Maxima入門

－　高機能なフリーの数式処理システム　－

中村勝則

2019年8月15日

免責事項

本書の内容は参考的資料であり，掲載したプログラムリストは全て試作品である．本書の使用に伴って発生した不利益，損害の一切の責任を筆者は負わない．

目 次

1　はじめに

筆者が子供であった 1970 年代はまだ電卓（電子卓上計算機）は珍しい存在であり，計算処理は人間の手で行うものであるというのが 1 つの常識であった．小学校では四則計算のトレーニングが重視され，珠算の技能を持つものは就職においても優遇されていた時代である．今では電卓だけでなくパソコン（パーソナルコンピュータ）が十分普及し，スプレッドシートなどの便利なソフトウェア・ツールも利用でき，計算処理は人間の手作業を離れて，誰もが実行できるものになった．計算機が万人のツールとなったことで，手作業による数値計算が苦手な人も計算という技能の恩恵を浴することができるようになったことは重要な社会的進歩であると考えることができる．

ここで紹介するのは数式処理システムの 1 つである **Maxima** というソフトウェア・ツールである．数式処理システムも計算処理を実行するシステムであるが，それは数値計算の実行のみならず，記号代数的に表現された数式を記号のまま計算処理することができるという点が最も重要である．簡単にいうと，

$$a+a$$

という式を

$$2a$$

と簡略化する作業を自動的に実行するような処理である．これだけでは数式処理システムの有用性がよくわからないという場合，$(x+a)^{30}$ を展開する例を見てみる．下の例は実際にこの展開作業を Maxima で行ったものである．

$$x^{30}+30\,a\,x^{29}+435\,a^2\,x^{28}+4060\,a^3\,x^{27}+27405\,a^4\,x^{26}+142506\,a^5\,x^{25}+593775\,a^6\,x^{24}+$$
$$2035800\,a^7\,x^{23}+5852925\,a^8\,x^{22}+14307150\,a^9\,x^{21}+30045015\,a^{10}\,x^{20}+54627300\,a^{11}\,x^{19}+$$
$$86493225\,a^{12}\,x^{18}+119759850\,a^{13}\,x^{17}+145422675\,a^{14}\,x^{16}+155117520\,a^{15}\,x^{15}+145422675\,a^{16}\,x^{14}+$$
$$119759850\,a^{17}\,x^{13}+86493225\,a^{18}\,x^{12}+54627300\,a^{19}\,x^{11}+30045015\,a^{20}\,x^{10}+14307150\,a^{21}\,x^{9}+$$
$$5852925\,a^{22}\,x^{8}+2035800\,a^{23}\,x^{7}+593775\,a^{24}\,x^{6}+142506\,a^{25}\,x^{5}+27405\,a^{26}\,x^{4}+4060\,a^{27}\,x^{3}+$$
$$435\,a^{28}\,x^{2}+30\,a^{29}\,x+a^{30}$$

勿論これは筆者が手で計算したものではなく，Maxima が基本的に持っている計算機能により自動的に算出されたものであり，計算結果は一瞬にして得られる．

Maxima では代数式の展開や因子分解（因数分解），微分や積分とそれに関連する解析学的な操作，方程式（代数方程式，微分方程式）の求解，各種の関数に関する計算，行列とベクトルに関する計算からグラフの描画まで行うことができる．このようなソフトウェア・ツールの普及により高度な数学的計算が手軽に行え，私達の知的活動の範囲が大幅に広がることが期待できる．

1.1　Maxima の生い立ち

Maxima の開発は 1960 年代まで遡ることができる．当初 Macsyma という数式処理システムが MIT（マサチューセッツ工科大学）で Maclisp（LISP 系言語の 1 つ）を用いて開発された．それが 1982 年に Common Lisp（これも LISP 系言語の 1 つ）の言語処理系の上に移植され，Maxima と呼ばれるシステムとなった．現在 Maxima は GNU の GPL に基づくフリーソフトウェアとして配布されている．

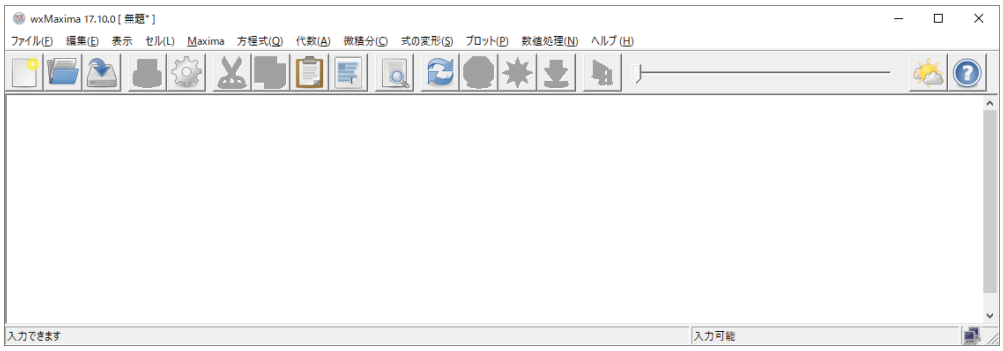
1.1.1　wxMaxima

現在は，TEX の数式清書機能を持つフロントエンドを備えた wxMaxima （図 1）というアプリケーションとして利用でき，

http://maxima.sourceforge.net/

から入手が可能である．

本書では，wxMaxima の操作環境を前提として Maxima の使用方法について解説する．



左がアプリケーションのアイコン，システムを起動すると右のようなウィンドウが表示される．

図 1: wxMaxima

2 基本的な機能

2.1 数式の入力

数式は Maxima のコマンドとして入力する．1 つの式の終端には ‘;’（セミコロン）を付け [1]，Shift + Enter を押すことで処理が実行される．(Enter のみの場合は単なる改行処理となる)

算術演算の記号は下記の通りである．

- **和** ‘+’ で記号や式を連結する．
  例．$a + a$ の簡単化

```
(%i1)  a+a;
(%o1)  2 a
```

- **積** ‘*’ で記号や式を連結する
  例．$x \times x$ の簡単化

```
(%i2)  x*x;
(%o2)  x^2
```

- **差** ‘-’ で記号や式を連結する
  例．$5a - 2a$ の簡単化

```
(%i3)  5*a - 2*a;
(%o3)  3 a
```

- **商** ‘/’ で記号や式を連結する
  例．$\frac{6y^7}{4y^3}$ の簡単化

```
(%i4)  6*y^7 / (4*y^3);
(%o4)  3 y^4 / 2
```

[1] 最新の wxMaxima では，終端のセミコロンを付けなくても自動的に付加される．

- **冪乗**　‘^’ で記号や式を連結する
  上記の例を参照のこと．

2.1.1　ノートブックとセル

wxMaxima のウィンドウ内には入力した式とそれを評価した結果が表示されている．この「入力と評価結果」の組は**セル**と呼ばれ，1 つの処理の単位となる．複数の処理の過程は**ノートブック**という形式で保持され，セルの連鎖から成る．

2.1.1.1　評価結果の非表示

入力する式の末尾に ‘$’ を付けると，評価結果を表示しない．

2.1.2　処理過程のヒストリ

上の実行例の中に見られる%記号は処理過程のヒストリを意味する．例えば%o2 には x^2 が，%o3 には $3a$ が割り当てられており，これをそのまま以後の処理に使用することができる．（下記参照）

```
(%i5)  %o2 + %o3;
(%o5)  x^2+3*a
```

2.2　式の展開と因数分解

式を展開する場合は expand(式)，因数分解する場合は factor(式) を実行する．（下記参照）

```
(%i6)  expand( (a*x^2+b*x+c)*(d*x^2+e*x+f) );
(%o6)  a*d*x^4+a*e*x^3+b*d*x^3+a*f*x^2+b*e*x^2+c*d*x^2+b*f*x+c*e*x+c*f
(%i7)  factor( % );
(%o7)  (a*x^2+b*x+c)*(d*x^2+e*x+f)
```

この例の%i7 に factor(%) と入力があるが，1 文字の%は直前の処理結果を意味する．

2.3　基本的な関数

Maxima で使える数学的関数の基本的なものを表 1 に挙げる．

2.4　重要な定数

表 2 に重要な定数を挙げる．これらの値を用いた計算の例を下に示す．

```
(%i8)  sin(%pi/2);
(%o8)  1
(%i9)  log(%e);
(%o9)  1
```

表 1: 基本的な関数

記述	意味	
sqrt(x)	$\sqrt{x}$	
log(x)	$\log x$	自然対数
exp(x)	e^x	指数関数
sin(x)	$\sin x$	正弦関数
cos(x)	$\cos x$	余弦関数
tan(x)	$\tan x$	正接関数
asin(x)	$\sin^{-1} x$	正弦関数の逆関数
acos(x)	$\cos^{-1} x$	余弦関数の逆関数
atan(x)	$\tan^{-1} x$	正接関数の逆関数
abs(x)	$\|x\|$	絶対値
mod(x,p)	$x \ mod \ p$	法 p での x の剰余

表 2: 重要な定数

記述	意味
%pi	π（円周率）
%e	e（ネイピア数：自然対数の底）
%i	i（虚数単位）

```
(%i10) %i * %i;
(%o10) -1
```

上から順に，$\sin\left(\frac{\pi}{2}\right)$，$\log_e e$，$i^2$ を計算した例である．

2.5 数値計算

式の値を数値の形で求めるには bfloat(式) を実行する．下に実行例を示す．

```
(%i11) tan(%pi/3);
(%o11) sqrt(3)
(%i12) bfloat(%);
(%o12) 1.7320508075688877b0
(%i13) bfloat(%pi);
(%o13) 3.1415926535897793b0
```

求める数値の精度（桁数）を指定するには，‘fpprec:桁数’ を実行する．（下記参照）

```
(%i14) fpprec:70;
(%o14) 70
```

```
(%i15) bfloat(%pi);
(%o15) 3.14159265358979323846264338327950288419716939937510582097494459230781640b0
```

これは計算する精度を 70 桁に指定した例である．

2.6 変数への値の設定と関数の定義

コロン ':' を使って，変数に値を設定することができる．(下記参照)

```
(%i17) a:21;
       b:57;
(a)    21
(b)    57
(%i18) a+b;
(%o18) 78
```

これは，変数 a に 21 を，変数 b に 57 を設定した後，$a+b$ を計算している例である．

':=' を使って，関数を定義することができる．(下記参照)

```
(%i19) f(x) := 3*x^2+1;
                          2
(%o19) f(x) := 3 x  + 1
(%i20) f(5);
(%o20) 76
```

これは，$f(x)=3x^2+1$ を定義した後，$f(5)$ を算出している例である．

変数への値の設定や関数の定義を解除するには kill(変数名もしくは関数名) を実行する．(下記参照)

```
(%i21) kill(a,b,f);
(%o21) done
(%i23) a+b;
       f(5);
(%o22) b + a
(%o23) f(5)
```

値の設定と関数の定義が解除されていることがわかる．

kill(all) を実行すると，全ての設定と定義が解除される．

2.7 総和

sum(f(k),k,i,n) を実行することで，式 $f(k)$ の $k = i \cdots n$ の範囲の総和を求めることができる．
1〜10 の合計を求めるには次のように実行する．

```
(%i24) sum(k,k,1,10);
(%o24) 55
```

下のように総和は形式的に処理することもできる．

```
(%i26) sum(k*d,k,i,n);
       sum(r^k,k,i,n);
(%o25) d \sum_{k=i}^{n} k
(%o26) \sum_{k=i}^{n} r^k
```

2.7.1 総和の一般化

sum の実行時に，simpsum=true [2] を付加することで計算結果を一般化（簡単化）できる場合がある．（下記参照）

```
(%i28) sum(k*d,k,i,n),simpsum=true;
       sum(r^k,k,i,n),simpsum=true;
(%o27) d \left( \frac{n^2+n}{2} - \frac{i+(i-1)^2-1}{2} \right)
(%o28) \frac{r^{n+1}-r^i}{r-1}
```

2.8 wxMaxima フロントエンド

wxMaxima のフロントエンドウィンドウの基本的な操作方法について説明する．

2.8.1 ノートブックの保存

「ファイル」メニューから「名前を付けて保存」を選択すると，保存作業のためのダイアログが開く．（図 2 参照）
ここで保存する場所とファイル名を設定して「保存」ボタンをクリックする．

保存するノートブックのファイル名の拡張子は *.wxmx となる．

参考）計算結果を Common Lisp のプログラムとしてファイルに保存するには，save(”ファイル名”,all) を実行する．（下記参照）

```
(%i29) save("TheHistory",all);
(%o29) C : /ProgramFiles/Maxima − 5.28.0 − 2/wxMaxima/TheHistory
```

[2] true は真理値である．偽は false である．

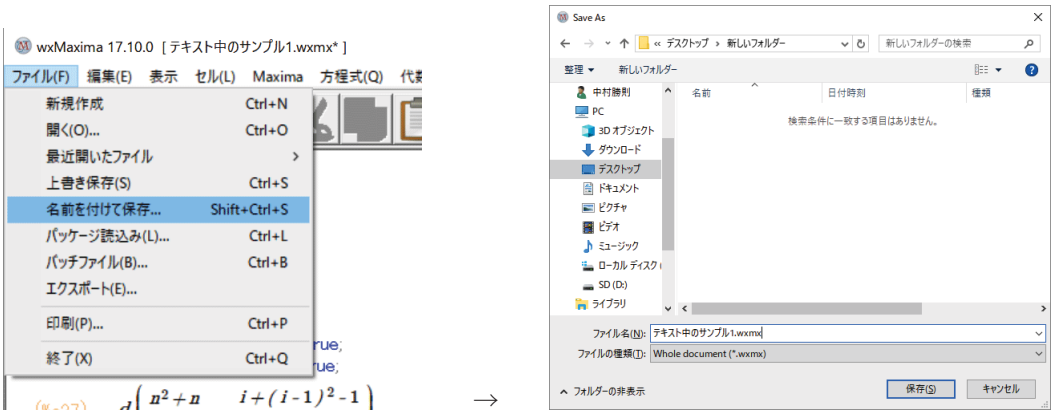
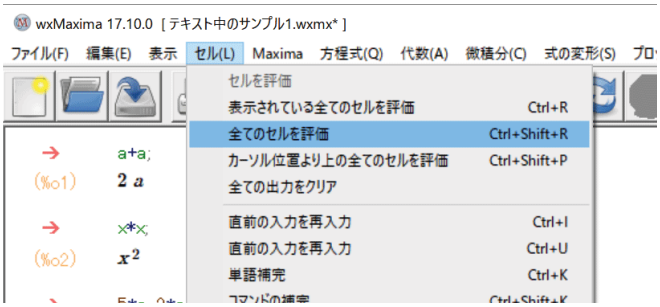


図 2: ノートブックの保存

2.8.2 ノートブックの読込み

保存したノートブックを読み込むには,「ファイル」メニューから「開く」を選択する．この操作によりファイルを選択するダイアログが開くので，場所とファイルを選択して「開く」ボタンをクリックする．

この操作によってノートブックを読み込んだ段階では，まだその内容は評価されていない．ノートブック内のすべてのセルを再度評価（図 3）することで，後続の処理をノートブック内に追加することができる．



「セル」メニューから「すべてのセルを評価」を選択する

図 3: セルの再評価

2.8.3 処理結果の TEX フォーマット出力

ノートブックの内容を TEX の形式で保存することができる.「ファイル」メニューから「エクスポート」を選択すると，出力作業のためのダイアログが開くので，保存する場所とファイル名，それにファイルの種類として「TeX ファイル」を設定して「保存」ボタンをクリックする．

3 微分・積分

式（関数）の導関数を求めるには diff(式, 変数) を実行する．また，原始関数を求めるには integrate(式, 変数) を実行する.（下記参照）

```
(%i1)  diff( a*x^3+b*x^2+c*x+d, x );
(%o1)  3 a x^2 + 2 b x + c
(%i2)  integrate( %, x );
(%o2)  a x^3 + b x^2 + c x
```

これは ax^3+bx^2+cx+d を変数 x で微分し，それの不定積分を求めた例である．(定数項は省略されている)
下記のように，形式的な関数の式 $f(x)$ の導関数や原始関数を求めることもできる．

```
(%i3)  diff( f(x), x );
```

$$(\%o3)\quad \frac{d}{dx}\,\mathrm{f}\,(x)$$

```
(%i4)  integrate( %, x );
```

$$(\%o4)\quad \mathrm{f}\,(x)$$

```
(%i5)  integrate( f(x), x );
```

$$(\%o5)\quad \int \mathrm{f}\,(x)\,dx$$

```
(%i6)  diff( %, x );
```

$$(\%o6)\quad \mathrm{f}\,(x)$$

複雑な関数の導関数や原始関数を求めることもできる．(下記参照)

```
(%i7)  diff( sin(cos(x))*cos(sin(x)), x );
```

$$(\%o7)\quad -\cos(x)\,\sin(\cos(x))\,\sin(\sin(x))-\sin(x)\,\cos(\cos(x))\,\cos(\sin(x))$$

```
(%i8)  integrate( 1/sin(x), x );
```

$$(\%o8)\quad \frac{\log(\cos(x)-1)}{2}-\frac{\log(\cos(x)+1)}{2}$$

3.1 級数展開

式（関数）をテーラー展開するには taylor(式, 変数, 展開の開始値, 展開の次数) を実行する．
下の例は $f(x)=e^{-\frac{x}{10}}\sin(x)+1$ と定義された関数（図 4）を $f(a)$ の位置を中心にして展開したものである．
まず，次のようにして関数 $f(x)=e^{-\frac{x}{10}}\sin(x)+1$ を定義する．

```
(%i9)  f(x):=%e^(-x/10) * sin(x)+1;
```

$$(\%o9)\quad \mathrm{f}\,(x):=e^{\frac{-x}{10}}\sin(x)+1$$

関数の定義には ':=' を用いる．
5 次の項まで計算した例を下に示す．

```
(%i11) taylor(f(x), x, a, 5);
```

$$(\%o11)/\mathrm{T}/\quad \frac{e^{\frac{a}{10}}+\sin(a)}{e^{\frac{a}{10}}}-\frac{(\sin(a)-10\cos(a))\,(x-a)}{10\,e^{\frac{a}{10}}}-\frac{(99\sin(a)+20\cos(a))\,(x-a)^2}{200\,e^{\frac{a}{10}}}+\frac{(299\sin(a)-970\cos(a))\,(x-a)^3}{6000\,e^{\frac{a}{10}}}+\frac{(9401\sin(a)+3960\cos(a))\,(x-a)^4}{240000\,e^{\frac{a}{10}}}-\frac{(49001\sin(a)-90050\cos(a))\,(x-a)^5}{12000000\,e^{\frac{a}{10}}}+\ldots$$

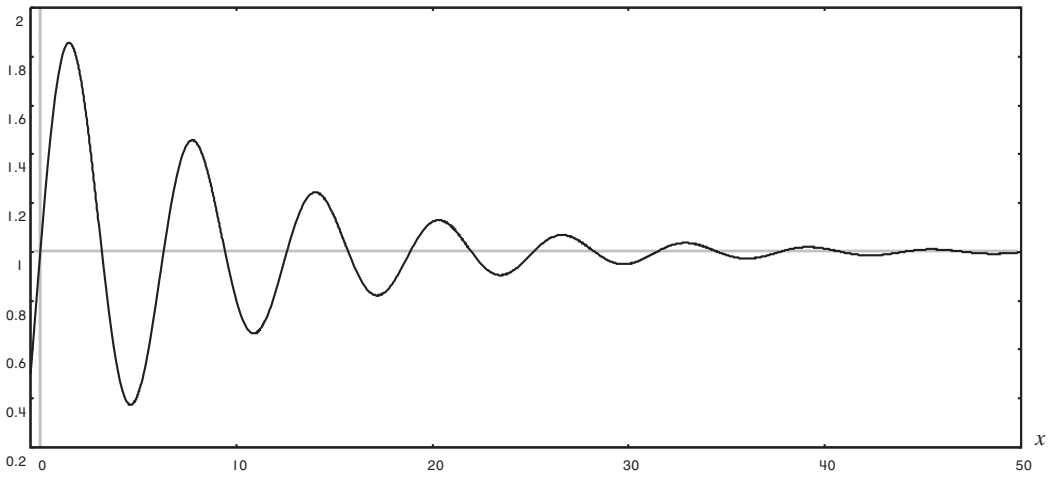


図 4: $f(x)$

3.2 極限

式（関数）の極限値を求めるには limit(式, 変数, 求める極限の位置（先）) とする．

上で定義した関数 $f(x)$ の $x \to \infty$ の極限を求めた例を下に示す．

```
(%i12) limit(f(x), x, inf);
```

(%o12) 1

上記 inf は ∞ を意味する．$-\infty$ を求める場合は minf を指定する．
関数の極限は，近づく方向（定義域の左右）によって値が異なることがある．次のような関数を考える．

$$g(x) = \frac{1}{x-1} + 1$$

これを Maxima で次のように定義する．

```
(%i13) g(x):=1/(x-1)+1;
```

(%o13) $\mathrm{g}\,(x) := \dfrac{1}{x-1} + 1$

この関数は図 5 のように $x = 1$ で発散するが，この点に近づく方向によって $-\infty$，∞ のどちらに発散するかが異なる．

下に示す例は，極限に近づく方向の左右を指定しているものである．

```
(%i15) limit(g(x), x, 1, minus);
```

(%o15) $-\infty$

```
(%i16) limit(g(x), x, 1, plus);
```

(%o16) ∞

定義域の左側から極限に近づく場合は minus を，右側から近づく場合は plus を指定する．

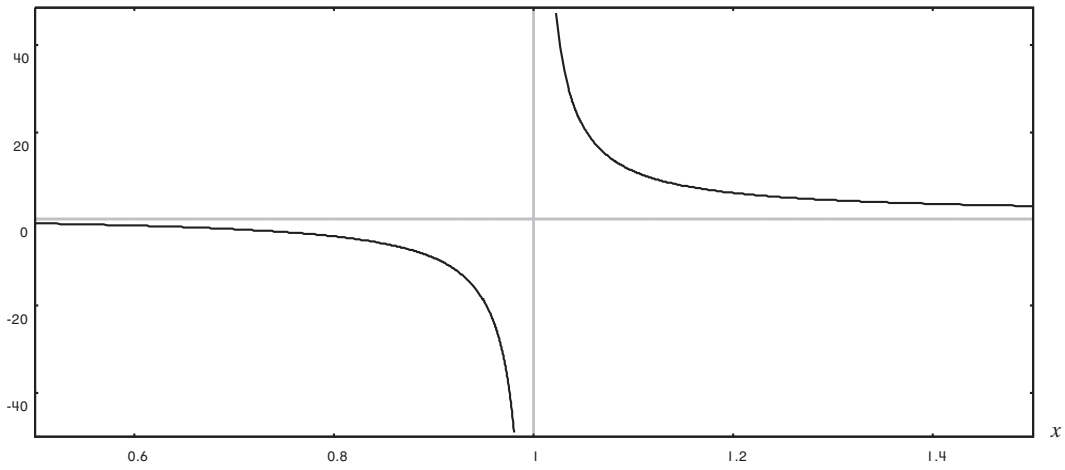


図 5: $f(x)$

4 方程式の求解

4.1 代数方程式の求解

代数方程式の解を求めるには，solve(方程式, 求める変数) を実行する．下に，2 次の代数方程式 $ax^2+bx+c=0$ を x について解く例を示す．

```
(%i1)  solve(a*x^2+b*x+c=0, x);
```

$$(\%o1)\quad [x=-\frac{\sqrt{b^2-4\,a\,c}+b}{2\,a},x=\frac{\sqrt{b^2-4\,a\,c}-b}{2\,a}]$$

高次の代数方程式も解くことができる．下の例は $ax^3+bx^2+cx+d=0$ を x について解いたものである．

```
(%i2)  solve(a*x^3+b*x^2+c*x+d=0, x);
```

$$\begin{aligned}(\%o2)\quad [x=&\left(-\frac{\sqrt{3}\,i}{2}-\frac{1}{2}\right)\\ &\left(\frac{\sqrt{27\,a^2\,d^2+\left(4\,b^3-18\,a\,b\,c\right)\,d+4\,a\,c^3-b^2\,c^2}}{2\,3^{\frac{3}{2}}\,a^2}-\frac{27\,a^2\,d-9\,a\,b\,c+2\,b^3}{54\,a^3}\right)^{\frac{1}{3}}-\\ &\frac{\left(\frac{\sqrt{3}\,i}{2}-\frac{1}{2}\right)\,\left(3\,a\,c-b^2\right)}{9\,a^2\,\left(\frac{\sqrt{27\,a^2\,d^2+\left(4\,b^3-18\,a\,b\,c\right)\,d+4\,a\,c^3-b^2\,c^2}}{2\,3^{\frac{3}{2}}\,a^2}-\frac{27\,a^2\,d-9\,a\,b\,c+2\,b^3}{54\,a^3}\right)^{\frac{1}{3}}}-\frac{b}{3\,a},\\ x=&\left(\frac{\sqrt{3}\,i}{2}-\frac{1}{2}\right)\,\left(\frac{\sqrt{27\,a^2\,d^2+\left(4\,b^3-18\,a\,b\,c\right)\,d+4\,a\,c^3-b^2\,c^2}}{2\,3^{\frac{3}{2}}\,a^2}-\frac{27\,a^2\,d-9\,a\,b\,c+2\,b^3}{54\,a^3}\right)^{\frac{1}{3}}-\\ &\frac{\left(-\frac{\sqrt{3}\,i}{2}-\frac{1}{2}\right)\,\left(3\,a\,c-b^2\right)}{9\,a^2\,\left(\frac{\sqrt{27\,a^2\,d^2+\left(4\,b^3-18\,a\,b\,c\right)\,d+4\,a\,c^3-b^2\,c^2}}{2\,3^{\frac{3}{2}}\,a^2}-\frac{27\,a^2\,d-9\,a\,b\,c+2\,b^3}{54\,a^3}\right)^{\frac{1}{3}}}-\frac{b}{3\,a},\\ x=&\left(\frac{\sqrt{27\,a^2\,d^2+\left(4\,b^3-18\,a\,b\,c\right)\,d+4\,a\,c^3-b^2\,c^2}}{2\,3^{\frac{3}{2}}\,a^2}-\frac{27\,a^2\,d-9\,a\,b\,c+2\,b^3}{54\,a^3}\right)^{\frac{1}{3}}-\\ &\frac{3\,a\,c-b^2}{9\,a^2\,\left(\frac{\sqrt{27\,a^2\,d^2+\left(4\,b^3-18\,a\,b\,c\right)\,d+4\,a\,c^3-b^2\,c^2}}{2\,3^{\frac{3}{2}}\,a^2}-\frac{27\,a^2\,d-9\,a\,b\,c+2\,b^3}{54\,a^3}\right)^{\frac{1}{3}}}-\frac{b}{3\,a}]\end{aligned}$$

4.1.1 連立 1 次方程式

連立 1 次方程式を解くには，linsolve(方程式リスト, 変数リスト) を実行する．**リスト**は ‘[ ]’ 内にコンマ ‘,’ で区切って項目を列挙したものである．
2 つの未知変数 x, y を含む連立方程式

$$\begin{cases} ax + by = e \\ cx + dy = f \end{cases}$$

を解く例を下に示す．

```
(%i3)  linsolve([a*x+b*y=e, c*x+d*y=f], [x,y]);
```

$$(\%o3) \quad [x = -\frac{d\,e - b\,f}{b\,c - a\,d}, y = \frac{c\,e - a\,f}{b\,c - a\,d}]$$

4.1.2 高次連立方程式

高次連立方程式を解くには，algsys(方程式リスト, 変数リスト) を実行する．
2 つの未知変数 x, y を含む高次連立方程式

$$\begin{cases} x^2 + x + y^2 + y = 1 \\ -x^2 - 2 * x + y^2 + y = -1 \end{cases}$$

を解く例を下に示す．

```
(%i4)  algsys([x^2+x+y^2+y=1, -x^2-2*x+y^2+y=-1], [x,y]);
```

$$(\%o4) \quad [[x = \frac{1}{2}, y = -\frac{\sqrt{2}+1}{2}], [x = \frac{1}{2}, y = \frac{\sqrt{2}-1}{2}],$$
$$[x = -2, y = -\frac{\sqrt{3}\,i+1}{2}], [x = -2, y = \frac{\sqrt{3}\,i-1}{2}]]$$

4.2 微分方程式の求解

微分方程式を解くには，desolve(微分方程式, 未知関数) [3] を実行する．未知の関数 $f(x)$ を含む微分方程式

$$\frac{d}{dx}f(x) = f(x)$$

を解く例を下に示す．

```
(%i5)  desolve( diff(f(x),x)=f(x), f(x) );
```

$$(\%o5) \quad \mathrm{f}(x) = \mathrm{f}(0)\,e^x$$

微分方程式を解く機能は desolve 以外にも ode2(微分方程式, 未知関数, 変数) [4] などがある．

[3] desolve はラプラス変換を用いて微分方程式を解く．
[4] 2 階までの常微分方程式を解く．

4.3　差分方程式の求解（漸化式の一般化）

Maxima には差分方程式（階差方程式）の解を求める（漸化式の一般化を行う）ためのパッケージ ‘solve_rec’ が提供されている．この機能を利用するには，下記のような操作により，必要なパッケージを Maxima に読み込んでおく必要がある．

```
(%i6)  load(solve_rec)$
```

差分方程式（階差方程式）の解を求めるには，

　　solve_rec(差分方程式, 一般化したい項, 初期値, ···)

を実行する．下に差分方程式 $y(n) = (1 + d)y(n - 1)$ を初期値 $y(0) = y0$ として求める例を示す．

```
(%i7)  solve_rec( y(n)=(1+d)*y(n-1), y(n), y(0)=y0 );
```

(%o7)　$\mathrm{y}\,(n) = (d+1)^n\, y0$

方程式は，下記のように添字付きの項（配列）で表現されるものとして記述することもできる．

```
(%i8)  solve_rec( y[n]=(1+d)*y[n-1], y[n], y[0]=y0 );
```

(%o8)　$y_n = (d+1)^n\, y0$

参考）　差分方程式の解の極限を求めることで，微分方程式の解と同じものを算出することができる場合がある．（下記参照）

```
(%i9)  limit( %o8, n, inf ),d=x/n;
```

(%o9)　$\lim_{n\to\infty} y_n = e^x\, y0$

5　ベクトル・行列

5.1　基本

下記のような行列とベクトルに対する処理を例に挙げて説明する．

$$\boldsymbol{m_1} = \begin{pmatrix} 1 & 3 & 0 \\ 0 & 10 & 0 \\ 13 & 0 & 17 \end{pmatrix},\quad \boldsymbol{m_2} = \begin{pmatrix} 2 & 0 & 6 \\ 0 & 9 & 11 \\ 0 & 0 & 18 \end{pmatrix},\quad \boldsymbol{v} = \begin{pmatrix} a & b & c \end{pmatrix},\quad \boldsymbol{v_t} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

行列を構成するには，matrix(行成分リスト 1, 行成分リスト 2,$\cdots$) を実行する．またベクトルは成分のリストで表現する．(ここでは全て行列として扱う．)

```
(% i4) m1: matrix(
       [1,3,0],
       [0,10,0],
       [13,0,17]
       );
       m2: matrix(
       [2,0,6],
       [0,9,11],
       [0,0,18]
       );
       v:matrix([a,b,c]);
       vt:transpose(matrix([x,y,z]));
```

(m1)
$$\begin{pmatrix} 1 & 3 & 0 \\ 0 & 10 & 0 \\ 13 & 0 & 17 \end{pmatrix}$$

(m2)
$$\begin{pmatrix} 2 & 0 & 6 \\ 0 & 9 & 11 \\ 0 & 0 & 18 \end{pmatrix}$$

(v)
$$\begin{pmatrix} a & b & c \end{pmatrix}$$

(vt)
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

これで行列とベクトルの定義ができた．上の例の中の transpose(行列) は与えた行列の**転置行列**を求めるもので，横並びの成分で構成されるベクトルを縦の形式にする場合にも使用する．

例. 行列 m1 の転置行列を求めるには次のように実行する.

```
(%i5)  transpose(m1);
```

$$(\%o5)\quad \begin{pmatrix} 1 & 0 & 13 \\ 3 & 10 & 0 \\ 0 & 0 & 17 \end{pmatrix}$$

5.2 行列の積

行列の積はドット '.' で表現する.(下記の例を参照)

```
(%i6)  m1.m2;
```

$$(\%o6)\quad \begin{pmatrix} 2 & 27 & 39 \\ 0 & 90 & 110 \\ 26 & 0 & 384 \end{pmatrix}$$

```
(%i7)  m1.vt;
```

$$(\%o7)\quad \begin{pmatrix} 3\,y + x \\ 10\,y \\ 17\,z + 13\,x \end{pmatrix}$$

```
(%i8)  v.vt;
```

$$(\%o8)\quad c\,z + b\,y + a\,x$$

5.3 行列式

行列式を求めるには determinant(行列) を実行する. 行列 m1 の行列式を求める例を下に示す.

```
(%i9)  determinant(m1);
```

$$(\%o9)\quad 170$$

5.4 逆行列

逆行列を求めるには invert(行列) を実行する. 行列 m1 の逆行列を求める例を下に示す.

```
(%i10) invert(m1);
```

$$(\%o10)\quad \begin{pmatrix} 1 & -\frac{3}{10} & 0 \\ 0 & \frac{1}{10} & 0 \\ -\frac{13}{17} & \frac{39}{170} & \frac{1}{17} \end{pmatrix}$$

5.5 固有値，固有ベクトル

行列の固有値を求めるには eigenvalues(行列) を実行する．固有値だけでなく，固有ベクトルまで求める場合は eigenvectors(行列) を実行する．

行列 m1 の固有値と固有ベクトルを求める例を下に示す．

```
(%i11) ev:eigenvalues(m1);
```

(ev) $[\,[1,10,17],\ [1,1,1]\,]$

処理結果は [固有値リスト, 重複度リスト] の形で得られる．

```
(%i12) ev2:eigenvectors(m1);
```

(ev2) $[\,[\,[1,10,17],\ [1,1,1]\,],\ [\,[\,[1,0,-\frac{13}{16}]\,]\,],\ [\,[1,3,-\frac{13}{7}]\,]\,,[\,[0,0,1]\,]\,]\,]$

処理結果は [[固有値リスト, 重複度リスト], 固有ベクトルのリストのリスト] の形で得られる．
「固有ベクトルのリストのリスト」は各固有値に対応する固有ベクトルの集合をリストにしたものである．

5.6 その他

余因子行列を求めるには，adjoint(行列) を実行する．行列 m1 の余因子行列を求める例を下に示す．

```
(%i13) adjoint(m1);
```

(%o13)
$$\begin{pmatrix} 170 & -51 & 0 \\ 0 & 17 & 0 \\ -130 & 39 & 10 \end{pmatrix}$$

6　グラフィックス

6.1　2 次元プロット

$y = f(x)$（x は独立変数，y は従属変数）という関数がある場合，それをグラフとして可視化することができる．

plot2d(関数の式,[独立変数, 開始値, 終了値]) を実行することでグラフが描画される．下の例は $\sin(x)$ を $-4 \leq x \leq 4$ の範囲で描画（図 6）するものである．（**wxplot2d** を使うとノートブックにグラフが埋め込まれる）

```
(%i1)  plot2d(sin(x),[x,-4,4]);
(%o1)
```

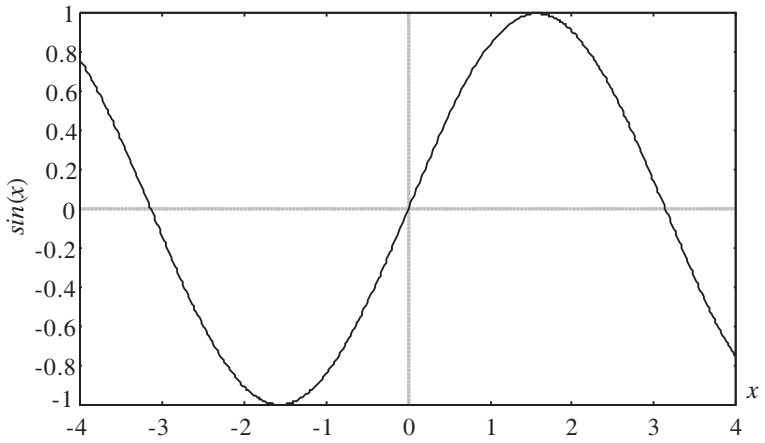


図 6: 描画結果

下の例のようにして実行することで，従属変数の範囲を指定して描画（図 7）することもできる．

```
(%i2)  plot2d(sin(x),[x,-20,20],[y,-2,2]);
(%o2)
```

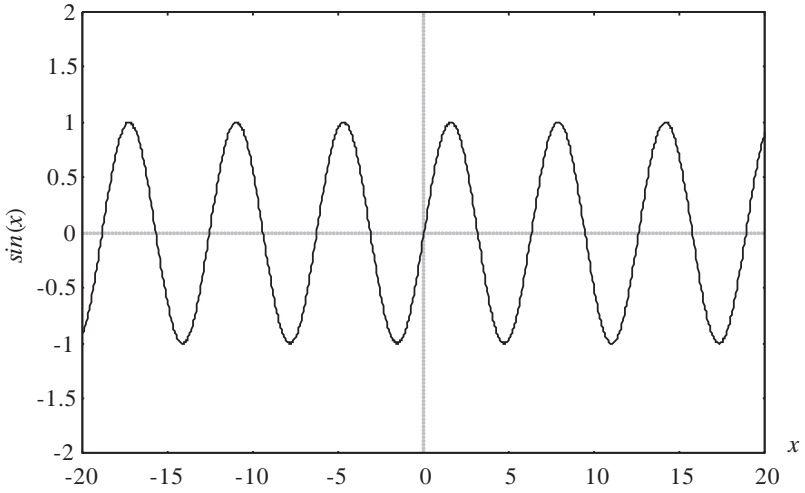


図 7: 描画結果（縦軸の範囲指定）

同一の座標平面に複数の関数を同時に描画する場合は，下の例のように関数のリストを与えて実行する．

```
(%i3)  plot2d([sin(x),cos(x),exp(x/10)],[x,-10,10],[y,-2,3]);
(%o3)
```

これは $\sin(x), \cos(x), \exp\left(\frac{x}{10}\right)$ を同時に描画するもので，図 8 のようなグラフができる．

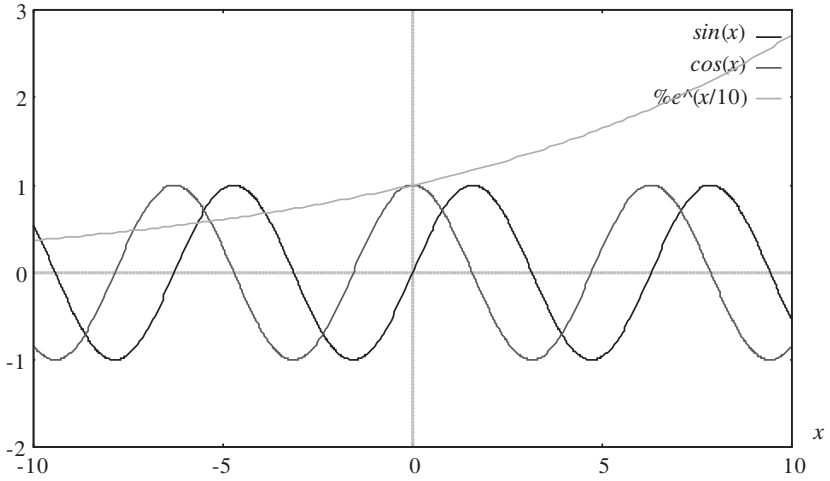


図 8: 複数の関数の同時描画

6.1.1 座標リストのプロット

下記のように discrete を指定することで，座標リストのプロットが可能となる．

plot2d([discrete, リスト], オプション)

wxplot2d([discrete, リスト], オプション)

```
(%i7)  p:[[0,0],[1,1],[2,2],[3,3],[4,4]]$
       plot2d([discrete,p],[style,points],[point_type,circle]);
(%o8)
```

この例のように，スタイルとして [style,points] を追加することで，点表示のプロットができる．またことのき，point_type オプションを追加することで，点表示のスタイルを変更できる．(図 9 の例を参照)

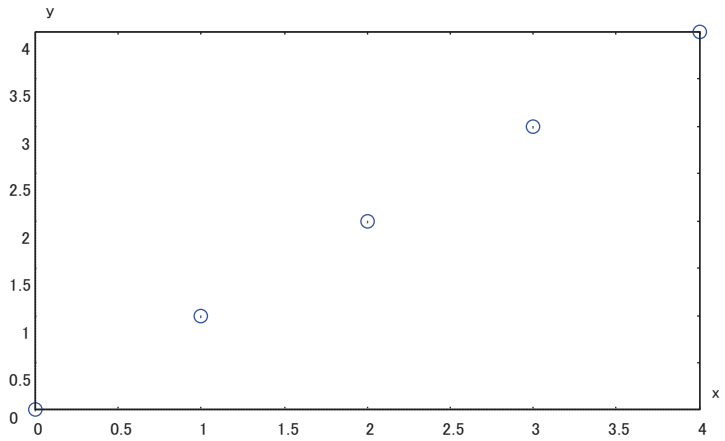


図 9: 点表示のプロット

point_type オプションに与えるスタイルの一部を表 3 に挙げる．

表 3: point_type オプションの代表的なもの

スタイル	形状
bullet	●
circle	○
plus	+
times	×
asterisk	*
box	■
square	□

6.2 3 次元プロット

$z = f(x, y)$（x, y は独立変数，z は従属変数）という関数がある場合，それを 3 次元のグラフとして可視化することができる．

plot3d(関数の式,[独立変数 1, 開始値, 終了値],[独立変数 2, 開始値, 終了値]) を実行することで 3 次元のグラフが描画される．(**wxplot3d** を使うとノートブックにグラフが埋め込まれる)

下の例は $\cos\left(\sqrt{x^2+y^2}\right)$ を $-4 \leq x \leq 4, -4 \leq y \leq 4$ の範囲で描画するものである．

```
(%i4)  plot3d(cos(sqrt(x^2+y^2)),[x,-4,4],[y,-4,4]);
(%o4)
```

この結果，図 10 のようなグラフができる．

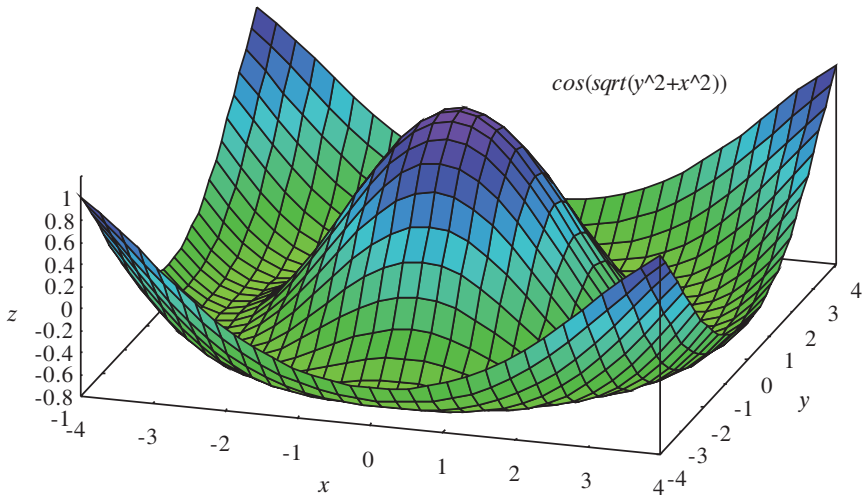


図 10: 3 次元のグラフの描画

下の例のようにして実行することで，描画のメッシュを増やして描画（図 11）することもできる．

```
(%i5)  plot3d(cos(sqrt(x^2+y^2)),[x,-4,4],[y,-4,4],[grid,100,100]);
(%o5)
```

この例のように，[grid, 独立変数 1 のメッシュの数, 独立変数 2 のメッシュの数] を実行時に指定する．

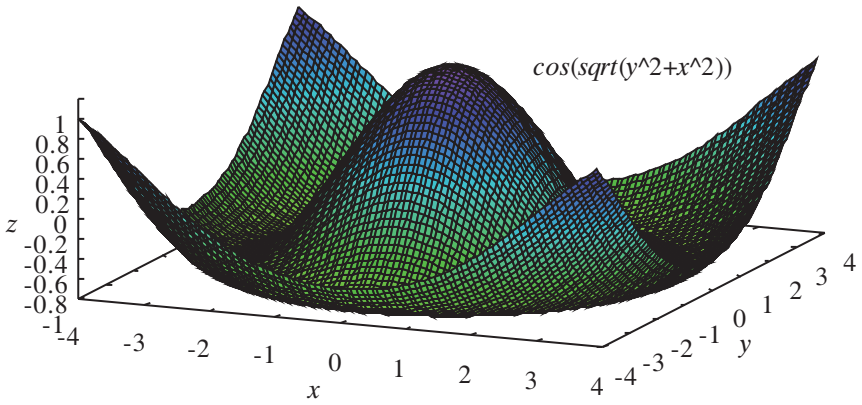


図 11: 3 次元のグラフの描画（メッシュ：100 × 100）

6.3 wxMaxima の場合の描画

wxMaxima の場合，plot2d,plot3d の代わりにそれぞれ wxplot2d,wxplot3d を実行することもできる．この場合，描画結果のグラフはノートブック内に表示される．

6.3.1 描画時のオプション： 色指定 （wxMaxima + gnuplot エンジン）

色を指定してグラフを描画するには color オプションを追加する．
例えば “[color, 色]”を追加して，グラフ描画の色を変えることができる．

例．plot2d(sin(x),[x,-4,4],[color,black])

これで描画色が黒になる．MS-Windows 環境下での色指定を表 4 に挙げる．

表 4: 色指定（MS-Windows の場合）

black	黒
blue	青
red	赤
green	緑
magenta	マゼンタ
cyan	シアン

6.3.2 描画時のオプション： 縦横比の設定 （wxMaxima + gnuplot エンジン）

オプションとして

[gnuplot_preamble,”set size 横の比率, 縦の比率”]

を追加することで，描画するグラフ領域の縦横の比率を変更することができる．

6.4 画像のファイル出力（wxMaxima + gnuplot エンジン）

wxMaxima で plot2d や plot3d を実行すると，描画エンジンが gnuplot の場合，描画結果のグラフが別のウィンドウで表示される．この場合，グラフが表示されているウィンドウに「Options」ボタン（メニュー）があり，メニュー項目「Save as EMF」を選択することでグラフを EMF 形式の画像ファイルとして保存することができる．

7　プログラミング

Maxima では関数定義を行うことでユーザが独自に機能を拡張することができる．ここでは更に，条件判定や繰り返しなどの制御を実現する方法について説明する．

7.1　条件判定

文法：　if　条件　then　式 1　else　式 2

与えた条件が真である場合に式 1 を実行し，条件が偽の場合は式 2 を実行する．
下の例は，条件判定を用いてフィボナッチ数列

$$f(n) = \begin{cases} n = 0 & \to \quad 1 \\ n = 1 & \to \quad 1 \\ \text{それ以外} & \to \quad f(n-1) + f(n-2) \end{cases}$$

を計算する関数を定義して実行している例である．

```
(%i1)  f(n) := if n=0 then 1 else if n=1 then 1 else f(n-1)+f(n-2);
(%o1)  f(n) := if n = 0 then 1 else if n = 1 then 1 else f(n - 1) + f(n - 2)
(%i2)  f(22);
(%o2)  28657
```

7.1.1　条件式（述語）

条件として記述できるもの（一部）を表 5 に挙げる．

表 5: 基本的な述語

記述	意味
$X = Y$	X と Y が等しい
$X\#Y$	X と Y は等しくない
$X > Y$	X は Y より大きい
$X >= Y$	X は Y 以上
$X < Y$	X は Y より小さい
$X <= Y$	X は Y 以下
evenp(X)	X は偶数
oddp(X)	X は奇数
P and Q	P かつ Q
P or Q	P または Q
not(P)	P でない

7.2　手続き

文法：　block([局所変数リスト], 式 1, 式 2, ···)

式 1, 式 2, ···
と順番に実行してゆく方法である．局所変数は当該 block を実行する範囲内でのみ有効なものである．block 文実行後は最後の式の値が戻り値となる．

```
(%i3)  block([c:0],print(c),c:c+1,print(c),c:c+1,print(c));
0
1
2
(%o3)  2
```

これは c の値を変えながら print 関数で表示している例である．

7.3 繰り返し

文法 1： for 制御変数:初期値 step 増分 thru 終値 do 式
文法 2： for 制御変数:初期値 step 増分 while 条件式 do 式
文法 3： for 制御変数 in [リスト] do 式

文法 1 の例．c の値を 1 から 5 まで 1 づつ増やしながらそれを印字する．

```
(%i4)  for c:0 step 1 thru 5 do print(c);
0
1
2
3
4
5
(%o4)  done
```

文法 2 の例．c の値を 1 づつ増やしながら c が 5 未満である間，それを印字する．

```
(%i5)  for c:0 step 1 while c < 5 do print(c);
0
1
2
3
4
(%o5)  done
```

文法 3 の例．c にリスト [v,w,x,y,z] の要素を順番に与えながら印字する．

```
(%i6)  for c in [v,w,x,y,z] do print(c);
v
w
x
y
z
(%o6)  done
```

7.4 リスト処理

プログラミングのためのデータ構造としてリストが有用である．

7.4.1 リストの要素の取り出し

リスト L の n 番目の要素を取り出すには，L[n] を評価する．

```
(%i7)  for c:1 step 1 thru 4 do print([s,t,u,v][c]);
s
t
u
v
(%o7)  done
```

これは，リスト [s,t,u,v] の要素を順番に表示する例である．

7.4.2 リストの合成と分解

● **cons**　リスト L の先頭に要素 a を追加したものを得るには，cons(a,L) を評価する．

```
(%i8)  cons(a,[b,c,d]);
(%o8)  [a, b, c, d]
```

これは，リスト [b,c,d] の先頭に要素 a を追加したものを得る例である．

● **endcons**　要素 d をリスト L の末尾に追加するには，endcons(d,L) を評価する．

```
(%i9)  endcons(d,[a,b,c]);
(%o9)  [a, b, c, d]
```

これは，リスト [a,b,c] の末尾に要素 d を追加したものを得る例である．

● **first**　リスト L の先頭の要素を取り出すには，first(L) を評価する．

```
(%i10) first([a,b,c,d]);
(%o10) a
```

これは，リスト [a,b,c,d] の先頭の要素を得る例である．

● **rest**　リスト L の先頭の要素を取り除いたリストを得るには，rest(L) を評価する．

```
(%i11) rest([a,b,c,d]);
(%o11) [b, c, d]
```

これは，リスト [a,b,c,d] の先頭の要素を取り除いたリスト得る例である．

● **last**　リスト L の末尾の要素を得るには，last(L) を評価する．

```
(%i12) last([a,b,c,d]);
(%o12) d
```

これは，リスト [a,b,c,d] の末尾の要素を得る例である．

● **append**　リスト L1 と L2 を連結するには，append(L1,L2) を評価する．

```
(%i13) append([a,b],[c,d]);
(%o13) [a, b, c, d]
```

これは，リスト [a,b] と [c,d] を連結したリストを得る例である．

7.4.3　リストの長さの算出

リスト L の長さを算出するには，length(L) を評価する．

```
(%i14) length([a,b,c,d,e]);
(%o14) 5
```

これは，リスト [a,b,c,d,e] の長さを算出した結果，5 が得られている例である．

7.4.4　空リストの判定

リスト L が空であるかどうかを判定するには，emptyp(L) を評価する．L が空の場合は true を，空でない場合は false を返す．（下記参照）

```
(%i15) emptyp([]);
       emptyp([a]);
(%o15) true
(%o16) false
```

7.5　プログラムの作成と読込み（wxMaxima）

関数の定義や値の設定などをテキストファイルの形で保存しておき，それを Maxima に読み込んで使用することができる．

下のようなプログラム（テキスト形式）が fb.mac というファイル名で保存されているとする．wxMaxima のフロントエンドウィンドウのメニュー「ファイル」から「パッケージ読込み」を選択すると，ファイルを選択するダイアログが表示される．ここでプログラムのファイルを選択して「開く」ボタンをクリックするとプログラムが読み込まれる．

プログラム：fb.mac

```
1 /* フィボナッチ数 */
2 f(n) :=
3     if n=0 then 1 else
4         if n=1 then 1 else
5             f(n-1)+f(n-2);
```

このプログラム[5] を読み込んだ後，関数 f が使用できる．実行例を下に示す．

```
(%i1)  load("D:/fb.mac")$
(%i2)  f(0);f(1);f(2);f(3);f(4);
(%o2)  1
(%o3)  1
(%o4)  2
(%o5)  3
(%o6)  5
```

この実行例の冒頭で load 関数が呼び出されているが,「ファイル」メニューから「パッケージ読込み」を選択する操作によって自動的に実行されるものである．

[5] このプログラムのアルゴリズムは最適ではない．計算に必要となる記憶域と時間に関して更に効率の良いアルゴリズムが存在する．

8　その他

8.1　Maxima ⇔ Lisp 処理系の切り替え

Maxima は Lisp 処理系（Common Lisp）の上に構築されており，Lisp 処理系の機能を使うことができる．具体的には Maxima の式 to_lisp() を評価すると Lisp 処理系に制御が移る．また Lisp 処理系から Maxima に戻るには Lisp の S 式 (to-maxima) を評価する．

```
(%i1)  to_lisp();
Type (to-maxima) to restart, ($ quit) to quit Maxima.

MAXIMA>  (car '(a b c d e));
A

MAXIMA>  (to-maxima)
Returning to Maxima
(%o1)  true

(%i2)  a+a;
(%o2)  2a
```

上の例の (%i1) で to_lisp() を評価して Common Lisp 処理系に切り替えている．この結果，プロンプトが

MAXIMA >

に変わり，Common Lisp の式の入力を受け付ける．この状態で S 式 (to-maxima) を評価すると Maxima に制御が戻り（上記例の%i2〜%o2），以後は Maxima の式を受け付ける．

索 引

「Maxima 入門」

－ 高機能なフリーの数式処理システム

著者：中村勝則

発行：2019 年 8 月 15 日

本書の最新版と更新情報

本書の最新版と更新情報を，プログラミングに関する情報コミュニティ Qiita で配信しています.

→ https://qiita.com/KatsunoriNakamura/items/4075337a4b8e355ce384

上記 URL の QR コード

● 連絡先

nkatsu2012@gmail.com

中村勝則