n)	$n \in \mathbb{N}$ の離散一様分布
*hypergeometric(n1,n2,n)	$n_1, n_2, n \in \mathbb{N}$, $n \le n_1 + n_2$ の超幾何分布
*negative_binomial(n,p)	$n \in \mathbb{N}$, $0 < p < 1$ の負二項分布

★★★入出力★★★

●出力形式

bfloat(exp)	exp の値を bigfloat 浮動小数に変換する
float(exp)	整数・有理数・bigfloat を浮動小数に変換する。
fpprec	bigfloat に対して計算の有効桁を指定する。標準は fpprec:16。
fpprintprec	float や bigfloat の画面表示で用いる桁数を指定する。float に対しては 2 から 16 が有効。bigfloat に対しては 2 から fpprec までが有効。fpprintprec:0 で 16 または fpprec が用いられる。
concat(a1,a2,...)	a1a2... とくっつけたものに変換する。
display(exp1,exp2,...)	exp1, exp2, ... をそのまま左辺で、右辺にその値を出力する。
disp(exp1,exp2,...)	exp1, exp2, ... の値のみを画面に表示する。
tex(exp)	TEX 形式で出力する。
printf(dest,format,exp1,...)	dest を false にすると、整形した文字列を出力する。format は別記する。

● printf 変換

~%	改行
~t	タブ文字
~d・~b・~o・~x	10 進・2 進・8 進・16 進整数
~f	浮動小数
~e	mmmmmEee = mmmmm × 10^{ee} の形式
~g	~f と~e を自動的に選ぶ
~h	bigfloat
~s	Maxima の関数 string() を用いる（文字列"..." を出力する）

●ファイル入出力

batch(filename)	filename を開いて実行する。
batchload(filename)	filename を開いて実行するが、画面表示せず、各式にラベルもつけない。
load(filename)	filename を開いて実行する（batchload する）。Maxima 以外の Lisp ファイルも実行できる。
writefile(filename)	Maxima の作業を filename に出力する。(画面出力の形式で出力されるので、読み込むことはできない。読み込める形式で保存するには stringout を使う。)。既存ファイルに追記したいときには appendfile を使う。
closefile()	writefile や appendfile で開かれたファイルを閉じる。
stringout(filename,exp1,...)	filename に exp1,... を入力するのと同じ形式で保存する。保存したファイルは batch や demo で読み込める。
file_search_maxima	ファイルを探す場所を格納するリスト。

★★★制御★★★

quit()	Maxima を終了する
reset()	変数・オプションなどを初期化する。

●繰り返し

for *var*: *ini* step *incr* thru *term* do *body*	*var* を *ini* から *thru* まで *incr* ずつ増加させながら *body* を実行する。
for *var*: *ini* step *incr* while *cond* do *body*	*var* を *ini* から始めて *cond* が満たされている間 *incr* ずつ増加させながら *body* を実行する。
for *var*: *ini* step *incr* unless *cond* do *body*	*var* を *ini* から始めて *cond* でない間 *incr* ずつ増加させながら *body* を実行する。

これらで step 1 のときには、それを省略できる。また、next を step の代わりに使用して、各段階での変数の変化を変えることができる。

○例

- for a:-3 thru 26 step 7 do display(a)$
- s: 0$
  for i: 1 while i <= 10 do s: s+i;
- series: 1$
  term: exp (sin (x))$
  for p: 1 unless p > 7 do
  (term: diff (term, x)/p,
  series: series + subst (x=0, term)*x^p)$
- step の代わりに next を使用する
  for count: 2 next 3*count thru 20 do display (count)$

●条件判断

<	より小さい
<=	より小さいか等しい
=	等しい
#	等しくない
>	より大きい
>=	より大きいか等しい
and	かつ
or	または
not	でない

●条件分岐

if *cond* then *exp1* else exp0	*cond* が true なら *exp1* でそうでなければ *exp0*
if *cond1* then *exp1* elseif *cond2* then *exp2* elseif ... else *exp0*	*cond1* が true なら exp1 を、そうでなくて cond2 が true なら *exp2* を...、それらでなければ *cond0*

●ブロック

block(exp1,...,expn)	exp1 から順に実行し、expn の値を持って終了する。複雑な関数定義で必須。
block([v1,...,vm],exp1,...,expn)	block と同じだが、$v_1, \ldots, v_m$ はこのブロックでのみ有効な局所変数である。
return(val)	ブロックから抜ける。
go(tag)	ブロック内のタグ tag へジャンプする。例：block([x], x:1, loop_start, x+1, ..., go(loop_start), ...)

★★★パッケージ★★★

augmented_lagrangian_method	近似的な最小を与える
bode	ボードゲイン線図
contrib_ode	常微分方程式
descriptive	記述統計の計算や図示
diag	対角行列（ジョルダン行列）生成
distrib	確率分布の諸量を計算
draw	gnuplot で図をかく
dynamics	力学系やフラクタルの様子をかく
f90	Fortran プログラム生成
ggf	母関数生成
graphs	（グラフ理論の）グラフを作成する
grobner	グレブナー基底を扱う
impdiff	他変数関数の陰導関数を求める
implicit_plot	陰形式の関数のグラフをかく
interpol	多項式のラグランジュ・線形・3 次スプライン補完
lapack	Fortran の LAPACK 移植
lbfgs	非束縛最小化問題を解く L-BFGS アルゴリズム
lindstedt	摂動方程式を扱う Lindstedt コード
linearalgebra	線形代数で用いる有用な関数を集めたもの
lsquares	最小二乗法で用いる有用な関数を集めたもの
makeOrders	指定した変数と指数についてすべての冪を作る
mnewton	多変数にも使えるニュートン法
numericalio	ファイルやストリームを読み書きする関数を集めたもの
opsubst	式のオペレータ部分のみの置換を行う
orthopoly	チェビシェフ・ラゲール・エルミート・ヤコビ・ルジャンドル・ゲーゲンバウアー多項式、球面ベッセル・球面ハンケル・球面調和関数を扱う。
plotdf	1 階・2 階常微分方程式のベクトル場を表示する。
romberg	Romberg 法による数値積分
simplex	単体法を用いた線形計画
simplification	absimp（絶対値）, facexp（関数関係）, ineq（不等式）, rducon（定数項削減）, scifac（因数分解）, sqdenest（平方根）等の簡約化パッケージ
solve_rec	漸化式を解く

stats	統計的推測・仮説検定・意志決定
stirling	gamma(x) を $O(1/x^{2n-1})$ のスターリングの公式で置き換える。
stringproc	文字列取り扱い
unit	単位の変換や次元の取り扱いなど
zeilberger	超幾何和の「定和」に関する Zeiberger のアルゴリズムと「不定和」に関する Gosper のアルゴリズム

★★★デモ★★★

demo("*demofile*") とするとデモが見られる。

cf	連分数
demo	基本機能のデモ
eaton?	?=1,2 ラプラス変換、微積などのデモ
ezgcd	最大公約数に関するデモ
hypgeo	特殊関数および specint によるラプラス変換のデモ
macex	マクロに関するデモ
macro	マクロを用いたリストや関数定義のデモ
newfac	正準有理表現（Canonical Rational Expression）のデモ
romberg	Romberg 法のデモ
sumcon	級数に関するデモ
trgsmp	三角関数の簡約に関するデモ
代数	recur
微積	cartan, fourie, optmiz, optmiz_? ?=1–4, optvar, optvar_? ?=1,2, qual
レヴィン変換	レヴィン変換による級数の和。levin
漸化式	solve_rec
3 次元ベクトル	vector3d
積分	antid, delta
マクロ	defm, keyarg
行列	eigen, eigen_1, nchrpl, pfaff
その他	declin, seqopt
数値計算	cfortr, dblint, dblint_1, diffeq, fft, simpsn, submac
直交多項式	h_atom, variational_method
物理	dimen
簡約	abssimp, disol, facexp, functs, ineq, lrats, rducon, rncomb, scifac, stopex
テンソル	ademo, adsitter, allnutt, atensor, bianchi, bradic, car_iden, ctensor? ?=1–8, einhil, ex_ealc, friedmann, godel, helicity, hodge, itensor? ?=1–9, kaluza, maxwell, papapetrou, petrov, plasma, rainich, reissner, schwarz, spinor, taubnut, tensor, tetrad, weyl
ベクトル	vect, vector