Maxima 入門ノート 1.2.1

中川義行

Copyright (c) 2005 中川義行 (Yoshiyuki NAKAGAWA).

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

GNU Free Documentation License については,

```
http://www.gnu.org/licenses/fdl.html
```

において全文を閲覧することができます。また，日本語を母語とする者が GNU Free Documentation License をより良く理解するための参考として,

```
http://www.opensource.jp/fdl/fdl.ja.html
```

こちらも一読いただければ嬉しく思います。

e-mail address : kyo-ju@iris.eonet.ne.jp

謝辞

ささやかな備忘録だったノートを再整理する機会を下さった濱田龍義氏に深く感謝致します。また，至らぬ文書であるにも関わらず，多くの方に目を通していただきましたこと，心から感謝致します。さらには，筆者が見落としていた誤植などの訂正指摘を受けましたこと感謝致します。それから Maxima の開発や普及に携わっておられる全ての方々に感謝致します。

目 次

はじめに **v**

第 1 章 Maxima の基本 **1**
1.1 Maxima の起動と実例，そして終了 1
1.2 オンラインヘルプの利用 5
1.3 四則演算など 7
1.4 関数計算 9
1.5 変数と関数の定義 11
1.6 章末練習問題 13

第 2 章 グラフの描画 **17**
2.1 平面的なグラフ (陽関数の場合) 17
2.2 平面的なグラフ (媒介変数表示の場合) 19
2.3 立体的なグラフ (陽関数の場合) 21
2.4 立体的なグラフ (媒介変数表示の場合) 22
2.5 gnuplot との連動 23
2.5.1 複数の立体的グラフを重ねて表示させる 24
2.5.2 等高線グラフを作成する 25
2.5.3 LaTeX にグラフを取り込ませる 28
2.6 章末練習問題 28

第 3 章 数式の操作 **31**
3.1 多項式の操作 31
3.1.1 多項式の展開 31
3.1.2 多項式の因数分解 32
3.2 有理式の操作 32
3.2.1 有理式の通分 32
3.2.2 有理式の部分分数展開 33
3.3 方程式を解く 34
3.4 方程式の近似解 36
3.5 章末練習問題 39

第 4 章 微分と積分 **43**
4.1 極限と微分 43
4.2 級数と積分 46
4.3 微分方程式 50
4.4 章末演習問題 52

第 5 章　リストの扱い **55**
5.1　はじめに . . . 55
5.2　データのリスト . . . 55
5.3　ベクトルの演算 . . . 58
5.4　行列の演算 . . . 60
5.5　章末演習問題 . . . 67

第 6 章　プログラミングの初歩 **69**
6.1　函数の作成 . . . 69
6.2　分岐構造 . . . 70
6.3　反復構造 . . . 73
6.4　ブロック化 . . . 75
6.5　外部ファイルの利用 . . . 77
6.5.1　手順や結果の記録 . . . 77
6.5.2　データファイルの利用 . . . 78
6.5.3　統計処理への応用例 . . . 80
6.6　章末演習問題 . . . 81

付 録 A　Mathematica から Maxima へ **83**
A.1　演算と数値 . . . 83
A.2　代数 . . . 83
A.2.1　方程式を解く . . . 83
A.2.2　多項式の操作 . . . 83
A.2.3　式の簡約化 . . . 84
A.2.4　複素数 . . . 84
A.3　リストと行列 . . . 84
A.3.1　リストや行列の作成 . . . 84
A.3.2　行列の演算 . . . 84
A.4　三角関数と指数関数 . . . 85
A.4.1　三角関数 . . . 85
A.4.2　指数と対数 . . . 85
A.4.3　双曲線関数 . . . 85
A.5　微分積分 . . . 85
A.5.1　通常の演算 . . . 85
A.5.2　微分方程式 . . . 86
A.5.3　変換 . . . 86
A.6　その他の関数 . . . 86
A.6.1　整数関数 . . . 86
A.6.2　特殊関数 . . . 86
A.7　グラフィクス . . . 87

付 録 B　インストールについて **89**
B.1　Maxima のインストール . . . 89
B.1.1　Linux の場合 . . . 89
B.1.2　MacOS X の場合 . . . 90

B.1.3 Microsoft Windows の場合 . . . 92
B.2 gnuplot のインストール . . . 92
B.2.1 Linux の場合 . . . 93
B.2.2 MacOS X の場合 . . . 93

付 録 C トラブルシューティング **95**
C.1 システム絡みのトラブル . . . 95
C.2 操作方法に関するトラブル . . . 97
C.3 函数に関するトラブル . . . 98
C.4 その他のトラブル . . . 99

付 録 D 演習問題の解答例 **101**
D.1 Maxima の基本 . . . 101
D.2 グラフの描画 . . . 106
D.3 数式の操作 . . . 116
D.4 リストの扱い . . . 134
D.5 プログラミングの初歩 . . . 148

はじめに

数学は，自然現象や社会現象を定量的に扱いたいときには，結構便利な道具ですが，厄介なのが**数式の計算**とか**グラフの描画**などでしょう。折角，面白いアイディアも，式が複雑過ぎて，人間の計算力が追いつかないということもしばしばあることでしょう。ならば，計算とかグラフ描画といった，機械作業でできるところは，全部コンピュータに任せ，人間は一番面白い理論のところだけを考えたいわけです。

計算やグラフ描画をコンピュータに任せるアプリケーションとしては，この頃では表計算ソフトウェアの一種である Microsoft Excel やその互換ソフトウェア[1]が有名でしょう。Microsoft Excel は，常用するアプリケーションとしては，大事なポイントを押さえています。

- ユーザが圧倒的に多い。
- GUI の操作性が快適である。
- 比較的，安価で購入できる。

しかし，研究に使おうとすると，少々問題点があります。

- 誤差の扱いが杜撰で，浮動小数の計算結果が信用できない。
- 多少，複雑な計算を行うには，マクロを組む必要がある。
- 毎回自動再計算しているので，大量のデータを扱うと操作性が異様に悪くなる。

そういう意味では，研究支援のアプリケーションとしては，ある意味，致命的な問題を抱えているわけです。それでも，最も誤差が積み重なるはずの反復計算でも，**有効数字 3 桁**は保証しているので，それほど正確な数値が必要なければ，十分な機能と言えます[2]。

これから紹介します Maxima というソフトウェアは，数式処理ソフトウェアと呼ばれ，同種の市販ソフトウェアに *Mathematica* や Maple などがあります。この大きな違いは，表計算ソフトウェアは数値計算を主目的としているので，常に数値近似であることに対して，Maxima は数学的にも厳密な計算を主目的としていることが挙げられます。

例えば，関数 $f(t) = \sin^3(t)$ の t による導関数を取り，その t が $\frac{\pi}{5}$ のときの値を求めたいとしましょう。表計算ソフトウェアであれば，数式そのままでは扱えませんから，微分計算そのものは人間の手で行うか，微小区間の差分を以って近似値とするしかありません。ところで，我々が微分を行うとき毎回，導関数の定義 $f'(t) = \lim_{h\to\infty} \frac{f(t+h) - f(t)}{h}$ に基づいて極限計算をしているでしょうか。答えは否です。我々が $f(t) = \sin^3(t)$ を t で微分するときは，まず文字列パターン $\square^3$ を見抜いて，それを $3\square^2\boldsymbol{d}\square$ に文字列置き換えをしています。次に，$\boldsymbol{d}\square$ を求めるために文字列パターン $\sin(t)$ を見抜いて，それを $\cos(t)$ に文字列置き換えをしています。そして，最後に t に $\frac{\pi}{5}$ を代入して数値評価しています。だから厳密な計算が可能なのです。

[1] KNOPPIX/Math にも OpenOffice の "Calc" が準備されています。

[2] 大量のデータを記録できる関数電卓として，私も重宝しています。正確な計算には使えなくても，あれこれ数値実験を行って目安を立てる程度には，十分ですから。

Maxima などの数式処理ソフトウェアとは，大雑把に言えば，このような文字列処理作業をするソフトウェアです。勿論，機械が文字列パターン探しと置き換えを行うのですから，人間よりも見落としが少ないので，学習や研究に大きな助けとなるはずです。と申しましても，市販の数式処理ソフトウェアは使いやすい代わり，コンピュータよりも高価[3]なので，購入が躊躇われることでしょう。そこで，無料で利用できる数式処理ソフトウェアの Maxima はいかがでしょうか。市販品に比べて，ユーザインターフェイスが見劣りしますし，ユーザも決して多くありません。また，アップデートなども自分でチェックしないといけないなどの不便さもありますが，数式処理ソフトウェアとしては引けをとりません。

今日，自然科学は言うに及ばず，社会科学においても，高校程度の数学の知識や感覚は不可欠な時代となりました。しかし，申し上げにくいことながら，平成 17 年現在の日本では，工学部の平均的大学生で，およそ高校 1 年生程度の数学力，同じく経済学部の平均的大学生では，およそ中学 2 年生程度の数学力しかないのが現状でございます。ですから，例えば今の大学生に不意に高校入試の数学の問題を解かせますと，およそ半数が不合格になることでしょう。そのような学生に対しても，例えば，微分方程式を多く盛り込んだ理論を教えなければならないとなると，教える側としてはかなり躊躇いがあります。と申しましても，大学教員はそれが仕事ですから，現在の講義時間の約 10 倍ほどの時間さえ取れるのでしたら，中学の内容にまで戻って，教えられないわけでもありません。しかし，これは現実問題として不可能に近いと言わざるを得ません。

そこで，補助的に Maxima を使用することで，数式計算とグラフ描画に費やす時間を節約することができます。その分，より深く面白いトピックスを講義する時間を得ることができます。勿論，この方法にも重大な問題点があります。人間の本質は，**怠け者**だということです。自分が学びたいことや知りたいことには貪欲でありますが，その為に補助的に学ばねばならないことや知らねばならないことがあっても，それ自体に興味が向かなければ極力避けたくなるものです。そして，少なからぬ学生にとって，避けられるものなら避けたいと思っている[4]のが数学です。そんな彼らに，Maxima を与えることは，下手をすると，それこそコンピュータ無しには，教科書も読めない数学力になってしまうでしょう。まさに，小学 1 年生に電卓を与える危険性と同じことです。教育補助ツールとして Maxima を採用するのでしたら，その点は慎重に準備せねばなりません。その危険性を考慮しても，なお Maxima は有益なツールかと思われます。

豊かに広がる Maxima の世界の，ほんの入り口までででございますが，これより皆様をご案内申し上げたいと思います。そして，皆様のお供をいたしますのは，私，中川でございます。

平成 17 年 10 月 26 日，自宅にて記す。

---

[3] 2005 年 10 月現在で，例えば *Mathematica* であれば，学生版が 30,450 円，通常版が 407,400 円です。学生版も卒業後，1 年間は使用可能ですが，その期間内に 126,315 円を支払えば，通常版にアップグレードできます。Maple であれば，学生版が 21,000 円，通常版 207,900 円です。資金さえ潤沢にあれば，このような市販品の方がアップグレードなどの管理が楽です。

[4] 例えば，経済学部の学生の大半が「経済学部へ行けば，全く数学を勉強しなくて済むと思ったから，進学してきたのに」と謎の愚痴をこぼします。こんな誤解が，何故，日本の高校生に蔓延しているのでしょうねえ。

第1章　Maxima の基本

1.1　Maxima の起動と実例，そして終了

Maxima は，様々な OS で起動する，アプリケーションの一種です。ですから，各 OS の一般的なアプリケーションの起動方法で起動します。

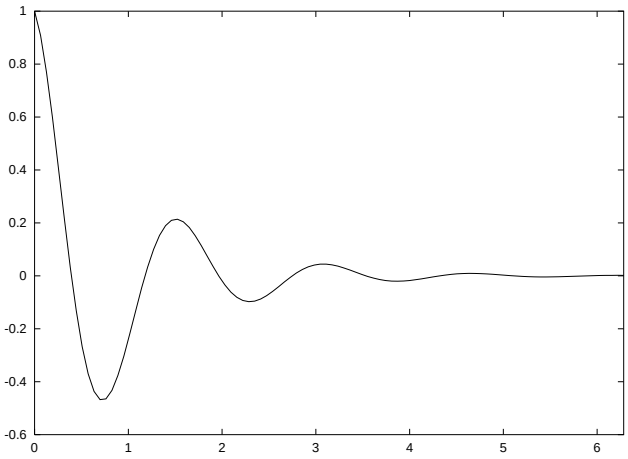
一般的なインストール方法[1]に従っているのであれば，各 OS で次のようになります。

- Linux や MacOSX であれば，シェルから `xmaxima &` とタイプインします。Linux Zaurus であれば，ターミナルから `lisp.run -M maxima-clisp.mem` と入力します。
- Microsoft Windows であれば，スタートボタンから プログラム → Maxima-x.x.x → XMaxima x.x.x と選びます。この x.x.x はプログラムのバージョン番号です。

では早速，Maxima を使ってみましょう。コマンドの説明については後の章に回します。また，出力結果は，Microsoft Windows 版のものに従っています[2]。

まずは例えば，物理学や工学では有名な，減衰振動[3]の一種である関数 $x = e^{-t}\cos(4t)$ のグラフを描いてみましょう。

```
(%i1)  plot2d(exp(-t)*cos(4*t),[t,0,2*%pi]);
```




```
(%o1)
```

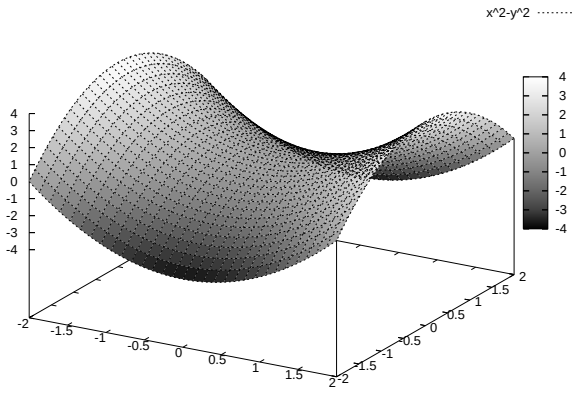
[1] 付録にまとめてあります。

[2] 筆者個人的としては，Linux 版 (KNOPPIX/Math, Linux Zaurus) の方が使いやすいので，そちらを愛用しています。しかし，OS のユーザの多さから，このドキュメントでは敢えて Microsoft Windows 版にしました。

[3] デパートなどの扉で，ある程度開いて手を離すと，前後にゆらゆら揺れながら閉まっていくものがあります。あの動きをイメージしてもらうといいでしょう。

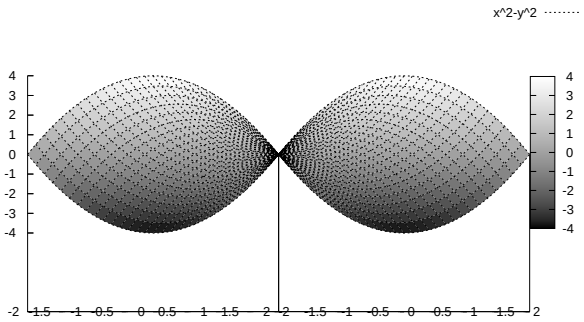
最後のセミコロン (;) は，Maxima に入力の区切りを伝えるものなので，忘れないようにしてください。また，立体的なグラフとして例えば，$z = x^2 - y^2$ のグラフを描いてみましょう。

```
(%i2)  plot3d(x^2-y^2,[x,-2,2],[y,-2,2],[grid,30,30]);
```




```
(%o2)
```

このグラフをクリックしたままドラッグしてみてください。



自由な視点で描くことができることがお分かりいただけると思います。グラフを描くだけではありません。限界があるというものの，様々な方程式を解くことができます。例えば，簡単なところで x の 2 次方程式 $x^2 + ax + b = 0$ を解かせてみましょう。

```
(%i3)  solve(x^2+a*x+b=0,x);
                                  2                       2
                            sqrt(a  - 4 b) + a      sqrt(a  - 4 b) - a
(%o3)              [x = - --------------------, x = --------------------]
                                  2                       2
```

このように，文字変数を含んだままでも，お構いなしに解いてしまいます。また，方程式の解が複雑であれば数値近似させることもできます。例えば x の方程式 $x^3+3x-7=0$ を厳密に解いてみます。

```
(%i4)  solve(x^3+3*x-7=0,x);
                                                    sqrt(3) %i   1
                                                    ---------- - -
           sqrt(53)   7 1/3    sqrt(3) %i   1         2        2
(%o4) [x = (-------- + -)    (- ---------- - -) - -----------------,
              2       2            2        2     sqrt(53)   7 1/3
                                                (-------- + -)
                                                    2       2
                                                  sqrt(3) %i   1
                                                - ---------- - -
           sqrt(53)   7 1/3  sqrt(3) %i   1         2        2
       x = (-------- + -)    (---------- - -) - -----------------,
              2       2          2        2     sqrt(53)   7 1/3
                                              (-------- + -)
                                                  2       2
           sqrt(53)   7 1/3           1
       x = (-------- + -)    - ------------------]
              2       2        sqrt(53)   7 1/3
                              (-------- + -)
                                 2       2
```

これでは，具体的にどれくらいの値なのか皆目見当がつかないので，可能な限り展開して数値近似してみます。

```
(%i5)  float(expand(solve(x^3+3*x-7=0,x)));

(%o5)   [x = - 2.117364769746451 %i - .7031437899802674,
         x = 2.117364769746451 %i - .7031437899802674, x = 1.406287579960535]
```

これで一つの実数解 $x \fallingdotseq 1.406287579960535$ と一組の共役複素数解 $x \fallingdotseq -0.7031437899802674 \pm 2.117364769746451i$ があることが分かります。

方程式を解くばかりではなく，微分や積分の計算もできます。例えば，x^x の微分や，$\dfrac{1}{x^3+1}$ の積分を求めてみましょう。

```
(%i6)  diff(x^x,x);
                                      x
(%o6)                                x  (log(x) + 1)


(%i7)  integrate(1/(x^3+1),x);
```

```
                                      2 x - 1
                          2         atan(-------)
                       log(x  - x + 1)    sqrt(3)    log(x + 1)
(%o7)                - --------------- + ---------- + ----------
                              6           sqrt(3)         3
```

また, 微分方程式を解くことができます。例えば，空気抵抗を考慮した自然落下の運動方程式 $mx''(t) = mg - rx'(t)$, $x'(0) = 0$, $x(0) = h$ を解いてみましょう。

```
(%i8)  atvalue(x(t),t=0,h);

(%o8)                                  h


(%i9)  atvalue(diff(x(t),t),t=0,0);

(%o9)                                  0


(%i10)  desolve(m*diff(x,t,2)=m*g-r*diff(x,t),x,t);

                                    r t
                                  - ---
                          2         m                2       2
                       g m  %e          g m t    h r  - g m
(%o10)          x(t) = ------------ + ------ + ------------
                             2           r           2
                            r                       r
```

これも方程式同様，文字変数を含んだままで解くことができます。
さらに数学量として，ベクトルや行列を扱うことができます。

```
(%i11)  matrix([1,-2],[2,1]).[2,3];

                                    [ - 4 ]
(%o11)                              [     ]
                                    [  7  ]


(%i12)  expand(determinant(matrix([1-t,t,1],[-t,-1,t-1],[1,1-t,t])));

                                      2
(%o12)                              6 t  - 6 t + 2
```

このように，フリーソフトウェアながら，豊かな機能を有していることがご理解いただけるかと思います。しかし，どれほど優れたソフトウェアも，使い方次第で益にも害にもなります。是非，有益に活用していただきたいものです。

続いて Maxima の終了方法を説明します。何故なら，暴走でなくとも，計算によっては長時間の計算になる場合があるので，そうなると無駄に時間を過ごすことになりかねないからです。特に重要なのは，マシンの能力を超える計算を要求してしまったときの，安定した停止方法です。

あまりにもおかしいと思ったら，Ctrl + g を押すと，

```
Maxima encountered a Lisp error:

 Console interrupt.

Automatically continuing.
To reenable the Lisp debugger set *debugger-hook* to nil.
```

と出てくるはずです。これでまた計算方法を工夫することができます。

計算によっては，Ctrl + g で止まらないことがあります。その場合は，オペレーティングシステムから，直接強制停止させましょう。

- Linux または MacOSX であれば，シェルで `ps auxc | more` でプロセスの様子を見て，maxima と xmaxima の PID 番号を確認後，maxima の方から順に `kill -KILL PID 番号` で強制終了させることができます。Linux ZAURUS の場合は，プロセス名が `lisp.run` になっているので注意してください。
- Microsoft Windows NT/2000/XP であれば，Ctrl + Alt + Delete キーでセキュリティパネルを出して，タスクマネージャを起動させます。そのタスクマネージャのプロセスタグで，maxima.exe, xmaxima.exe の順に，プロセスの終了ボタンで停止させましょう。
- Microsoft Windows 98/Me であれば，強制終了もできるのですが安定性の問題で，素直に再起動することをお勧めします。Ctrl + Alt + Delete キーを 2 回押して再起動しましょう。

Maxima を終了させるときに何か計算中であれば，必ずこのように計算プロセスが停止したのを確認してから，通常のアプリケーションの終了で終わるようにしてください。計算中に Maxima を終了させると，**Maxima の計算プロセスが残存**してしまい，コンピュータの性能を著しく下げる可能性があります。

1.2 オンラインヘルプの利用

次にオンラインヘルプの使い方を説明します。Maxima のヘルプは，すべて英語で記されていますが，函数[4]の書式などを辞書のように引くことができます。XMaxima であれば，下にでているヘルプがウェブブラウザ的で使いやすく本格的ですが，シェル版や作業スペースであれば，コマンドで簡単に引く方法もあります。 書式は，

```
?  知りたい函数名;
```

で知ることができます。例えば，陽関数 $z = f(x, y)$ で定義された曲面を描く函数 `plot3d` について調べてみましょう。

```
(%i13)  ?  plot3d;
```

[4]数学の function は「関数」で，Maxima のコマンドとしての function は「函数」で書くことにします。

```
Info from file C:/PROGRA~1/MAXIMA~1.2/info/maxima.info:
 -- Function: plot3d (<expr>, <x_range>, <y_range>, ..., <options>, ...)
 -- Function: plot3d ([<expr_1>, <expr_2>, <expr_3>], <x_range>,
          <y_range>, ..., <options>, ...)
          plot3d (2^(-u^2 + v^2), [u, -5, 5], [v, -7, 7]);
     plots ‘z = 2^(-u^2+v^2)’ with ‘u’ and ‘v’ varying in [-5,5] and
     [-7,7] respectively, and with <u> on the x axis, and ‘v’ on the y
     axis.

     (説明の英文だけで 2 ページ近いので途中省略)

     See ‘plot_options’ for more examples.

(%o13)                                  FALSE
```

英語が苦手な人は泣きそうになりますが，そこは腹を括って解読，もとい，翻訳しましょう。各オプションの働きもすべて同じようにして，オンラインヘルプで知ることができます。粗い訳でよければ，オンライン機械翻訳サービスに通せばいいでしょう。例えば，

```
http://www.infoseek.co.jp/Honyaku/
```

などをご利用ください。また，Maxima のバージョンが，少々古いものでもよければ，

```
http://www.bekkoame.ne.jp/~ponpoko/Math/maxima/maxima_toc.html
```

に，ぽんぽこ氏による日本語訳が提供されています。

このように辞書のように函数そのものの説明を調べる以外に，函数名の一部分しか思い出せないときに，その一部分を含む函数名の一覧を作る検索もできます。例えば，函数名のどこかに `plot` が付くものを全部列挙してもらいましょう。

```
(%i14)  ?  plot;

 0: (maxima.info)Plotting.
 1: Definitions for Plotting.
 2: openplot_curves :Definitions for Plotting.
 3: plot2d :Definitions for Plotting.
 4: plot2d_ps :Definitions for Plotting.
 5: plot3d :Definitions for Plotting.
 6: plot_options :Definitions for Plotting.
 7: set_plot_option :Definitions for Plotting.
Enter space-separated numbers, ‘all’ or ‘none’:
```

と，`plot` を含む関数名や変数名の一覧が表示され，そのうちどれを表示させるかを指示します。複数見たい場合は，`1 3;` のようにその番号をスペースで区切って入力します。すべて見たい場合は `all;` と入力します。特に見るものがなければ `none;` と入力します。

```
Enter space-separated numbers, 'all' or 'none': 7;
      Info from file C:/PROGRA~1/MAXIMA~1.2/info/maxima.info:
       -- Function: set_plot_option (<option>)
           Assigns one of the global variables for plotting.  <option> is
           specified as a list of two or more elements, in which the first
           element is one of the keywords on the 'plot_options' list.

           'set_plot_option' evaluates its argument.  'set_plot_option'
           returns 'plot_options' (after modifying one of its elements).

           See also 'plot_options', 'plot2d', and 'plot3d'.

           Examples:

           (長めの具体例が続くので以下省略)

     (%o14)                                  FALSE
```

まだ Maxima の日本語による解説書はほとんど存在しません[5]ので，この英語のヘルプが頼りです。

1.3 四則演算など

Maxima での四則演算および指数乗は，一般の表計算ソフトウェアと同様に，加法が +，減法が -，乗法が *，除法が /，そして指数乗が ^ で記述できます。

算法の優先順位も自動的に判断されますので，必要に応じて ( ) を入れます。なお，何段階になっても優先順位を指定する括弧はすべて ( ) を用います。詳細は後で必要に応じて述べますが，[ ] は範囲の指定やリスト表現に利用されますので注意が必要です。では実際に計算してみましょう。

```
     (%i15)  3+5;

     (%o15)                                   8


     (%i16)  3-5;

     (%o16)                                  - 2


     (%i17)  3*5;

     (%o17)                                  15


     (%i18)  3/5;
```

[5]2005 年 10 月現在，いくつかの日本語解説書作成のプロジェクトが立ち上がって，進行中です。筆者もその中の 2 つに参加していますが，手に入っている情報だけでも他に 3 つあるようです。

```
(%o18)                                  3
                                        -
                                        5

(%i19)  3^5;

(%o19)                                 243

(%i20)  (4*5+2)/7;

                                       22
(%o20)                                 --
                                       7
```

このように，例えば整数値による分数は分数のまま扱うことによって，可能な限り数学的に厳密な扱いをしようとする特徴があります。 これが具体的にどれ位の値か評価するには，

```
float(数値評価する式);
```

に引き渡します。

```
(%i21)  float((4*5+2)/7);

(%o21)                          3.142857142857143
```

また，指数表現を用いて，任意の精度で数値評価するには，精度を

```
fpprec:桁数;
```

で指定した後，

```
bfloat(数値評価する式);
```

に引き渡します。

```
(%i22)  fpprec:50;

(%o22)                                  50

(%i23)  bfloat(%pi);

(%o23)          3.1415926535897932384626433832795028841971693993751B0
```

```
(%i24)  fpprec:16;

(%o24)                                    16


(%i25)  bfloat(1000!);

(%o25)                          4.023872600770938B2567
```

ここで使った %pi とは，円周率の π を意味します。同様に数学の定数として，自然対数の底 e を表す %e や，虚数単位 $i = \sqrt{-1}$ を表す %i があります。また，bfloat では，必ず末尾に B に続いて，最高桁の指数が表示されます。つまり，上の後者の例であれば，$4.023872600770938 \times 10^{2567}$ ということを意味します。

1.4 関数計算

Maxima には，初等関数を始めとして，様々な超関数まで定義されています。一般の表計算ソフトウェアで採用されている関数名とよく似ていますので，すぐに利用できることでしょう。 利用頻度が高い，代表的なものをあげますと，平方根を求める函数は

```
sqrt(式);
```

絶対値を求める函数は

```
cabs(式);
```

で求めることができます。

```
(%i26)  sqrt(72);

(%o26)                                 6 sqrt(2)


(%i27)  cabs(-1.2);

(%o27)                                    1.2


(%i28)  cabs(2+3*%i);

(%o28)                                 sqrt(13)
```

指数関数は，

```
exp(式);
```

で与えることもできますし，底である e を %e と入力することで，^ を使った形でも記述できます。 対数関数は

```
log(式);
```

で自然対数を与えることができます。つまり，常用対数であれば，log(10.0) で割る必要があります。

```
(%i29)  exp(2.3);

(%o29)                          9.974182454814718


(%i30)  %e^2.3;

(%o30)                          9.974182454814718


(%i31)  log(20.0);

(%o31)                          2.995732273553991


(%i32)  log(2.0)/log(10.0);

(%o32)                          0.301029995663981
```

三角関数であれば，sin(式), cos(式), tan(式) を始めとして，sec(式), csc(式), cot(式) も揃っています。このとき角度は，弧度法 (radian) で与えます。度数法で与えたいときは，角度を π 倍して 180 で割るといいでしょう。

```
(%i33)  cos(%pi/3);

                                        1
(%o33)                                  -
                                        2


(%i34)  tan(45*%pi/180);

(%o34)                                  1
```

逆三角関数は，頭に a を付けた形，つまり，asin(式), acos(式), atan(式), asec(式), acsc(式), acot(式) があります。これらも弧度法で値が返されるので，度数法の値が必要であれば 180 倍して π で割ることで得られます。

```
(%i35)  asin(0.70710678118655)*4;

(%o35)                          3.141592653589807
```

```
(%i36)  float(atan(0.08)*180/%pi);

(%o36)                          4.573921259900861
```

双曲線関数であれば，末尾に h を付けた形，つまり，sinh(式), cosh(式), tanh(式), sech(式), csch(式), coth(式) があります。 同様に逆双曲線関数も，asinh(式), acosh(式), atanh(式), asech(式), acsch(式), acoth(式) があります。

一般の表計算ソフトウェアと同様，函数の中にさらに函数を入れることも可能です。

```
(%i37)  sin(cos(0.6948196907307875));

(%o37)                          0.6948196907307875
```

また，これらの函数も**厳密な計算が可能なものは，必ず厳密なまま**扱います。ですから，数値化の必要があれば float(式); または bfloat(式); に引き渡します。

```
(%i38)  sqrt(5);

(%o38)                                sqrt(5)


(%i39)  float(sqrt(5));

(%o39)                          2.23606797749979
```

1.5 変数と関数の定義

Maxima では，数値など，Maxima で扱えるあらゆる数学的量を記憶させておく変数を利用することができます。それには，

変数名:変数の内容;

で記憶させます。例えば，a という文字変数に，2 という値を記憶させるには，

```
(%i40)  a:2;

(%o40)                                  2
```

とします。同様に a という文字変数に，$\sin(x)$ という数式を記憶させるには，

```
(%i41)  a:sin(x);

(%o41)                                sin(x)
```

とします。また，演算結果を代入することも可能です。

```
(%i42)  a:2;
```

```
(%o42)                                  2
```

```
(%i43)  a:a+2;
```

```
(%o43)                                  4
```

このように，変数として数学的量を記録させられますと，再代入などで変更させられない限り，セッションの間，ずっと記憶しています。そのため，以前使った変数を別の計算のときにうっかり使ってしまうと，勝手に以前の計算結果を代入してしまい，正しい答えを返さない可能性もでてきます。例えば，今，a には 4 という値が入っていますから，それを忘れていたとすると，

```
(%i44)  a*sin(2*t);
```

```
(%o44)                              4 sin(2 t)
```

となって，突然 4 が沸いてくるわ，a はどこかへいってしまうわと，驚くことになります。 こういうときは，a と 4 の関連を消去するために，函数

kill(値を消去する変数名 1, 値を消去する変数名 2,···);

を用います。

```
(%i45)  kill(a);
```

```
(%o45)                               done
```

```
(%i46)  a*sin(2*t);
```

```
(%o46)                              a sin(2 t)
```

となり，期待していたようになります。
定義できるのは変数だけではありません。オリジナルの数学関数も

関数名 (変数名):=関数式;

で定義することができます。この関数に引き渡す変数のことを，引数 (ひきすう) といいます。
例えば，経済学でよく出てくる 2 財におけるコブ・ダグラス型生産関数 $p(x,y,s,t) = x^s y^t$ を定義してみます。

```
(%i47)  p(x,y,s,t):=x^s*y^t;
                                                    s  t
(%o47)                          p(x, y, s, t) := x  y
```

```
(%i48)  p(1.2,0.8,1/3,2/3);
```

```
(%o48)                              .9157713940426656

(%i49)  p(s+1,t-1,u,v);
                                          u        v
(%o49)                               (s + 1)  (t - 1)
```

この定義では，後から引数である x, y, s, t の値や式を代入することで，繰り返し関数を利用することができます。このように定義した関数を，さらに他の関数の定義に利用することができます。

```
(%i50)  q(x,y,z,s,t,u):=p(x,y,s,t)*z^u;
                                                                  u
(%o50)                 q(x, y, z, s, t, u) := p(x, y, s, t) z

(%i51)  q(x,y,z,s,t,u);
                                        s  t  u
(%o51)                                 x  y  z
```

また，計算手順を適切に盛り込むことで，複雑な計算をすることができる函数を製作することができます。これについては，第 6 章「プログラミングの初歩」で解説することにします。 また，この関数を消去する時も，kill を使います。

```
(%i52)  kill(p);

(%o52)                                 done

(%i53)  p(1.2,0.8,1/3,2/3);
                                           1   2
(%o53)                           p(1.2, 0.8, --, --)
                                           3   3
(%o53)                             q(x,y,z,s,t,u);
                                                 u
(%o53)                            p(x, y, s, t) z

(%o53)                               kill(q);

(%o53)                                 done
```

ここまでで，Maxima の機能のごく一部である，関数電卓程度の機能までを説明しました。

1.6 章末練習問題

1. オンラインヘルプ

函数 solve とは何か，オンラインヘルプで調べなさい。余力があれば和訳しなさい。

2. 四則演算と指数乗

次の計算を Maxima で実行しなさい。

(1) $2735 + 1846 - 3099$

(2) $2.35 \times 3.91 - 1.83 \times 7.01$

(3) $\dfrac{6174}{58 + 89}$

(4) $2.21^{\frac{1}{3}} \times 3.44^{1.25}$

(5) $\left(\dfrac{1}{3^{\frac{1}{2}} + 1}\right)^{\frac{1}{3}}$

3. 近似計算

次の値をそれぞれ有効桁数 30 桁で近似しなさい。

(1) $\sqrt{\dfrac{\pi}{2}}$

(2) $\cos(\sin(2.63^3))$

(3) $\log_{10}(42)$

(4) $16 \arctan\left(\dfrac{1}{5}\right) - 4 \arctan\left(\dfrac{1}{239}\right)$

(5) $e^{\sqrt{163}\pi}$

4. 代入計算

(1) 文字変数 t に $\cos(30°)$ の値を代入しなさい。
(2) 文字変数 c に $\sqrt{\dfrac{1+t}{2}}$ の値を代入しなさい。
(3) 文字変数 s に $\sqrt{\dfrac{1-t}{2}}$ の値を代入しなさい。
(4) $2cs$ の値を求めなさい。
(5) 文字変数 t, c, s と値の関係を消去しなさい。

5. 関数定義

(1) 引き数 x を与えると，それを度数法の角度として，対応する弧度法の角度 $\dfrac{x\pi}{180}$ を返す関数 $p(x)$ を定義しなさい。
(2) 引き数 x を与えると，それを華氏の温度として，対応する摂氏の温度 $\dfrac{5}{9}(x - 32)$ を返す関数 $q(x)$ を定義しなさい。
(3) 引き数 x, y, z を与えると，それらを成分にもつ 3 次元ベクトル (x, y, z) の大きさ $\sqrt{x^2 + y^2 + z^2}$ を返す関数 $r(x, y, z)$ を定義しなさい。

(4) 引き数 a, b, c, x, y, z を与えると，それらを成分にもつ二つの 3 次元ベクトル (a,b,c), (x,y,z) がなす角度 $\cos^{-1}\left(\dfrac{ax+by+cz}{r(a,b,c)r(x,y,z)}\right)$ を返す関数 $s(a,b,c,x,y,z)$ を定義しなさい。

(5) 関数 $p(x)$, $q(x)$, $r(x,y,z)$, $s(a,b,c,x,y,z)$ と関数式の関係を消去しなさい。

第2章　グラフの描画

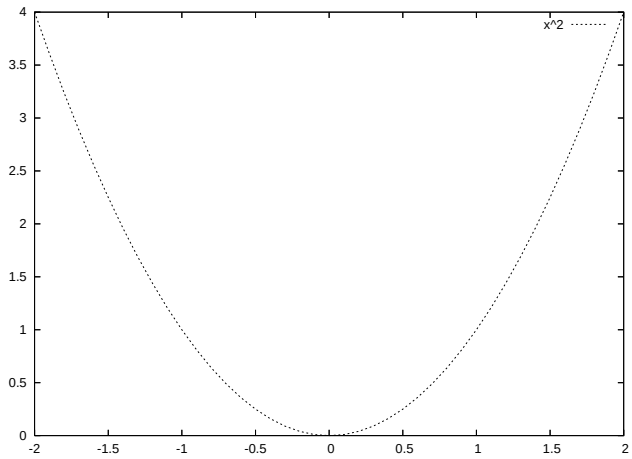
2.1　平面的なグラフ (陽関数の場合)

Maxima の重要な機能の一つとして，関数のグラフを描くことが挙げられるでしょう。諺にも「百聞は一見に如かず」と申します。どれほど多くの学生が，数学の先生の板書する，ゆがみまくったグラフのために苦しんだことでしょう。

表計算ソフトウェアにもグラフを描く機能はありますが，あくまでも数値データに基づいて描くだけで，文字変数を含んだままの関数を直接描くことは不可能ではないまでも，かなりの手間がかかります。それに対して，Maxima 単独では市販の数式処理ソフトウェアほどの多機能性はない[1]にせよ，様々な形で定義された関数のグラフを，手軽に描くことができます。

まずは，最も基本的な $y = f(x)$ 型の陽函数から描いてみましょう。例えば，$y = x^2$ のグラフを $-2 \leqslant x \leqslant 2$ の範囲で描いてみます。

```
(%i54)  plot2d(x^2,[x,-2,2]);
```




```
(%o54)
```

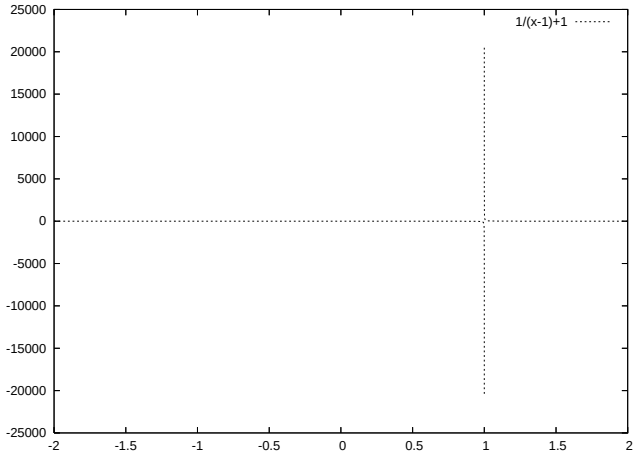
この例のように plot2d 函数でグラフを描きます。書式は，

plot2d(関数式,[定義変数名, 開始値, 終了値], 必要なオプション各種);

です。このグラフを描く原理は，定義範囲を 100 の区間に等分し，その点を陽函数に代入して得られた点を順に線でつないでいます。ですからもし，指定した定義変数の範囲で不連続な点があると，思うような動作をしてくれないことがあります。例えば，$y = \dfrac{1}{x-1} + 1$ などがそうです。

[1] 個人的な見解ですが，市販の数式処理ソフトウェアの中では，Maple が最も優れていると思います。

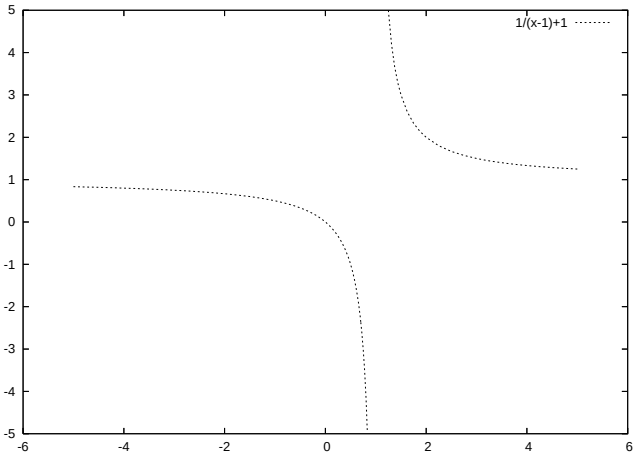
```
(%i55)  plot2d(1/(x-1)+1,[x,-2,2]);
```




```
(%o55)
```

このように何やらよく分からないグラフになります。これは $x = 1$ の近辺で，Maxima が適当なところで計算を打ち切って描くからですが，御覧の通り,「十分大きい」ところまで計算します。そこで，値域の範囲も指定しますと，そこからはみ出す部分は描画しなくなります。

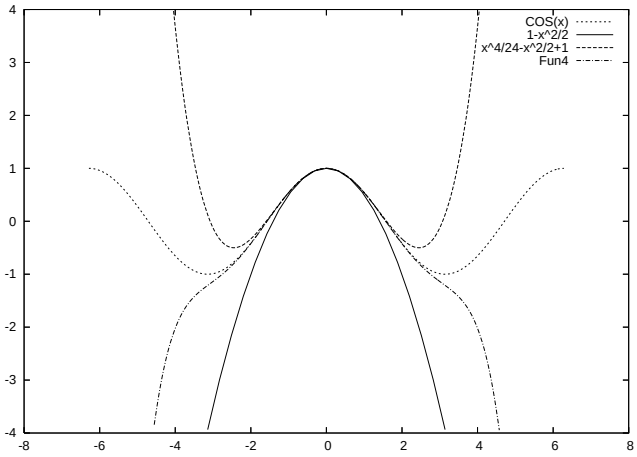
```
(%i56)  plot2d(1/(x-1)+1,[x,-5,5],[y,-5,5]);
```




```
(%o56)
```

さらに，この plot2d 函数は，複数のグラフを重ねて同時に描くことが可能です。plot2d 函数に複数の陽関数を [ ] で括ったリストの形で引き渡すだけで重ねてくれます。例えば，複数のグラフを重ねて描くと，近似関数がどれくらい「よい近似」になっているかが分かります。ここでは，$\cos(x)$ のテイラー多項式近似で，少しずつ次数を上げたものを一度に描いてみます。

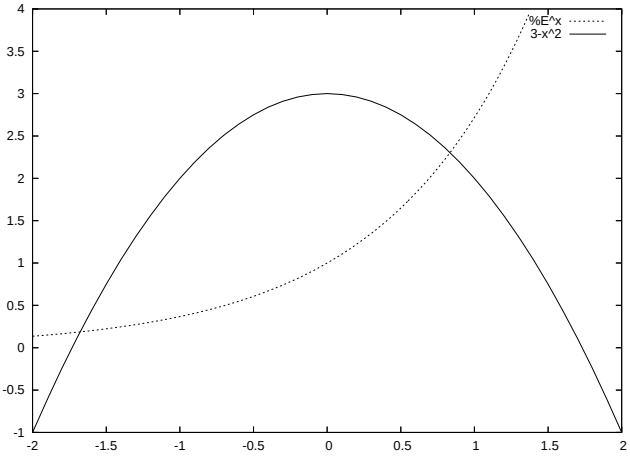
```
(%i57)  plot2d([cos(x),1-x^2/2,1-x^2/2+x^4/24,1-x^2/2+x^4/24-x^6/720],[x,
        -2*%pi,2*%pi],[y,-4,4]);
```



(%o57)

また，方程式の左辺のグラフと右辺のグラフを重ねて描くとその方程式が実数解を持つかどうかの判断にも使えることがあります。

```
(%i58)  plot2d([exp(x),3-x^2],[x,-2,2],[y,-1,4]);
```



(%o58)

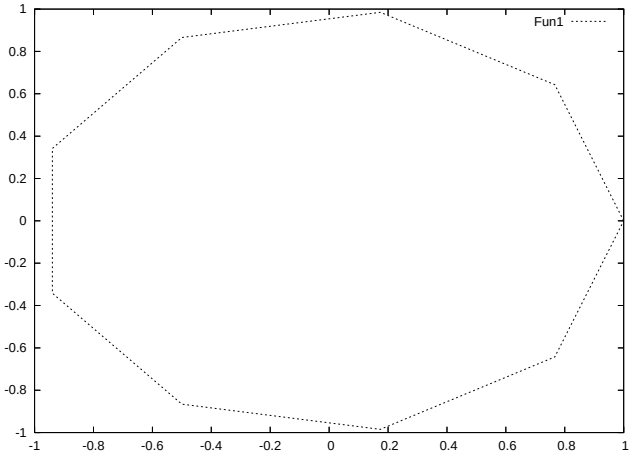
このように $y = e^x$ と $y = 3 - x^2$ のグラフは，2 点で交わるので，実数解を 2 つもつことが分かります。

2.2 平面的なグラフ (媒介変数表示の場合)

次に，2 変数連立の微分方程式や差分方程式の解など，時間によって変化する 2 変数の関係を調べるときに有効な手段として，$\{x = f(t), y = g(t)\}$ 型の媒介変数表示のグラフを描いてみましょう。これも，`plot2d` で描きますが，関数の与え方が少し異なってきます。

例えば，$x = \cos(t), y = \sin(t)$ のグラフを $0 \leqslant t \leqslant 2\pi$ の範囲で描いてみます。

```
(%i59)  plot2d([parametric,cos(t),sin(t)],[t,0,2*%pi]);
```



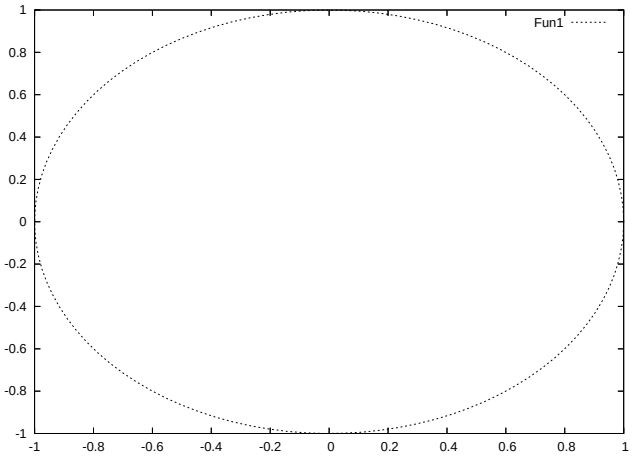

```
(%o59)
```

この例のように plot2d 函数は，$\{x = f(t), y = g(t)\}$ 型の函数のグラフも描くことができます。書式は，

plot2d([parametric,x 成分の関数,y 成分の関数],[媒介変数名, 開始値, 終了値]，必要なオプション各種);

となります。この plot2d 函数が媒介変数表示関数を描く原理は，媒介変数の範囲を 10 の区間に等分し，その点を各座標成分関数に代入して得られた点を線で繋いでいます。ですから，このように荒く見えることも少なくありません。そこで，等分する数を増やすことで，もう少し滑らかにしてみます。それには，オプション [nticks,100] のように分割数を増します。

```
(%i60)  plot2d([parametric,cos(t),sin(t)],[t,0,2*%pi],[nticks,100]);
```



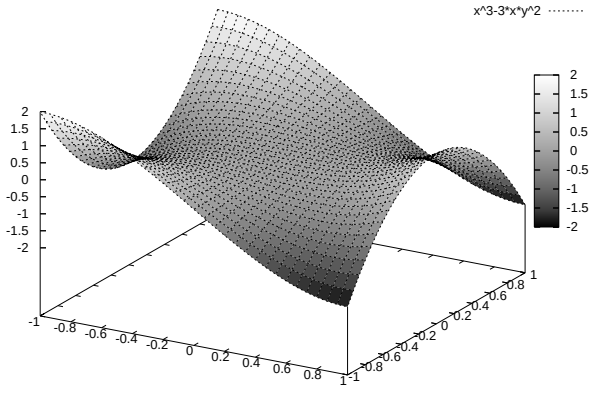

```
(%o60)
```

こうすることで，滑らかなグラフに見えます。

2.3 立体的なグラフ (陽関数の場合)

Maxima は 3 次元内の曲面も無数のポリゴンを使って描くことができます。まずは，$z = f(x, y)$ 型の陽函数から描いてみましょう。例えば，曲面 $z = x^3 - 3xy^2$ のグラフを $-1 \leqslant x, y \leqslant 1$ の範囲で描いてみます。

```
(%i61)  plot3d(x^3-3*x*y^2,[x,-1,1],[y,-1,1]);
```



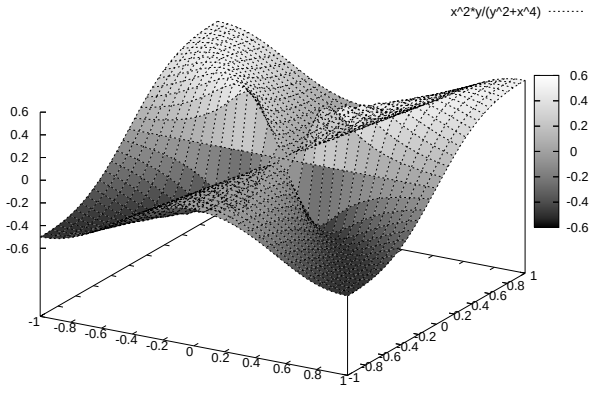

```
(%o61)
```

この例のように plot3d 函数で立体的なグラフを描きます。書式は，

plot3d(関数式，[x 成分変数名，開始値，終了値]，[y 成分変数名，開始値，終了値]，必要なオプション各種);

です。このグラフを描く原理も，定義範囲を 30×30 の区間に等分し，代入して得られた点を四辺形で埋めています。ですからこの函数も，場合によって表示が荒くなります。例えば，$\dfrac{x^2y}{x^4 + y^2}$ がそうです。

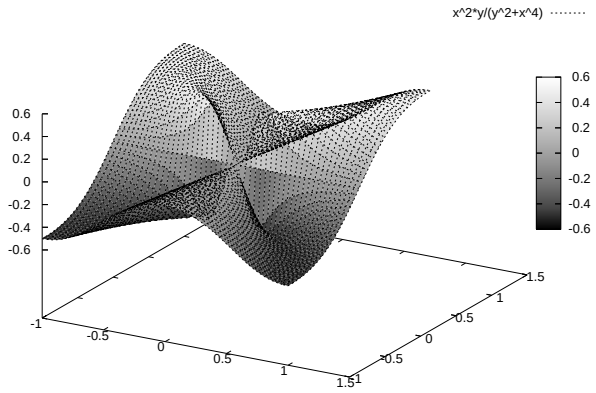
```
(%i62)  plot3d(x^2*y/(x^4+y^2),[x,-1,1],[y,-1,1]);
```




```
(%o62)
```

この荒さを減少させるには，オプション [grid,50,50] のように x, y の分割数を増やすように調節するとそれぞれの四辺形がもっと細かくなって滑らかさが増します。

```
(%i63)  plot3d(x^2*y/(x^4+y^2),[x,-1,1],[y,-1,1],[grid,50,50]);
```




```
(%o63)
```

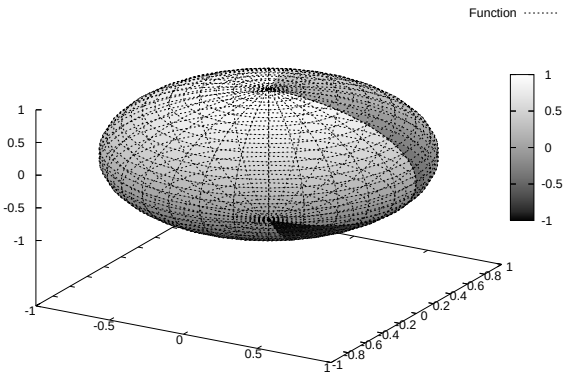
滑らかさが増した代わり，四辺形の境界を表す線のために曲面が真っ青に見えてしまうことがしばしば起こりますので，何度か試行錯誤が必要です。

さて，このような立体的なグラフの形を把握するためには，様々な方向から眺める必要があります。この曲面の一部をクリックでつかんでドラッグしてみましょう。曲面を自在に回転させることができます。

2.4　立体的なグラフ (媒介変数表示の場合)

もっと一般に $x = f(s,t)$, $y = g(s,t)$, $z = h(s,t)$ 型の媒介変数表示のグラフを描いてみましょう。例えば，球面 $x(s,t) = \cos(s)\cos(t)$, $y(s,t) = \cos(s)\sin(t)$, $z(s,t) = \sin(s)$ を $-\dfrac{\pi}{2} \leqslant s \leqslant \dfrac{\pi}{2}$, $0 \leqslant t \leqslant 2\pi$ の範囲で描いてみます。

```
(%i64)  plot3d([cos(s)*cos(t),cos(s)*sin(t),sin(s)],[s,-%pi/2,%pi/2],[t,0,2*%pi]);
```



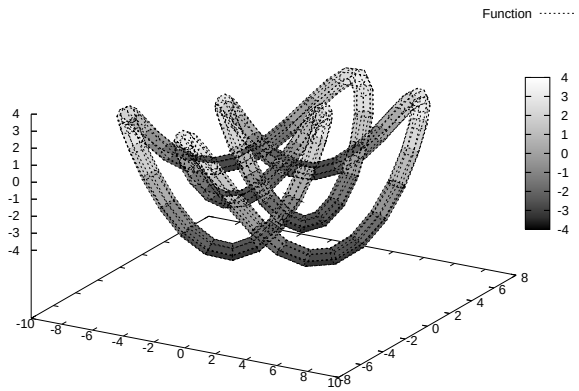
(%o64)

この例のように plot3d 函数は，$\{x = f(s,t), y = g(s,t), z = h(s,t)\}$ 型の函数のグラフも描くことができます。書式は，

plot3d([f(s,t),g(s,t),h(s,t)],[s, 開始値, 終了値],[t, 開始値, 終了値], 必要なオプション各種);

となります。この plot3d 函数の描画原理は，描画領域を 30×30 の区間に等分し，その点を各座標成分関数に代入して得られた点をポリゴンで繋いでいます。後は，オプション [grid,50,50] のように分割数を増やすように調節することで，画像の滑らかさを調節できます。折角ですから，かなり複雑な関数を描画させてみます。

```
(%i65)  plot3d([3*cos(u) + 5*cos(3*u) + (3*(cos(u) + 5*cos(3*u))*cos(v))
        / (2*sqrt(234 + 90*cos(2*u))) - (3*cos(6*u)*(sin(u) +
        5*sin(3*u))*sin(v)) / (2*sqrt(13 + 5*cos(2*u))*sqrt(22 + 5*cos(2*u)
        + 9*cos(12*u))), 3*sin(u) + 5*sin(3*u) + (3*cos(v)*(sin(u) +
        5*sin(3*u))) / (2*sqrt(234 + 90*cos(2*u))) + (3*(5*cos(3*u)
        + cos(5*u) + cos(7*u) + 5*cos(9*u))*sin(v)) / (4*sqrt(13 +
        5*cos(2*u))*sqrt(22 + 5*cos(2*u) + 9*cos(12*u))), 3*sin(6*u)
        - (sqrt(13 + 5*cos(2*u))*sin(v)) / (2*sqrt(22 + 5*cos(2*u) +
        9*cos(12*u)))], [u, 0, 2*%pi], [v, 0, 2*%pi], [grid, 80, 8]);
```



(%o65)

これらを学生にプリントで配布したいときは，次の章を参考にしてください。

2.5 gnuplot との連動

Maxima のグラフ描画は，強力なグラフ描画のソフトウェアである gnuplot を利用しています。これを Maxima と上手に連動させて利用しますと，さらに豊かなグラフ表現が可能になります。

plot2d や plot3d は，この gnuplot で扱えるデータを出力することが可能です。 そのためには，オプションとして [plot_format,gnuplot] を指定します。すると，maxout.gnuplot というファイルが，現在のディレクトリに出力されます。それを gnuplot の方で，

- plot2d で生成したデータなら，
  ```
  load 'maxout.gnuplot'
  ```
- plot3d で生成したデータなら，
  ```
  set hidden3d
  load 'maxout.gnuplot'
  unset hidden3d
  ```

で，それぞれ表示することができます。 set hidden3d とは，立体的なグラフにおいて，陰になって見えない部分は表示しない[(2)]ように指示する[(3)]ことです。では，一部先走ることになりますが，少々テクニカルな使い方を紹介します。

2.5.1　複数の立体的グラフを重ねて表示させる

現在，Maxima には，立体的な曲面を複数重ねて表示させる機能がありません。そこで，各曲面のデータを引き渡して，それらをまとめて gnuplot に描いてもらう手があります。先程の効用曲面に，予算平面 $2x+3y=3$ を同時に描いてみます。

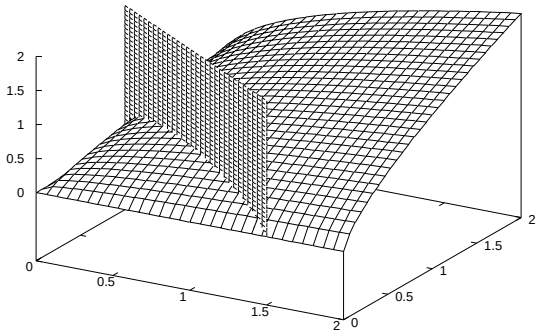
まず，先程の maxout.gnuplot を何か別の名前に書き換えます。ここでは仮に，utility.gnuplot に書き換えたとします。次に，予算平面を媒介変数表示 $x=s$, $y=\dfrac{2-2s}{3}$, $z=t$ と式変形して，$0 \leqslant s \leqslant 1.5$, $0 \leqslant t \leqslant 2$ の範囲で描き出します。

```
(%i66)  plot3d([s,(3-2*s)/3,t],[s,0,1.5],[t,0,2],[plot_format,gnuplot]);

(%o66)
```

次に，テキストエディタを起動させて，utility.gnuplot と maxout.gnuplot の頭にある set pm3d と splot の行を削除します。そして，gnuplot を起動させて，次のように入力します。

```
gnuplot> set hidden3d

gnuplot> splot 'maxout.gnuplot' notitle with lines, 'utility.gnuplot' notitle
         with lines
```

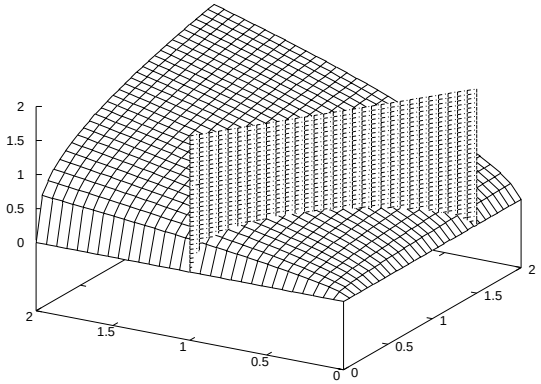
(2) 陰線処理といいます。
(3) 陰線処理解除のオプションは，gnuplot 3.x 以前でしたら，unset hidden3d ではなく set nohidden3d となります。




```
gnuplot> unset hidden3d
```

また，グラフへの視点の位置を変えるには，view に x 軸を中心とする回転角と z 軸を中心とする回転角を度数法でそれぞれ与えます。

```
gnuplot> set view 60, 300

gnuplot> set hidden3d

gnuplot> splot 'maxout.gnuplot' notitle with lines, 'utility.gnuplot' notitle
         with lines
```




```
gnuplot> unset hidden3d

gnuplot> set view 60, 30
```

2.5.2 等高線グラフを作成する

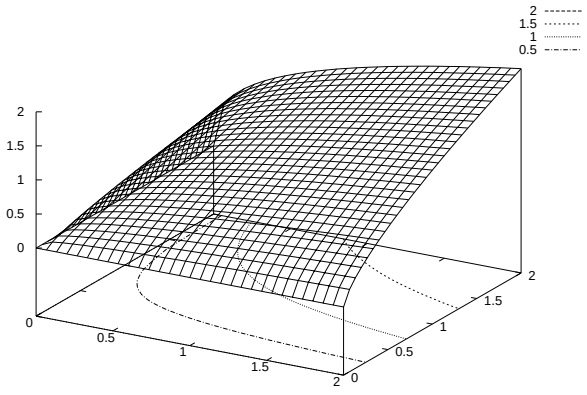
現在，Maxima には，立体的な曲面の等高線を表示させる機能がありません。そこで，データを引き渡して，それに基づいて gnuplot に描いてもらう手があります。例えば，経済学によく出てくるコブ・ダグ

ラス型効用 (生産) 関数 $z = Ax^s y^t$ の等高線を描いてみます。$A = 1$, $s = \dfrac{1}{3}$, $s = \dfrac{2}{3}$ として，$0 \leqslant x, y \leqslant 2$ の範囲で描かせてみます。

```
(%i67)  plot3d(x^(1/3)*y^(2/3),[x,0,2],[y,0,2],[plot_format,gnuplot]);

(%o67)
```

次に，gnuplot を起動させて，次のように入力します。

```
gnuplot> set contour

gnuplot> load 'maxout.gnuplot'
```




```
gnuplot> unset contour
```

このように，contour オプションをつけることで等高線が描かれます(4)。このとき，hidden3d オプションをつけると，等高線の一部が曲面で隠れてしまうのでお勧めしません。

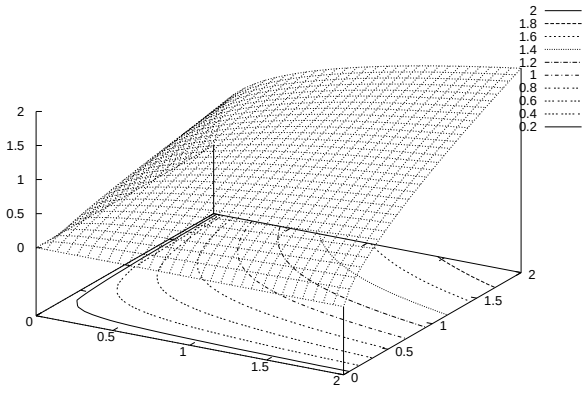
また，等高線の高さレベルも次の 3 種類の方法で自由に変更できます。

- set cntrparam levels auto 等分数
  高さの最大値と最小値の間を，指定された等分数に分割して等高線を描画します(5)。

- set cntrparam discrete 高さリスト
  高さリストは，高さデータをコンマで区切ってリストにします。指定した各高さについて等高線を描画します。

- set cntrparam increment 初期値，増分，終了値
  初期値から終了値まで，指定した増分で高さがリスト化され，その各高さについて等高線を描画します。

```
gnuplot> set contour

gnuplot> set cntrparam levels auto 10
```

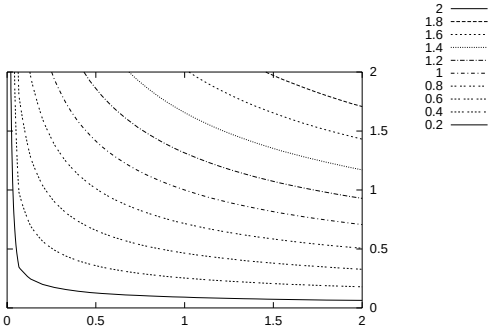



```
gnuplot> load 'maxout.gnuplot'

gnuplot> unset contour
```

さらに，曲面の描画なしで等高線だけを描き，真上から眺めますと，等高線だけのグラフになります。曲面の描画を消すには unset surface と指定[6]します。そして真上から眺めるには，view オプションに 0，0 を指定します。

```
gnuplot> set contour

gnuplot> set cntrparam levels auto 15

gnuplot> unset surface

gnuplot> set view 0, 0

gnuplot> load 'maxout.gnuplot'
```

[4]等高線解除のオプションは，gnuplot 3.x 以前でしたら，unset contour ではなく set nocontour となります。
[5]gnuplot 3.x 以前でしたら，set cntrparam auto 等分数 で十分です。
[6]曲面描画解除のオプションは，gnuplot 3.x 以前でしたら，unset surface ではなく set nosurface となります。




```
gnuplot> set surface

gnuplot> unset contour
```

2.5.3　LaTeX にグラフを取り込ませる

確かに，Maxima の plot2d, plot3d のオプションには，[plot_format, ps] があり，PostScript 形式のグラフを出力が可能ではあります。しかし，LaTeX に取り込むには，必要に応じて視点の変更などの細工を施したいと思うこともあります。そこで，一度，gnuplot に取り込んで細工してから，eps 形式に変換する方法を紹介します(7)。これはあちらこちらで紹介されておりますので，ここでは簡単に説明します。

gnuplot でグラフに満足する細工ができますと，次の手順で eps 形式に変換します。

(1) set output "ファイル名" と入力します。後でファイル形式を把握しやすくするために，ファイル名の末尾に拡張子 .eps とつけておく方がいいでしょう。
(2) set terminal postscript eps と入力します。
(3) load 'maxout.gnuplot' で描画します。
(4) Linux であれば set terminal x11 で，MacOSX であれば set terminal macosx で，Microsoft Windows であれば set terminal windows で出力先を元に戻します。

また，この方法の応用例として，gif 形式や png 形式で出力させ，ウェブページやワードプロセッサなどに取り込む方法があります。そのためには，set terminal postscript eps の代わりに，set terminal gif とか，set terminal png とします。ファイル名の拡張子も，それぞれ .gif とか，.png としておくほうがよいでしょう。

2.6　章末練習問題

1. $y = f(x)$ のグラフ

(1) 関数 $y = x(x-1)^2$ のグラフを $-2 \leqslant x \leqslant 2$ の範囲で描画しなさい。

(7) 先述しましたように，このテキストのグラフはこの方法で作成しています。

(2) 需要曲線 $y = 10 - 0.8x^{0.4}$ のグラフを $0 \leqslant x \leqslant 100$ の範囲で描画しなさい。

(3) 有理関数 $y = \dfrac{x^3 + 2x^2 - 4x - 3}{x + 1}$ のグラフを $-3 \leqslant x \leqslant 3$ の範囲で描画しなさい。

(4) 次の 5 つの指数函数 $y = e^{kx}, (k = -2, -1, 0, 1, 2)$ のグラフを $-2 \leqslant x \leqslant 2$ の範囲で一緒に描画しなさい。

(5) 方程式 $\dfrac{x}{8} = \cos(x)$ が実数解をいくつ持つか求めなさい。

2. $x = f(t)$, $y = g(t)$ のグラフ

(1) 関数 $x = 3\cos(t) + \cos(13t)$, $y = 3\sin(t) + \sin(13t)$ のグラフを $0 \leqslant t \leqslant 2\pi$ の範囲で描画しなさい。

(2) リサージュ曲線 $x = \cos(7t)$, $y = \sin(4t)$ のグラフを $0 \leqslant t \leqslant 2\pi$ の範囲で描画しなさい。

(3) ベルヌーイのレムニスケート $x = \dfrac{\cos(t)}{1 + \sin^2(t)}$, $y = \dfrac{\sin(t)\cos(t)}{1 + \sin^2(t)}$ のグラフを $0 \leqslant t \leqslant 2\pi$ の範囲で描画しなさい。

3. $z = f(x, y)$ のグラフ

(1) 関数 $z = \exp\left(-\dfrac{\sqrt{x^2 + y^2}}{2}\right)\cos\left(\pi\sqrt{x^2 + y^2}\right)$ のグラフを，$-4 \leqslant x, y \leqslant 4$ の範囲で描画しなさい。

(2) シャークの極小曲面 $z = \log\left(\dfrac{\cos(x)}{\cos(y)}\right)$ のグラフを $-\dfrac{\pi}{2} < x, y < \dfrac{\pi}{2}$ の範囲で描画しなさい。

(3) 2 財に対する CES 型効用函数 (生産函数) $u(a, x, y, p, q, r) = a\left(px^{-r} + qy^{-r}\right)^{-\frac{1}{r}}$ を定義して，それを用いて $a = 2$, $p = 2$, $q = 3$, $r = 2$ のグラフを，$0 < x, y \leqslant 2$ の範囲で描画しなさい。

4. $x = f(s, t)$, $y = g(s, t)$, $z = h(s, t)$ のグラフ

(1) トーラス $x = \cos(s)\,(3 + \cos(t))$, $y = \sin(s)\,(3 + \cos(t))$, $z = \sin(s)$ のグラフを $0 \leqslant s, t \leqslant 2\pi$ の範囲で描画しなさい。

(2) アステロイド的球面 $x = \cos^3(s)\cos^3(t)$, $y = \sin^3(s)\cos^3(t)$, $z = \sin^3(t)$ のグラフを $0 \leqslant s, t \leqslant 2\pi$ の範囲で描画しなさい。

(3) ヘンネベルグの極小曲面 $x = 2\sinh(s)\cos(t) - \dfrac{2}{3}\sinh(3s)\cos(3t)$, $y = 2\sinh(s)\sin(t) - \dfrac{2}{3}\sinh(3s)\sin(3t)$, $z = 2\cosh(2s)\cos(2t)$ のグラフを $0.3 \leqslant s \leqslant 0.9$, $0 \leqslant t \leqslant 2\pi$ の範囲で描画しなさい。

5. gnuplot との連動

(1) Maxima で $y = \frac{1}{1+e^{-2x+1}}$ のグラフを，$-2 \leqslant x \leqslant 2$ の範囲で作成し，それを `gnuplot` に取り込みなさい。

(2)　gnuplot を用いて，$z = \sqrt{x^2 + y^2}$ と $z = x + 1$ のグラフを $-1 \leqslant x, y \leqslant 1$ の範囲で一緒に描画しなさい。このとき，交わりである放物線が分かるような視点を探しなさい。

(3)　gnuplot を用いて，$z = \sin(xy)$ の等高線のグラフを $0 \leqslant x, y \leqslant \pi$ の範囲で描画しなさい。このとき，高さレベルを 7 段階用意しなさい。

(4)　上で作成したグラフを，TEX で取り込める eps 形式で出力しなさい。また，gif 形式で出力しなさい。

第3章　数式の操作

3.1　多項式の操作

一概に数式の操作と言っても，いろいろあります。多項式を展開したり因数分解したり，有理関数を通分したり部分分数に分解したり，複数の等式から不要な変数を除去したりなど，どれをとってもミスしやすいものです。そこで，Maxima にそれらの数式操作をさせてみましょう。

3.1.1　多項式の展開

多項式を展開するには，expand 函数に引き渡します。expand 函数の書式は，

```
expand(多項式);
```

です。

```
(%i68)  expand((x^2+x+1)*(x^2-x+1));
                                          2    4
(%o68)                                1 + x  + x


(%i69)  expand((a+b)^5);
                     5      4         3  2       2  3        4    5
(%o69)              a  + 5 a  b + 10 a  b  + 10 a  b  + 5 a b  + b
```

これで展開してくれるのは便利なのですが，項が多くなってくると欲しい係数が分からなくなる可能性があります。 これに対して，求める変数の係数をつむぎだしてくれるのが，ratcoef 函数です。ratcoef 函数の書式は，

```
ratcoef(多項式, 変数);
```

です。

```
(%i70)  ratcoef(a*x^2+b*x^2+c*x+d,x^2);

(%o70)                                    b + a
```

3.1.2　多項式の因数分解

逆に，多項式を因数分解するには，factor 函数に引き渡します。factor 函数の書式は，

```
factor(多項式);
```

です。

```
(%i71)  factor(2*x^2-5*x+2);

(%o71)                          (- 2 + x) (- 1 + 2 x)


(%i72)  factor(a^6-b^6);

                                    2          2    2          2
(%o72)              (a - b)(a + b)(a  - a b + b ) (a  + a b + b )


(%i73)  factor(x^4 + 4);

                                       2            2
(%o73)                        (2 - 2 x + x )(2 + 2 x + x )
```

また，この factor に自然数を与えると，素因数分解をさせることもできます。

```
(%i74)  factor(2520);

                                      3  2
(%o74)                               2  3  5 7
```

3.2　有理式の操作

3.2.1　有理式の通分

ここでいう有理式とは，分母子ともに多項式になっているものを扱います。さて，有理式を取り扱う上での基本は，まず通分することから始まります。

有理式を通分するには，ratsimp に引き渡します。

```
(%i75)  ratsimp(2/(x+1)-1/(x-1));

                                       x - 3
(%o75)                                -------
                                        2
                                       x  - 1
```

また，通分したときの分子だけを取り出すには

```
num(ratsimp(有理式));
```

で，分母だけ取り出すには

```
denom(ratsimp(有理式));
```

でそれぞれ求めることができます。

```
(%i76)  num(ratsimp(2/(x+1)-1/(x-1)));

(%o76)                                  x - 3


(%i77)  denom(ratsimp(2/(x+1)-1/(x-1)));

                                         2
(%o77)                                  x  - 1
```

3.2.2 有理式の部分分数展開

通分の逆の作業に当たるのが，この部分分数分解です。これは，高校で有理関数の積分の関係で教わる式操作です。手計算で行うときは，未定係数法で行いますが，Maxima では partfrac に引き渡します。partfrac の書式は，

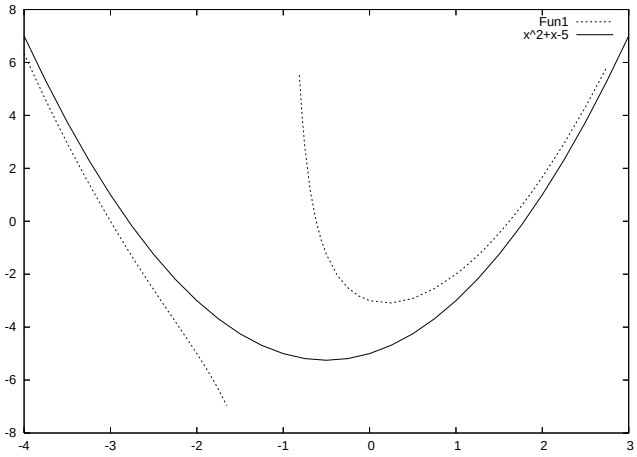
```
partfrac(有理式，変数);
```

です。

```
(%i78)  partfrac((x^3+2*x^2-4*x-3)/(x+1),x);

                                   2       2
(%o78)                           ------ + x  + x - 5
                                 x + 1
```

こうすることで，有理関数の無限大における，漸近線の多項式が分かります。つまり，部分分数分解を行ったときの，多項式部分として得ることができます。実際に，$y=\dfrac{x^3+2x^2-4x-3}{x+1}$ と $y=x^2+x-5$ の 2 つのグラフを見比べてみると分かります。

```
(%i79)  plot2d([(x^3+2*x^2-4*x-3)/(x+1),x^2+x-5],[x,-5,5],[y,-7,7]);
```




```
(%o79)
```

3.3　方程式を解く

Maxima は，方程式や連立方程式はもちろんのこと，微分方程式も代数的に計算可能なものは，ほぼすべて解いてくれます。しかも，文字変数が入ったままで何も問題なく計算を進められます。そればかりではなく，代数的に解けなかったとしても，計算パッケージを使えば，一般的な要求を十分満たす精度で数値解を出してくれるのも見逃せない機能だと思います。

方程式を解くには，`solve` を使います。`solve` の書式は，

```
solve(方程式，解かせる変数);
```

です。例えば，x の 3 次方程式 $x^3 + 2x^2 + 3x + 4 = 0$ を解かせてみます。

```
(%i80)  solve(x^3+2*x^2+3*x+4=0,x);


                    sqrt(3) %i   1
                5 (---------- - -)
                       2        2
(%o80) [x = - ----------------------
                  5 sqrt(2)   35 1/3
               9 (--------- - --)
                  3 sqrt(3)   27
                  5 sqrt(2)   35 1/3    sqrt(3) %i   1    2
              + (--------- - --)    (- ---------- - -) - -,
                  3 sqrt(3)   27            2        2    3
```

```
                                                      sqrt(3) %i   1
                                                   5 (- ---------- - -)
     5 sqrt(2)   35 1/3  sqrt(3) %i   1             2        2
x = (--------- - --)    (---------- - -) - ----------------------
     3 sqrt(3)   27          2        2       5 sqrt(2)   35 1/3
                                           9 (--------- - --)
                                              3 sqrt(3)   27
  2       5 sqrt(2)   35 1/3             5             2
- -, x = (--------- - --)    - ---------------------- - -]
  3       3 sqrt(3)   27          5 sqrt(2)   35 1/3   3
                               9 (--------- - --)
                                  3 sqrt(3)   27
```

また，厳密解が計算可能なときは，必ず厳密解を返します。ですから，厳密すぎてさっぱり分からない場合は，solve の結果を float や bfloat に引き渡します。ただ，括弧を展開しないことがあるので，その数値評価の結果をさらに expand に引き渡す方が無難です。つまり，

```
expand(float(solve(方程式, 解かせる変数)));
```

の形で求められます。

```
(%i81)  expand(float(solve(x^3+2*x^2+3*x+4=0,x)));

(%o81)   [x = - 1.546868887231395 %i - 0.17468540428031,
          x = 1.546868887231395 %i - 0.17468540428031, x = - 1.650629191439386]
```

また，係数が文字のままでも解くことができます。例えば，お馴染みの 2 次方程式を解かせてみます。

```
(%i82)  solve(a*x^2+b*x+c=0,x);
                              2                          2
                        sqrt(b  - 4 a c) + b       sqrt(b  - 4 a c) - b
(%o82)          [x = - --------------------, x = --------------------]
                              2 a                        2 a
```

このように文字が残ったままでも，処理可能なものはそのまま計算することができます。
また，連立方程式も solve で解かせることができます。書式は，

```
solve([方程式のリスト], [解かせる変数のリスト]);
```

となります。このリストは，要素をコンマで区切ります。例えば，連立 1 次方程式

$$\begin{cases} 2x + 4y = 30 \\ x + y = 12 \end{cases}$$

を解かせて[(1)]みます。

[(1)]連立 1 次方程式に限って言えば，もっと効率のよいアルゴリズムを適用する linsolve というのがあります。

```
(%i83)  solve([2*x+4*y=30,x+y=12],[x,y]);

(%o83)                                [[x = 9, y = 3]]
```

さらに不定方程式や不能方程式についても適切な情報を返してくれます。例えば，次のような連立 1 次方程式

$$\begin{cases} 2x+4y-3z=30 \\ x+y+z=12 \end{cases}$$

は，変数 3 つ，方程式 2 つですので，代表的な不足方程式でしょう。

```
(%i84)  solve([2*x+4*y-3*z=30,x+y+z=12],[x,y,z]);

                                 7 %r1 - 18       5 %r1 + 6
(%o84)              [[x = - ------------, y = ----------, z = %r1
                                    2               2
```

この場合でも，Maxima は解けるだけ解こうとします。それで解くべき変数のリストについて左から順に解き始め，不定変数 %Rn を用いて，すべての解を記述します。

また，例えば次のような連立 1 次方程式

$$\begin{cases} 2x+y=10 \\ x-y=20 \\ x+2y=30 \end{cases}$$

は，変数 2 つなのに，拡大係数行列の階数が 3 もありますので，代表的な不能方程式でしょう。

```
(%i85)  solve([2*x+y=10,x-y=20,x+2*y=30],[x,y]);


Inconsistent equations:  (3)
 -- an error.  Quitting.  To debug this try DEBUGMODE(TRUE);)
```

このように，不能方程式であると告げて処理を中断し，また入力モードに戻ります。

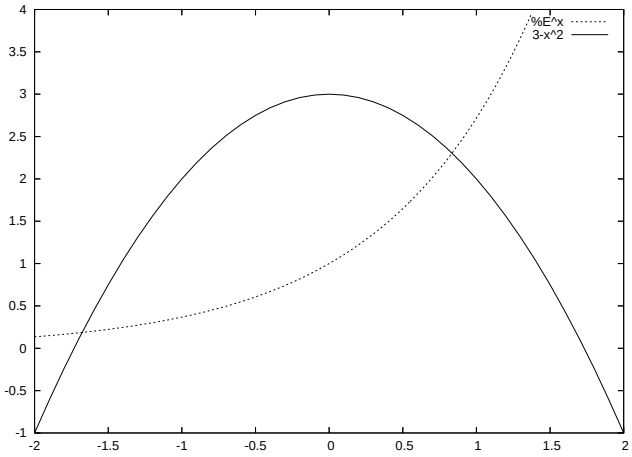
3.4　方程式の近似解

さて，この solve はあれこれ解いてくれるので便利なのですが，あらゆる方程式が解けるとは限りません。試しに次の x の方程式 $e^x = 3 - x^2$ を解かせてみてください。

```
(%i86)  solve(exp(x)=3-x^2,x);

                                          x                   x
(%o86)                 [x = - sqrt(3 - %e ), x = sqrt(3 - %e )]
```

右辺にも x が残っているので，これでは解いたことにはなりません。では，解がないのかと言えば，グラフから判断すると，あることはありそうです。

```
(%i87)  plot2d([exp(x),3-x^2],[x,-2,2],[y,-1,4]);
```




```
(%o87)
```

こういうときでも，実用上，解が必要なときにしばしば出くわします。これをニュートン法を用いて，数値的に解を近似する方法があります。 そのために必要な計算パッケージ newton.mac を読み込ませます[(2)]。パッケージを読み込むには

```
load(パッケージ名);
```

で行います。

```
(%i88)  load(newton);

(%o88)     C:/PROGRA~1/MAXIMA~1.2/share/maxima/5.9.2/share/numeric/newton.mac
```

これによって新しく newton という函数が追加されました。これは，$f(x) = 0$ を満たす x の値，つまり $f(x)$ の零点を近似的に求めることができます。 newton の書式は，

```
newton(零点を求める関数，探索開始値);
```

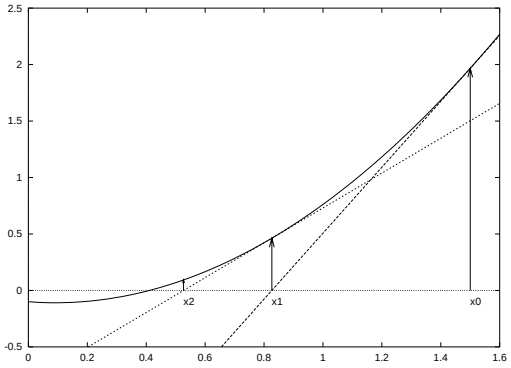
です。この問題の場合，方程式を $e^x - 3 + x^2 = 0$ と変形する必要があります。また，探索開始値は，先程のグラフの交点付近を指定します。すると，その交点に近い解を数値近似します。グラフから読み取りますと，右の交点は，0.8 くらい，左の交点は -1.6 くらいだと推測できます。

```
(%i89)  newton(exp(x)-3+x^2,0.8);

Warning:  Float to bigfloat conversion of 0.80000000000000004
```

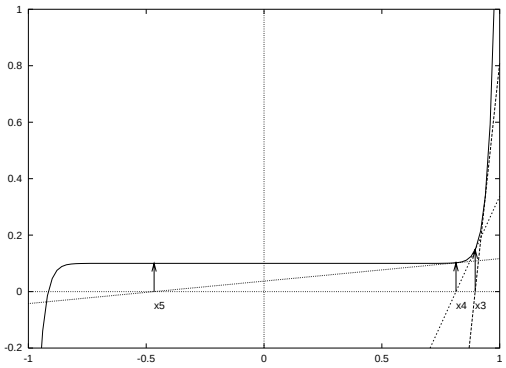
[(2)]Linux ZAURUS 版にはパッケージが準備されていませんので，Linux 版または Microsoft Windows 版のパッケージ類を maxima-clisp.mem と同じディレクトリにコピーしましょう。パッケージは，本質的には単なるテキストファイルなので，そのままコピーで大丈夫です。

```
(%o89)                          8.344868653087891B-1


(%i90)  newton(exp(x)-3+x^2,-1.6);

Warning:  Float to bigfloat conversion of -1.6000000000000001

(%o90)                          - 1.677232708533473B0
```

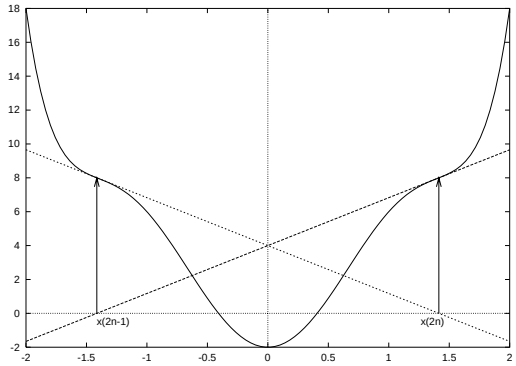
上の例のように，ニュートン法の原理から，探索開始値に最も近い実数解程度しか探索できない難点がありますが，それでもグラフである程度当たりをつければ，ほとんどの方程式を強引に解いてしまえるのが強みと言えます。このニュートン法の原理を簡単に述べますと，関数 $f(x)$ を局所的に 1 次式 (直線) で近似して解くというものです。探索開始値 x_k による点 $(x_k, f(x_k))$ で，グラフ $y = f(x)$ の接線を引きます。そして，その接線が x 軸と交わる点を x_{k+1} とします。これを新たな探索開始値として，十分な精度を得られるまで繰り返します。



ですが，ニュートン法が持つ弱点も持っています。
例えば，$x^{35} + x^{33} + \dfrac{1}{10} = 0$ を解こうとして，探索開始値に 1 を与えたとします。



$x_k = -0.46639$ の辺りで接線の傾きが計算機の限界[3]を下回ってしまい，x 軸に平行な接線と判断してしまいます。これでは，ニュートン法が続行できなくなります。



また例えば，$x^6 - 6x^4 + 13x^2 - 2 = 0$ を解こうとして，探索開始値に $\sqrt{2}$ を与えたとします。すると，$x_1 = -\sqrt{2}, x_2 = \sqrt{2}$ となり，堂々巡りをしてしまいます。

このような場合は，探索開始値を取り替えてみるといいでしょう。

3.5 章末練習問題

1. 多項式の整理

(1) 多項式 $(-a+b+c)^3(a-b+c)^3(a+b-c)^3$ を展開しなさい。

(2) 多項式 $(x+1)^{36}(x-2)^{64}$ の x^{50} の係数を求めなさい。

(3) 多項式 $-bc^4 - ac^4 + b^2c^3 + abc^3 + a^2c^3 + b^3c^2 + a^3c^2 - b^4c + ab^3c + a^3bc - a^4c - ab^4 + a^2b^3 + a^3b^2 - a^4b$ を因数分解しなさい。

(4) 193480610806663641976612700346380 81 を素因数分解しなさい。

2. 有理式の整理

(1) 有理式 $x - 1 + \dfrac{1}{3(x+1) - \dfrac{x-2}{3(x^2-x+1)}}$ を通分しなさい。

(3)計算機イプシロンと呼ばれます。

(2)　有理式 $\cfrac{1}{1+\cfrac{\boldsymbol{i}}{2+\cfrac{\boldsymbol{i}}{3+\cfrac{\boldsymbol{i}}{x}}}}$ の分子，分母をそれぞれ求めなさい。

$$\frac{1}{3+\cfrac{\boldsymbol{i}}{2+\cfrac{\boldsymbol{i}}{1+\cfrac{\boldsymbol{i}}{x}}}}$$

(3)　有理式 $\dfrac{1}{x^9+1}$ を部分分数展開しなさい。

(4)　有理式 $\dfrac{x^3-3x^2+3}{x-1}$ の無限大における漸近線を求め，それを図示しなさい。

3. 方程式の厳密解

(1)　方程式 $x^3+px+q=0$ を x について解きなさい。

(2)　連立方程式

$$\begin{cases} x^3+y^3+z^3=8 \\ x^2+y^2+z^2=27 \\ x+y+z=1 \end{cases}$$

を x, y, z について解きなさい。

(3)　連立方程式

$$\begin{cases} xy+yz+zx=1 \\ x+y+z=1 \end{cases}$$

を x, y, z について解きなさい。

(4)　連立方程式

$$\begin{cases} x^3+y^3=5 \\ x^2+y^2=3 \\ x+y=1 \end{cases}$$

を x, y, z について解きなさい。

4. 実数解の数値近似

(1)　方程式 $x=\cos(x)$ の実数解を数値近似しなさい。

(2) 方程式 $x^{35} + x^{33} + \dfrac{1}{10} = 0$ の実数解を数値近似しなさい。

(3) 方程式 $x^6 - 6x^4 + 13x^2 - 2 = 0$ の実数解を数値近似しなさい。

第4章　微分と積分

4.1　極限と微分

さて，これまでは代数的な話を中心にしてきました。ここで「連続的に変化する」ということを丁寧に突き詰めて行くと，どうしても微分や積分の話に至ります。従来の数値計算では，誤差があって当たり前，数字は不連続に変化するコンピュータにとって，連続的な変化は近似値でしか取り扱えませんでした。これを人間が行うのと同様の文字列処理をすることで，厳密に取り扱うことができるわけです。

極限は，収束性を確認した後，計算されるべきものです。しかし，単に計算するだけなら，Maxima に函数が用意されています。書式は，

```
limit(函数，変数，近づける値);
```

となります。 さらに，右極限 (左極限) を指定したいときは，オプションとして，

```
limit(函数，変数，近づける値，plus [左極限の場合は minus]);
```

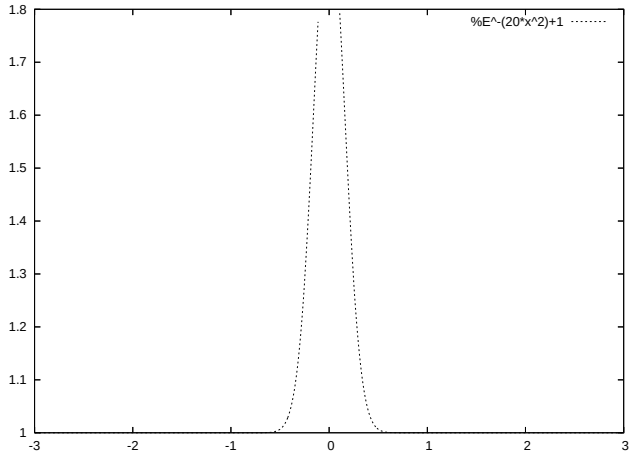
で指定します。 また近づける値が正の無限大の場合は inf, 負の無限大の場合は minf と書きます。

```
(%i91)  limit(sin(x)/x,x,0);

(%o91)                                  1


(%i92)  limit((1+1/n)^n,n,inf);

(%o92)                                 %E


(%i93)  limit(1/x,x,0,plus);

(%o93)                                INF


(%i94)  limit(1/x,x,0,minus);

(%o94)                               MINF
```

さて，グラフを描きますと，y 軸に平行な漸近線は一目で分かります。しかし，視認だけでは本当に漸近線かどうかは分かりません。

そこで逆数を取りますと，それが 0 となる点があれば，そこが y 軸に平行な漸近線となります。式から判断すると普通の連続した関数でも，係数で極端な歪みがあると，漸近線があるように見えてしまうことがあります。例えば，極端に平均に片寄っている正規分布を考えてみます。

```
(%i95)  plot2d(exp(-x^2*20)+1,[x,-3,3],[y,0,1.8]);
```




```
(%o95)
```

これだけ見ると $x = 0$ という漸近線があるように[(1)]見えます。そこで，この逆数について，x を 0 に近づけてみましょう。

```
(%i96)  limit(1/(exp(-x^2*20)+1),x,0);

                                    1
(%o96)                              --
                                    2
```

0 でなかったということは，$x = 0$ で漸近線を持たないということが分かります。

また，無限大において漸近的な挙動がある場合も，十分大きな定義域でグラフを描くと，ぼんやり見えてきます。これは，第 2 章で $x \sin\left(\dfrac{1}{x}\right)$ でも少し触れました。しかし，先程の例のように，グラフでそう見えたからといって正しいとは限りません。勿論，実用上はそれでも十分なケースが多いですが，ここではもう少し手間をかけて，それが正しいかを確認してみましょう。

$x \sin\left(\dfrac{1}{x}\right)$ の例であれば，グラフから $y = 1$ が予想されます。そして，その予想した漸近線の関数を元の式から引いて，無限大における極限値を取ります。それが 0 であれば，一般に漸近線であると言えます。

```
(%i97)  limit(x*sin(1/x)-1,x,inf);

(%o97)                              0
```

このように，直観的に把握できるグラフのメリットを生かし，最後は `limit` で確認すると，関数の漸近的な挙動がよく分かります。

(1) 勿論，わざと誤解を生むように描いています。通常の `plot2d` で描くと，普通に連続関数であることが分かります。

微分の計算も $f(x)$ を x で微分することは，近傍での微分可能性を吟味した後，$\lim_{h\to 0}\dfrac{f(x+h)-f(x)}{h}$ で求めるべきです。しかし，これも単に計算するだけでしたら，Maxima に函数が用意されています。導関数を求める函数の書式は，

```
diff(関数，微分用変数 1，階数 1，微分用変数 2，階数 2，···);
```

です。微分する変数が一つの場合は，階数は省略可能で，その場合は一階の偏導関数を求めます。また，微分用変数も与えなければ，全導関数を求めます。

```
(%i98)  diff(sin(x^2),x,2);
                                        2       2      2
(%o98)                          2 cos(x ) - 4 x  sin(x )


(%i99)  diff(sin(x*y),x,2,y,2);
                       2  2
(%o99)                x  y  sin(x y) - 2 sin(x y) - 4 x y cos(x y)


(%i100)  diff(x^2*y^2);
                                   2                 2
(%o100)                       2 x  y del(y) + 2 x y  del(x)
```

最後の `del(x)` や `del(y)` が，x や y の微小量 dx や dy を表します。

微分を利用した重要な応用例として，多項式によって近似関数を作るテイラー展開があります。勿論，後で説明します級数と，高階微分を利用して定義通り計算させることも可能ですが，ここでは一つの函数で得る方法を説明します。書式は，

```
taylor(函数, 展開変数, 展開の中心, 近似次数);
```

で計算してくれます。

```
(%i101)  taylor(tan(x),x,0,9);
                              3       5        7        9
                             x     2 x     17 x     62 x
(%o101)                 x + --- + ----- + ------ + ------ + .  .  .
                             3      15      315      2835


(%i102)  taylor(x/(1-x-x^2), x, 0, 7);
                          2      3      4      5      6       7
(%o102)             x + x  + 2 x  + 3 x  + 5 x  + 8 x  + 13 x  + .  .  .
```

4.2　級数と積分

微分が微小区間での引き算なら，積分は微小区間における関数の変化量の足し算と捉えられます。Maxima での処理も，そのようになっています。つまり，主にジョルダン積分を扱うことになりますが，それだけでも人間の手で計算する手間を考えれば，大いなる助けになる[2]ことでしょう。

まずは，積分の基本となる級数和から説明します。有限個，それも高々 10 項程度の加法や乗法であれば，すべての項を書き下したところで労力は知れています。しかし，もっと項数が多くなるとか，さらに一般の項数の場合まで扱いたければ，総和記号や総積記号に該当するものを利用したくなります。 総和を求める函数の書式が，

```
sum(関数, 添え字変数, 初期値, 終値);
```

で，総積を求める函数の書式が，

```
product(関数, 添え字変数, 初期値, 終値);
```

です[3]。

```
(%i103)  sum(i,i,1,30);

(%o103)                                465


(%i104)  product(i,i,1,30);

(%o104)                 265252859812191058636308480000000
```

[2]実際，大学の 1 年次の微分積分学の講義では，少なからぬ時間を積分の手計算方法に裂きます。それほど計算に習熟が必要なわけですが，その努力の割に，何も面白くないのもまた事実です。··· って，数学の先生が言っている場合ではないな ···。

[3]Maxima は，数学記号のアスキーアートもなかなか味わいがあります。

$\left.\frac{d}{dx} f(x)\right|_{x=0} =$

```
                          !
 d                        !
(-- (f(x))!               )
 dx                       !
                          !x = 0
```

$\sum_{k=1}^{n} x_k =$

```
 n
====
\
 >    x
/      k
====
k = 1
```

$\prod_{k=1}^{n} x_k =$

```
  n
/===\
 ! !
 ! !  x
 ! !   k
k = 1
```

$\int_a^b f(x)\,dx =$

```
 b
/
[
I  f(x) dx
]
/
 a
```

いかがですか。なお，OS によっては，\ は ¥ で表示されます。

ただ，これらは数値計算のみであり，記号的処理までは行いません。

記号的処理が必要であれば，nusum 函数によって，k が 1 から n までといった一般的な計算ができます。書式は sum 函数同様，

```
nusum(関数, 添え字変数, 初期値, 終値);
```

で計算できます。例えば，$\sum_{k=1}^{n} k^7$ や $\sum_{k=1}^{n} kr^k$ を求めてみましょう。

```
(%i105)  nusum(k^7,k,1,n);
                          2        2     4      3    2
                         n  (n + 1)  (3 n  + 6 n  - n  - 4 n + 2)
(%o105)                  ---------------------------------------
                                        24

(%i106)  nusum(k*r^k,k,1,n);
                           n + 1
                          r      (n r - n - 1)      r
(%o107)                   ------------------- + --------
                                     2                2
                           (r - 1)          (r - 1)
```

次に積分について説明します。高校までに出てくる積分は，不定積分と積分区間を指定した定積分があります。これも Maxima では積分可能なものは簡単に計算できます。不定積分を求める函数の書式が，

```
integrate(被積分関数, 積分変数);
```

で，定積分を求める函数の書式が，

```
integrate(被積分関数, 積分変数, 開始値, 終了値);
```

です。なお，不定積分のときの積分定数は，Maxima は常に 0 に設定されています。

```
(%i107)  integrate(a*x^2+b*x+c,x);
                                    3        2
                                   a x      b x
(%o107)                            ------ + ------ + c x
                                     3        2

(%i108)  integrate(exp(-x^2), x, 0, inf);
```

```
                                        sqrt(%pi)
(%o108)                                 ---------
                                            2
```

```
(%i109)  integrate(integrate(x^2,x,sqrt(y),y),y,0,1);

                                          1
(%o109)                                 - ---
                                          20
```

```
(%i110)  integrate(sin(x^2/2),x,0,2);

                 sqrt(%pi) ((%i + 1) erf(%i + 1) + (%i - 1) erf(%i - 1))
(%o110)          -------------------------------------------------------
                                          4
```

ここに出てきた erf とは，誤差関数という特殊関数であり，数学的には $\mathrm{erf}(x) = \int_{-x}^{x} \frac{e^{t^2}}{\sqrt{\pi}} dt$ で定義されるものです。このような特殊関数が混ざると，一般に float などで数値化できなくなります。

```
(%i111)  expand(float(integrate(sin(x^2/2),x,0,2)));

(%o111)     0.44311346272638 %i erf(%i + 1.0) + 0.44311346272638 erf(%i + 1.0)
          + 0.44311346272638 %i erf(%i - 1.0) - 0.44311346272638 erf(%i - 1.0)
```

そこで，計算パッケージ simpsn.mac を読み込むと，台形公式やシンプソン公式を用いて，数値積分することができます。

```
(%i112)  load(simpsn);

(%o112)     C:/PROGRA~1/MAXIMA~1.2/share/maxima/5.9.2/share/numeric/simpsn.mac
```

これによって，台形公式やシンプソン公式による数値積分が使えるようになります。 台形公式で求めるには

```
traprule(関数名, 初期値, 終了値, 等分数);
```

で行います。シンプソン公式で求めるには

```
simpson(関数名, 初期値, 終了値, 等分数);
```

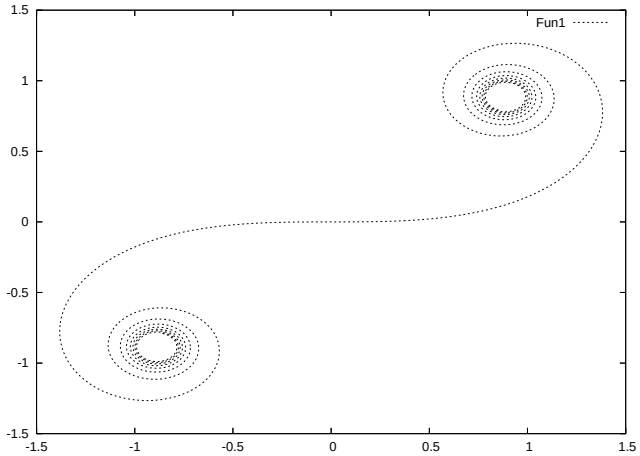
で行います。予め 1 変数被積分関数を定義しておいて，その関数名を使います。

```
(%i113)  fresnelsin(x):=sin(x^2/2);
                                                    2
                                                   x
(%o113)                          fresnelsin(x) := sin(---)
                                                   2
```

```
(%i114)  float(traprule(fresnelsin,0,2,100));

(%o114)                              0.99759596867572

(%i115)  float(simpson(fresnelsin, 0, 2, 100));

(%o115)                              0.99762370943206

(%i116)  float(simpson(fresnelsin, 0, 2, 1000));

(%o116)                              0.99762371132523
```

このように，分割数を増やすと計算時間はかかりますが，より正確な数値近似を得ることができます。この応用例として，コルヌの螺旋，別名，クロソイド[4]を描いてみます。コルヌの螺旋は，媒介変数 t を用いて $\left(\int_0^t \cos\left(\frac{s^2}{2}\right) ds, \int_0^t \sin\left(\frac{s^2}{2}\right) ds\right)$ として与えられます。

```
(%i117)  fresnelcos(x):=cos(x^2/2);

                                                    2
                                                   x
(%o117)                        fresnelcos(x) := cos(---)
                                                   2

(%i118)  plot2d([parametric,float(simpson(fresnelcos,0,t,500)),
         float(simpson(fresnelsin,0,t,500))],[t,-10,10],[nticks,1000]);
```



[4]歴史的には，コルヌによって，波動の回折の研究において初めて用いられました。現在では，高速道路のジャンクションなどのカーブなどに応用されています。

```
(%o118)
```

4.3　微分方程式

同じ方程式であっても，導関数を含むと代数操作だけでは解くことができません。つまり，solve では解けないわけです。 微分方程式を解かせるには，まず，初期値を

```
atvalue(関数, 独立変数名=値, 関数の値);
```

で与え，その後で

```
desolve(微分方程式, 求める関数);
```

で行います。例えば，バネの運動方程式 $mx''(t) = -kx(t)$, $x(0) = A$, $x'(0) = 0$ を解いてみます。

```
(%i119)  atvalue(x(t),t=0,A);

(%o119)                                    A


(%i120)  atvalue(diff(x(t),t),t=0,0);

(%o120)                                    0


(%i121)  desolve(m*diff(x(t),t,2)=-k*x(t),x(t));


Is  k m  positive, negative, or zero?
positive;

                                                 sqrt(k m) t
(%o121)                        x(t) = A cos(------------)
                                                      m
```

また例えば，連立微分方程式

$$\begin{cases} x'(t) = 2x(t) - y(t) \\ y'(t) = x(t) + 2y(t) \\ x(0) = y(0) = 1 \end{cases}$$

を解いてみます。

```
(%i122)  atvalue(x(t),t=0,1);

(%o122)                                    1
```

```
(%i123)  atvalue(y(t),t=0,1);

(%o123)                                 1


(%i124)  desolve([diff(x(t),t)=2*x(t)-y(t),diff(y(t),t)=x(t)+2*y(t)],[x(t),y(t)]);

                      2 t                               2 t
(%o124)      [x(t) = %e    (cos(t) - sin(t)), y(t) = %e    (sin(t) + cos(t))]
```

とはいえ，この desolve で解ける微分方程式も，解けない微分方程式もあります。例えば，微分方程式 $f'(x)+xf(x)=\dfrac{\sin(x)}{x}$ を解かせてみましょう。

```
(%i125)  desolve(diff(f(x),x)+x*f(x)=sin(x)/x,f(x));

(%o126) f(x) =
                d
    2 (----- (laplace(f(x), x, lvar))) - 2 atan(lvar) + 2 f(0) + %pi
       dlvar
ilt(-------------------------------------------------------------------,
                                  2 lvar
```

となり，何のことやら，さっぱり分かりません[5]。これに対して，2 階までの微分方程式であれば，既存の解法の大部分を網羅した ode2 函数を試してみましょう。書式は，

ode2(微分方程式，求める関数，独立変数);

となります。

```
(%i126)  ode2(diff(f(x),x)+x*f(x)=sin(x)/x,f(x),x);

                                    2
                                   x
                             2     --
                            x      2
                          - -- /   2
                            2   [ %e   sin(x)
(%o127)         f(x) = %e     (I ------------ dx + %c)
                                 ]      x
                                 /
```

このように，時には積分を残しながらも解けるものを解いてくれます[6]。

それでもなお，微分方程式を用いる研究や学習の，大幅な時間短縮になることは間違いありません。

---

[5] まあ，分かる人にとっては，変数 lvar(Laplace VARiable) を用いてラプラス変換 laplace した後，それを解いて，逆ラプラス変換 ilt(Inverse Laplace Transform) で戻しているだけであることは，見え見えなのですが。

[6] 線形微分方程式など代表的なものなら解けますが，ロトカ・ヴォルテラ型など非線形微分方程式は解けないものが少なくありません。

4.4　章末演習問題

1. 極限値

(1)　極限値 $\displaystyle\lim_{x\to 0}\frac{\cos(2x)-1}{\cosh(3x)-1}$ を求めなさい。

(2)　極限値 $\displaystyle\lim_{x\to 0+0}\frac{\cos(x)-1}{x\,|x|}$ を求めなさい。

(3)　極限値 $\displaystyle\lim_{n\to\infty}\left(1-\frac{2}{3n}\right)^{4n}$ を求めなさい。

(4)　関数 $y=\dfrac{x^4-x^2+x+5}{x^2-1}$ が正負の無限大において漸近線 $y=x^2$ を持つことを確認しなさい。

2. 関数の微分

(1)　関数 $f(x)=\cos\left(\cos\left(\cos(x^3)\right)\right)$ の導関数を求めなさい。また，3 階導関数を求めなさい。

(2)　関数 $f(x)$ の極大値や極小値，いわゆるグラフの山の頂きや谷の底を与える x の値は，$f'(x)=0$ を満たします。これを用いて，$f(x)=x^x,\ (x>0)$ の極小値を求めなさい。

(3)　関数 $f(x,y)=\dfrac{x^2y}{x^4+y^2}$ の x による偏導関数 $f_x(x,y)$ および，y による偏導関数 $f_y(x,y)$ を求めなさい。また，$f(x,y)$ の全導関数を求めなさい。

(4)　関数 $\log\left(\cos(\pi x)\right)$ の，$x=1$ を展開の中心とするテイラー展開を 6 次まで表示しなさい。

3. 級数和と級数積

(1)　級数和 $\log(10000)-\displaystyle\sum_{k=1}^{10000}\frac{1}{k}$ を数値近似しなさい。

(2)　関数 $\arctan(x)$ の，$x=1$ を展開の中心とする 10 次のテイラー多項式を求めなさい。

(3)　級数和 $\displaystyle\sum_{k=1}^{n}k^2e^k$ を求めなさい。

(4)　級数積 $\displaystyle\prod_{k=1}^{100}\left(1-\frac{\sin^2(k)}{k}\right)$ を数値近似しなさい。

(5)　級数積 $\displaystyle\prod_{k=1}^{10000}\left(1-\frac{\sin^2(k)}{k}\right)$ を数値近似しなさい。

4. 関数の積分

(1)　不定積分 $\displaystyle\int\frac{1}{x^5+1}\,dx$ を求めなさい。

(2) 積分方程式 $\displaystyle\int_{-\infty}^{\infty} Ce^{-\frac{x^2}{2}}\,dx = 1$ を満たす C を求めなさい。

(3) グラフ $y = f(x)$ に沿った曲線 $(a, f(a)) \sim (b, f(b))$ の長さは $\displaystyle\int_a^b \sqrt{1 + (f'(x))^2}\,dx$ で与えられます。これを用いて，関数 $y = \cosh(x)$ に沿った曲線 $(\log(2 - \sqrt{3}), 2) \sim (\log(2 + \sqrt{3}), 2)$ の長さを求めなさい。

(4) 関数 $y = f(x),\ (a \leqslant x \leqslant b, f(x) > 0)$ を x 軸を中心に回してできた，帯状の回転体の面積は，$\displaystyle 2\pi \int_a^b f(x)\sqrt{1 + (f(x))^2}\,dx$ で与えられます。
これを用いて，楕円 $\dfrac{x^2}{a^2} + \dfrac{y^2}{b^2} = 1,\ (a > b)$ を x 軸を中心に回してできた回転体の表面積を求めなさい。また，y 軸を中心に回してできた回転体の表面積を求めなさい。

5. 微分方程式

(1) 微分方程式 $v'(t) = 9.8 - 4.9v(t)$ を，初期条件 $v(0) = 0$ について解きなさい。

(2) 微分方程式 $x'''(t) - 2x''(t) + 3x'(t) - 2x(t) = \sin(x)$ を，初期条件 $x(0) = 1,\ x'(0) = 0,\ x''(0) = -1$ について解きなさい。

(3) 微分方程式 $p'(x) = -xp(x)$ を，初期条件 $p(0) = \dfrac{1}{\sqrt{2\pi}}$ について解きなさい。

(4) 連立微分方程式

$$\begin{cases} x'(t) = 4x(t) + 2y(t) + \sin(t) \\ y'(t) = 3x(t) + 3y(t) - \cos(t) \end{cases}$$

を，初期条件 $x(0) = 1,\ y(0) = -1$ について解きなさい。

第5章　リストの扱い

5.1　はじめに

このノート以外にも，足立健朗さんが『行列計算における数式処理ソフト maxima の利用について』および『maxima による行列計算超入門』がフリードキュメントとして公開されています。

```
http://www.yo.rim.or.jp/~kenrou/index-j.html
```

そちらも是非，参考にして下さい。

5.2　データのリスト

Maxima では，リストと呼ばれる，順序を込めて複数のデータを組にした集合を扱うことができます。これによって離散的なデータを統計的に処理することもできるわけですし，順序に従って演算を施せば数ベクトルのように扱うこともできます。また，そもそも数式の文字列処理ができるのも，このデータのリスト構造を分析する巧みさにあります。

リストを作るには，データの各要素をコンマで区切って，その全体を [ ] で括ります。この要素には，数値だけではなく，変数，数式，別のリスト，文字列[1]などあらゆるデータ形式を自由にとることができます。逆に，与えられたリストの要素を抽出するには，

```
リスト変数名 [要素番号];
```

で得られます。リストの要素として，リストがあるとき，**ネストしたリスト**と言います。その奥まったリストの要素を抽出するには，親リストから順に，要素番号を並べて列挙します。

```
(%i127)  lst:[[1,2,3],[4,5,6],[7,8,9]];

(%o127)                     [[1, 2, 3], [4, 5, 6], [7, 8, 9]]


(%i128)  lst[2];

(%o128)                                 [4,5,6]


(%i129)  lst[3][2];

(%o129)                                   8
```

リストを作成するときに，各要素が要素番号を変数として作成できる場合も少なくありません。そのように計算でリストを作成するときには，

[1] 文字列と変数を区別するために，文字列はダブルクォーテーションで括る必要があります。

```
makelist(計算式, カウンタ変数, 初期値, 終値);
```

で作成できます。

```
(%i130)  makelist(i^2,i,1,5);
(%o130)                              [1, 4, 9, 16, 25]

(%i131)  makelist(makelist(i*j,i,1,3),j,1,4);
(%o131)              [[1, 2, 3], [2, 4, 6], [3, 6, 9], [4, 8, 12]]
```

あるリストに要素を加えるとき，冒頭に加えるなら

```
cons(要素, リスト);
```

で，末尾に加えるなら

```
endcons(要素, リスト);
```

で行います。

```
(%i132)  cons(x,[a,b,c]);
(%o132)                                 [x, a, b, c]

(%i133)  endcons(x,[a,b,c];)
(%o133)                                 [a, b, c, x]
```

二つ以上のリストを連結させるには,

```
append(リスト 1, リスト 2,···);
```

で行います。

```
(%i134)  append([1,2,3],[4,5,6],[7,8,9]);
(%o134)                        [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

逆に，リストの冒頭を n 個除いた残りを得るには

```
rest(リスト,n);
```

を用います。そして，リストの末尾を n 個除いた残りを得るには

```
rest(リスト,-n);
```

を用います。

```
(%i135)  rest([a,b,c,d,e],2);

(%o135)                                [c, d, e]


(%i136)  rest([a,b,c,d,e],-2);

(%o136)                                [a, b, c]
```

また，リストの順番を辞書式昇順や辞書式降順に並べ替えることができます。昇順に並べるには

```
sort(リスト);
```

で，降順に並べるには

```
sort(リスト,ordergreatp);
```

または,

```
reverse(sort(リスト));
```

で並べ替えることができます。

```
(%i137)  sort([6,2,3,1,5,4,3]);

(%o137)                          [1, 2, 3, 3, 4, 5, 6]


(%i138)  sort([6,2,3,1,5,4,3],ordergreatp);

(%o138)                          [6, 5, 4, 3, 3, 2, 1]


(%i139)  reverse(sort([6,2,3,1,5,4,3]));

(%o139)                          [6, 5, 4, 3, 3, 2, 1]
```

5.3　ベクトルの演算

では，数ベクトルとしてのリスト計算の方法を説明します。例えば，ある 4 次元ベクトルを変数 $\boldsymbol{v}_1$ と $\boldsymbol{v}_2$ に代入してみます。Maxima の変数名表記法の問題で，それぞれ v1, v2 と記述します。

```
(%i140)  v1:[1,0,2,-1];

(%o140)                              [1, 0, 2, - 1]


(%i141)  v2:[2,1,-2,0];

(%o141)                              [2, 1, - 2, 0]
```

これをベクトルとして扱うとき，加減やスカラー倍は，普通に数学の表記に従います。

```
(%i142)  v1+v2;

(%o142)                              [3, 1, 0, - 1]


(%i143)  3*v1;

(%o143)                              [3, 0, 6, - 3]


(%i144)  -2*v1+3*v2

(%o144)                             [4, 3, - 10, 2]
```

また，複数の数ベクトルを行ベクトルとして，行列属性を与えることができます。例えば，ある 2 行 2 列の行列を，$\boldsymbol{M}_1$ と $\boldsymbol{M}_2$ に代入してみます。これも Maxima の変数名表記法の問題で，それぞれ M1, M2 と記述します。

```
(%i145)  M1:matrix([1,-2],[2,1]);
                                   [ 1  - 2 ]
(%o145)                            [        ]
                                   [ 2   1  ]


(%i146)  M2:matrix([2,-1],[1,2]);
                                   [ 2  - 1 ]
(%o146)                            [        ]
                                   [ 1   2  ]
```

このように行列であっても，ベクトルとしての加法およびスカラー倍は同様にできます。

```
(%i147)  -2*M1+3*M2;
                                    [  4   1 ]
(%o147)                             [        ]
                                    [ - 1  4 ]
```

さて，このように定義されたベクトルの特定の成分を指定するには，リストとして第何番目の要素かを指定します。例えば，$\boldsymbol{v}_1$ の第 2 成分を指定するには，`v1[2]` と記述します。また，行列の特定の成分の指定も同様で，例えば $\boldsymbol{M}_1$ の第 $(1,2)$ 成分を指定するには，`M1[1,2]` と記述します。

```
(%i148)  v1[2];
(%o148)                                  0

(%i149)  M1[1,2];
(%o149)                                - 2
```

次に 2 つのベクトルの積について説明します。実ベクトル空間の場合の内積 $v_1 \cdot v_2$ は，第 n 成分同士の積の総和を取ったもので，これを得るには `.` で積を取ります。

```
(%i150)  v1.v2;
(%o150)                                - 2
```

また，ベクトルの大きさ $||\boldsymbol{v}||$ は，自分自身との内積の平方根 $\sqrt{\boldsymbol{v}\cdot\boldsymbol{v}}$ として与えられます。

```
(%i151)  sqrt(v1.v1)
(%o151)                               sqrt(6)
```

そして 3 次元数ベクトル空間の場合には，外積が定義できます。外積を求める函数は Maxima にデフォールトではないので，少し工夫してやると良いでしょう。例えば，次のようにすれば得られます。

```
(%i152)  v1:[a,b,c];
(%o152)                              [a, b, c]

(%i153)  v2:[x,y,z];
(%o153)                              [x, y, z]

(%i154)  transpose(adjoint(matrix(v1,v2,[1,1,1])))[3];
(%o154)                    [q z - r y, r x - p z, p y - q x]
```

`transpose`, `adjoint` については，後で説明します。

5.4　行列の演算

先も述べましたように，行列の場合でも，加減やスカラー倍はベクトルと同様に行います。 ここでは行列独特の計算を紹介します。まず行列同士の積ですが，これは左の行列の行ベクトルと右の行列の列ベクトルとの内積をとったものが各成分になるわけです。つまり，ベクトルの場合と同様 . で積をとることができます。

```
(%i155)  M1:matrix([1,-2,3],[2,1,0]);

                              [ 1  - 2  3 ]
(%o155)                       [           ]
                              [ 2   1   0 ]


(%i156)  M2:matrix([2,-1],[1,2],[-1,1]);

                               [  2   - 1 ]
                               [          ]
(%o156)                        [  1    2  ]
                               [          ]
                               [ - 1   1  ]


(%i157)  M1.M2;

                               [ - 3  - 2 ]
(%o157)                        [          ]
                               [  5    0  ]


(%i158)  M2.M1;

                             [ 0  - 5   6  ]
                             [             ]
(%o158)                      [ 5   0    3  ]
                             [             ]
                             [ 1   3   - 3 ]
```

また，行列の累乗は，^^ で指数のように記述できます。

```
(%i159)  M1:matrix([1,1],[1,0]);


(%i160)  M1^^100;

           [ 573147844013817084101  354224848179261915075 ]
(%o160)    [                                             ]
           [ 354224848179261915075  218922995834555169026 ]
```

ベクトルに行列を作用させるときも，n 次ベクトルを n 行 1 列行列と見れば，これも行列の積で書けば良いことが分かります。しかも便利なことに，ベクトルは . の前にあるものは自動的に行ベクトル，後にあるものは列ベクトルであると判断してくれます。

```
(%i161)  mtx:matrix([1,-2],[2,1]);
                                  [ 1  - 2 ]
(%o161)                           [        ]
                                  [ 2   1  ]


(%i162)  vec:[3,-2];

(%o162)                                [3, - 2]


(%i163)  mtx.vec;
                                       [ 7 ]
(%o163)                                [   ]
                                       [ 4 ]


(%i164)  vec.mtx;

(%o164)                              [ - 1  - 8 ]
```

行列の行と列を交換したものを転置行列といい，

```
transpose(行列);
```

で得ることができます。

```
(%i165)  mtx:matrix([2,-1,0,0],[-1,2,-1,0],[0,-1,2,-1]);
                        [  2   - 1   0    0  ]
                        [                    ]
(%o165)                 [ - 1   2   - 1   0  ]
                        [                    ]
                        [  0   - 1   2   - 1 ]


(%i166)  transpose(mtx);
```

```
                              [  2   - 1   0  ]
                              [               ]
                              [ - 1   2   - 1 ]
(%o166)                       [               ]
                              [  0   - 1   2  ]
                              [               ]
                              [  0    0   - 1 ]
```

この transpose は，少し工夫するとデータ操作などで結構使える函数であります。例えば，行列の中から第 3 行ベクトルを抜き出す[2]には，次のようにします。

```
(%i167)  freefall:matrix([1.0,9.8,4.9],[2.0,19.6,19.6],[3.0,29.4,44.1],
         [4.0,39.2,78.4],[5.0,49.0,122.5]);
                              [ 1.0   9.8    4.9  ]
                              [                   ]
                              [ 2.0  19.6   19.6  ]
                              [                   ]
(%o167)                       [ 3.0  29.4   44.1  ]
                              [                   ]
                              [ 4.0  39.2   78.4  ]
                              [                   ]
                              [ 5.0  49.0  122.5  ]


(%i168)  freefall[3];

(%o168)                              [3., 29.4, 44.1]
```

これに対して，第 3 列ベクトルを抜き出すには，transpose で転置しておいてから，第 3 行ベクトルを抜き出せばよい[3]わけです。

```
(%i169)  transpose(freefall)[3];

(%o169)                        [4.9, 19.6, 44.1, 78.4, 122.5]
```

以後，行列はすべて正方行列，つまり行数と列数が等しい行列に限定して説明します。まず，単位行列や対角行列，そして零行列といった特定の行列は，すべての成分を記述する必要がありません。これらには専用の函数や上手い作り方があります。単位行列は，

```
ident(次数);
```

で作成できます。対角行列は，

[2] row(行列, 行番号); というものもあります。
[3] col(行列, 列番号); というものもあります。

```
apply(matrix,makelist(ident(次数)[i]*対角成分リスト [i],i,1, 次数));
```

で作成できます。零行列は,

```
zeromatrix(行数, 列数);
```

で作成できます。具体的に 4 次正方行列で作ってみましょう。

```
(%i170)  ident(4);
                                [ 1  0  0  0 ]
                                [            ]
                                [ 0  1  0  0 ]
(%o170)                         [            ]
                                [ 0  0  1  0 ]
                                [            ]
                                [ 0  0  0  1 ]


(%i171)  apply(matrix,makelist(ident(4)[i]*[2,3,5,7][i],i,1,4));
                                [ 2  0  0  0 ]
                                [            ]
                                [ 0  3  0  0 ]
                                [            ]
(%o171)                         [ 0  0  5  0 ]
                                [            ]
                                [ 0  0  0  7 ]


(%i172)  zeromatrix(4,4);
                                [ 0  0  0  0 ]
                                [            ]
                                [ 0  0  0  0 ]
(%o172)                         [            ]
                                [ 0  0  0  0 ]
                                [            ]
                                [ 0  0  0  0 ]
```

また，正則行列であれば逆行列を持つわけですが，正則かどうかを判断するには，その行列の行列式が 0 か否かを調べるとよいわけです。その行列式を求めるのも

```
determinant(行列);
```

と函数化されています。

```
(%i173)  determinant(matrix([1,-2],[2,1]));

(%o173)                                5
```

これが 0 でなければ，その行列 M の余因子行列 $\tilde{M}$ を行列式で割ることで，M の逆行列 M^{-1} を得ることができます。余因子行列を求めるには

```
adjoint(行列);
```

で求めます。

```
(%i174)  adjoint(matrix([1,-2],[2,1]));
                                 [  1   2 ]
(%o174)                          [        ]
                                 [ - 2  1 ]


(%i175)  adjoint(matrix([1,-2],[2,1]))/determinant(matrix([1,-2],[2,1]));
                                 [  1   2 ]
                                 [  -   - ]
                                 [  5   5 ]
(%o175)                          [        ]
                                 [   2  1 ]
                                 [ - -  - ]
                                 [   5  5 ]
```

勿論，こんな面倒なことをしなくても

```
invert(行列)
```

として函数化されています。

```
(%i176)  invert(matrix([1,-2],[2,1]));
                                 [  1   2 ]
                                 [  -   - ]
                                 [  5   5 ]
(%o176)                          [        ]
                                 [   2  1 ]
                                 [ - -  - ]
                                 [   5  5 ]
```

また，行列の跡 (トレース) も

```
mattrace(行列);
```

と函数化されていますが，これはパッケージ nchrpl.mac に含まれていますので，先にこれを読み込む必要があります。

```
(%i177)  load(nchrpl);

(%o177)     C:/PROGRA~1/MAXIMA~1.2/share/maxima/5.9.2/share/matrix/nchrpl.mac


(%i178)  mtx:matrix([1,0,-1],[0,2,1],[-1,0,0]);
                                [  1   0  - 1 ]
                                [             ]
(%o178)                         [  0   2   1  ]
                                [             ]
                                [ - 1  0   0  ]


(%i179)  mattrace(mtx);

(%o179)                                  3
```

行列式が出せるのならば，定義に基づいて固有多項式を得ることができます。さらに solve と組み合わせると固有値や固有ベクトルが計算できます。ですが，固有多項式であれば nchrpl.mac を読み込めば，

```
ncharpoly(行列, 変数名);
```

で得られます。 また，eigen.mac を読み込めば，固有値は

```
eigenvalues(行列);
```

で，それぞれの固有値に対応する固有ベクトルは

```
eigenvectors(行列);
```

で求めることができます。

```
(%i180)  load(eigen);

(%o180)     C:/PROGRA~1/MAXIMA~1.2/share/maxima/5.9.2/share/matrix/eigen.mac


(%i181)  ncharpoly(mtx,t);
```

```
(%o181)                              3      2
                                     t  - 3 t  + t + 2

(%i182)  eigenvalues(mtx);
                              sqrt(5) - 1   sqrt(5) + 1
(%o182)                [[- ------------, ------------, 2], [1, 1, 1]]
                                    2             2

(%i183)  eigenvectors(mtx);
                    sqrt(5) - 1   sqrt(5) + 1                          sqrt(5) - 1
(%o183)       [[[- ------------, ------------, 2], [1, 1, 1]],[1, - ------------,
                         2             2                                  2
                   sqrt(5) + 1        sqrt(5) + 1     sqrt(5) - 1
                  ------------], [1, ------------, - ------------], [0, 1, 0]]
                       2                  2               2
```

ただ，5 次以上の方定式は一般には解けないので，eigenvalues では厳密解が求まらない場合があります。

```
(%i184)  mtx:matrix([2,-2,5,8,-3],[1,4,3,2,7],[5,7,-4,0,4],[6,3,6,9,2],[5,8,9,1,5]);
                         [ 2  - 2   5   8  - 3 ]
                         [                     ]
                         [ 1   4    3   2    7 ]
                         [                     ]
(%o184)                  [ 5   7   - 4  0   4  ]
                         [                     ]
                         [ 6   3    6   9   2  ]
                         [                     ]
                         [ 5   8    9   1    5 ]


(%i185)  eigenvalues(mtx);

SOLVE is unable to find the roots of
the characteristic polynomial.

(%o185)                                  []
```

このような時は，定義に戻りましょう。つまり，

(1) まだ読み込んでいなければ，load(newton); と load(nchrpl); で必要なパッケージを読み込みます。
(2) n:length(mtx); で正方行列の大きさを求めます。
(3) c:ncharpoly(mtx,x); で固有多項式を求めます。
(4) t:sqrt(sum(sum(mtx[i,j]^2,i,1,n),j,1,n)); で固有値の範囲を求めます。
(5) plot2d([0,c],[x,-t,t]); でグラフを描き，固有値の近似値 s をグラフから読み取ります。
(6) a:newton(c,s); で数値近似解を求めます。

これで，全ての固有値が実数であれば，比較的小さい誤差で求めることができます。また，複素数の固有値を持つ場合は，ベアストウ法とか，ヒッチコック法と呼ばれる数値近似法があります。ここでは詳しく紹介しませんので，専門書などを参考にして下さい。

5.5 章末演習問題

データのリスト

(1) リスト $[1-t, 1+t, 1-t^2, 1+t^2]$ を変数 p に定義しなさい。また，その第 3 項を抽出しなさい。

(2) リスト p から第 3 項を削除しなさい。さらに，p の末尾に $1+t^4$ という項を追加しなさい。

(3) リスト $[1, 1+t, 1+t+t^2, 1+t+t^2+t^3]$ を変数 q に定義しなさい。そして，リスト p とリスト q を連結しなさい。

(4) `makelist` を用いて，掛け算の九々の表を作りなさい。そして，それを行列に変換して，正方形状に表示しなさい。

(5) 500 個の 0 から 10000 までの乱数を要素に持つリストを作りなさい。そして，それを降順に並べ直しなさい。なお，0 から 10000 までの乱数は，Maxima では `random(10000)` で与えられます。

ベクトルの演算

(1) ベクトル $\boldsymbol{v}_1 = (2.62, -0.12, 1.21)$, $\boldsymbol{v}_2 = (2.66, -0.44, 2.12)$ について，$1.27\boldsymbol{v}_1 - 0.92\boldsymbol{v}_2$ を求めなさい。

(2) ベクトル $\boldsymbol{v}_1 = (9+\sqrt{5}, 9-\sqrt{5}, 3\sqrt{2})$, $\boldsymbol{v}_2 = (5-\sqrt{5}, 5+\sqrt{5}, 2)$ について，ベクトル $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ の長さを求めなさい。また，それぞれの単位ベクトル (長さが 1 のベクトル) を求めなさい。

(3) 内積は 2 つの n 次ベクトル $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ が n 次元空間の中で為す角度 θ を用いて，$\boldsymbol{v}_1 \cdot \boldsymbol{v}_2 = ||\boldsymbol{v}_1||\,||\boldsymbol{v}_2|| \cos(\theta)$ としても定義されます。このことを利用して，2 つの 3 次ベクトル $\boldsymbol{v}_1 = (-1, 0, 2)$, $\boldsymbol{v}_2 = (0, -1, 3)$ が 3 次元空間の中で為す角度を求めなさい。

(4) 三角形 OAB の面積 S は，ベクトル $\overrightarrow{OA}, \overrightarrow{OB}$ を用いて，$S = \dfrac{1}{2}|\overrightarrow{OA} \times \overrightarrow{OB}|$ として，外積の大きさの半分で求まります。このことを利用して，3 点 $O = (0, 0, 0)$, $A = (1.12, 0.37, -0.88)$, $B = (1.71, 0.64, 0.95)$ が成す三角形の面積を求めなさい。

行列の演算

(1) 行列

$$A = \begin{pmatrix} t-1 & 1 \\ 0 & -t \end{pmatrix}, B = \begin{pmatrix} t & 0 \\ 1-t & -1 \end{pmatrix}$$

について，$2A + 3B$, $AB - BA$, $A^5 + B^5$ を求めなさい。

(2) 行列

$$\begin{pmatrix} -2.87 & 0.36 & -2.78 & -2.90 & -0.22 \\ -1.95 & -1.69 & 0.56 & -2.87 & -1.67 \\ -0.47 & 0.21 & -2.71 & 0.60 & 1.99 \\ 1.01 & 1.52 & -1.79 & 1.54 & 1.35 \\ -1.06 & -1.92 & -2.86 & -1.15 & 0.24 \end{pmatrix}$$

の行列式と逆行列を求めなさい。

(3)　行列

$$\begin{pmatrix} 1.51 & 1.48 & -0.81 \\ -2.45 & -1.17 & -2.21 \\ -2.30 & -2.12 & 1.74 \end{pmatrix}$$

の固有値と固有ベクトルを求めなさい。

(4)　線形漸化式 $x_n = 5x_{n-1} - 6x_{n-2} + 2$, $x_1 = 1$, $x_2 = 2$ を，行列を用いて解きなさい。

(5)　行列

$$\begin{pmatrix} 2 & -2 & 5 & 8 & -3 \\ 1 & 4 & 3 & 2 & 7 \\ 5 & 7 & -4 & 0 & 4 \\ 6 & 3 & 6 & 9 & 2 \\ 5 & 8 & 9 & 1 & 5 \end{pmatrix}$$

の固有値を数値近似しなさい。

第6章　プログラミングの初歩

これまでは，様々な函数を駆使して必要な数式処理をする方法を説明してきました。この章では，さらに一歩進んで，外部ファイルを読み込んで処理したり，必要な函数を作成する方法を説明します。これができますと，かなり本格的に Maxima を駆使することができるようになります。しかし，その代償として，細かなミスを発見しては修正するという**デバグ作業**が不可欠になります。このデバグ作業は，Maxima に限らず，どのようなプログラミング言語を用いたとしても，かなり経験が必要です。ここでは，わざと「問題ある函数」を作っては，それを修正していく作業も紹介します。

6.1　函数の作成

Maxima で，ユーザに必要な函数を新しく作ることは，一般的にプログラミングのセンスが要求されます。基本的な手順は次の通りです。

(1)　新しいユーザ函数は，何を入力して，結果として何を出力するかを明確にします。
(2)　その結果を得るために，手作業ではどのような手順を踏むか，細かく丁寧に考えます。
(3)　ある処理の結果をそのまま次の手順の入力データとするならば，ネストさせます。
(4)　複数の処理の結果を引き数とするならば，各処理結果を変数に保管しておきます。
(5)　一通り完成したら，いくつかの値を代入してテストします。特に,「いじわる」な値を選ぶといいでしょう。

例えば複素数を与えたら，その分母を実数値化する関数を作ってみます。 手計算で行うように，分母の共役複素数を分母子に掛けて求めてもいいのですが，Maxima には複素数の実部や虚部だけを取り出す函数

```
realpart(複素数);
```

と

```
imagpart(複素数);
```

がありますので，それを使うのが楽でしょう。

```
(%i186)  mycomplex(z):=ratsimp(realpart(z)+imagpart(z)*%i);

(%o186)           mycomplex(z) := ratsimp(realpart(z) + imagpart(z) %i)


(%i187)  mycomplex(1/(3+2*%i));
                                          2 %i - 3
(%o187)                                 - ---------
                                            13
```

このように，データ z が入力されたら，分母を実数化することができる関数を作成することができます。

6.2　分岐構造

ある変数の値によって処理の仕方を変えるようにプログラムすることを，分岐構造をもたせるといいます。 そのためには，if を用います。書式は，

```
if 条件式 then 真の場合の処理 else 偽の場合の処理;
```

となります。「真」とは条件式が成立する場合,「偽」とは不成立の場合を意味する言葉です。

条件式は，ある変数の属性を問うものや，二項間の関係を問うものが使われます。　前者であれば，例えば，整数かを問う integerp, 偶数かを問う evenp, 奇数かを問う oddp, 素数かを問う primep, 有理数かを問う ratnump, リストかを問う listp, 行列かを問う matrixp などがあります。後者であれば，等しいかを問う =, 等しくないかを問う #, 実数として大小関係を問う <, >, >=, <= があります。

その条件式が成立する時は真の場合の処理を行い，成立しない時は偽の場合の処理を行います。

複数の条件を同時に満たすとか，どちらかを満たすかといった条件式を記述することもできます。すべて同時に満たすかを問うには，

```
条件式 1 and 条件式 2 and ...
```

を用います。また，どれかを満たすかを問うには，

```
条件式 1 or 条件式 2 or ...
```

を用います。例えば，ある数 p と $p+2$ がともに素数であるとき，その組を双子素数といいます。それを判定する函数 mytwinprime を[(1)]作ってみます。

```
(%i188)  mytwinprime(num):=if primep(num) and primep(num+2) then "twin prime";

(%o188)     mytwinprime(num) := if primep(num) and primep(num + 2) then "twin
                                                                  prime"


(%i189)  mytwinprime(11777);

(%o189)                                      twin prime


(%i190)  mytwinprime(11779);

(%o190)                                       false
```

このように，else を省略すると，条件不成立のときは false とだけ返ってきます。

”5.3 行列の演算” で，対角行列を入力する方法を説明しましたが，ここでは引き数に対角成分のリストを引き渡すとそれを対角行列にする函数 mydiagonal を作ってみます。

[(1)]ただし，素数の判定は難しい問題を含んでおり，primep が正しいと仮定しての話ですが。

```
(%i191)  mydiagonal(lst):=apply(matrix,
                makelist(
                    makelist(
                        if i=j then lst[i] else 0,
                    i,1,length(lst)),
                j,1,length(lst))
             );
```

```
(%o191)  mydiagonal(lst) := apply(matrix,
         makelist(makelist(if i = j then lst  else 0, i, 1, length(lst)), j,
                                             i
         1, length(lst)))
```

```
(%i192)  mydiagonal([1,2,-1,-2]);
```

```
                          [ 1  0   0    0  ]
                          [                ]
                          [ 0  2   0    0  ]
(%o192)                   [                ]
                          [ 0  0  - 1   0  ]
                          [                ]
                          [ 0  0   0   - 2 ]
```

この例から分かりますように，if, then, else で纏めて一つの函数というわけです。

また，条件によって 3 つ以上の場合に分けるときには，else の後にまた if による分岐を与えるようにします。例えば，身長を cm で，体重を kg で引き渡したとき，痩身か正常か肥満かを表示する函数を作ってみます。その指標は，身長を h (cm), 体重を w (kg) として，$\frac{10000w}{h^2}$ で与えられ，指標が 18.5 未満であれば痩身，18.5 以上 25 未満であれば正常，25 以上であれば肥満とされています。

```
(%i193)  mybmi(h,w):=if 10000*w/h^2<18.5 then "thin" else
                (if 10000*w/h^2<25 then "normal" else "fat");
```

```
                            10000 w
(%o193)  mybmi(h, w) := if -------- < 18.5 then "thin"
                              2
                             h
                          10000 w
                 else (if -------- < 25 THEN "normal" else "fat")
                            2
                           h
```

```
(%i194)  mybmi(167.4,58.2);
```

```
(%o194)                                normal
```

では，デバグの例を紹介します。例えば，Maxima には Microsoft Excel の INT や FLOOR に当たる函数がありません。小数部分を切り捨てるものとして ?truncate という函数[2]はありますが，これは正ならば切り捨て，負ならば切り上げになるという，つまりより 0 に近い整数値にするものです[3]。これと if を使って，与えられた x を超えない最大の整数を得る函数 myint を作ってみます。

```
(%i195)  myint(x):=if x>=0 then ?truncate(x) else ?truncate(x-1);

(%o195)        myint(x) := if x >= 0 then truncate(x) else truncate(x - 1)


(%i196)  myint(3.14);

(%o196)                                      3


(%i197)  myint(-3.14);

(%o197)                                    - 4
```

何となく上手くいっているようですが，このままではいくつも重大な問題点があります。まず論理的な問題として，負の整数値を入れたときに，一つ小さな値を返してしまうことです。

```
(%i198)  myint(-3);

(%o198)                                    - 4
```

これを回避するには，x が負の場合，つまり if を入れ子にして更なる整数か否かで場合分けをする必要があります。

```
(%i199)  myint(x):=if x>=0 then ?truncate(x) else
                 (if integerp(x) then x else ?truncate(x-1));

(%o199)                   myint(x) := if x >= 0 then truncate(x)
                        else (if integerp(x) then x else truncate(x - 1))


(%i200)  myint(-3)

(%o200)                                    - 3
```

これで負の整数値の問題が解決されました。しかし，問題はまだ残ります。?truncate は，小数部分を切り捨てるものですから，22/7 のような有理数表現されたものや %pi のようなシンボルにはエラーを返します。

```
(%i201)  myint(22/7);
```

[2]頭の ? は，truncate が Maxima の函数ではなく，Maxima のベースとなっている LISP の函数であることを意味します。
[3]正確には，?truncate(被除数, 除数); として使い，その整数商を求める函数です。この除数が省略されると自動的に 1 が採用されるので，正ならば切り捨て，負ならば切り上げになります。

```
Maxima encountered a Lisp error:

 Error in MACSYMA-TOP-LEVEL [or a callee]: ((RAT SIMP) 22 7) is not of type (OR
                                                                              RATIONAL
                                                                              LISP:FLOAT).

Automatically continuing.
To reenable the Lisp debugger set *debugger-hook* to nil.


     (%i202)  myint(%pi);


Maxima encountered a Lisp error:

 Error in MACSYMA-TOP-LEVEL [or a callee]: $%pi is not of type (OR
                                                               RATIONAL
                                                               LISP:FLOAT).

Automatically continuing.
To reenable the Lisp debugger set *debugger-hook* to nil.
```

これを回避するには，x を float に引き渡しておけば良いということになります。

```
     (%i203)  myint(x):=if x>=0 then ?truncate(float(x)) else
                      (if integerp(x) then x else ?truncate(float(x-1)));

     (%o203)   myint(x) := if x >= 0 then truncate(float(x))
                    else (if integerp(x) then x else truncate(float(x - 1)))


     (%i204)  myint(22/7);

     (%o204)                                    3


     (%i205)  myint(%pi);

     (%o205)                                    3
```

これでデバグ完了です。

6.3 反復構造

人間にとって，単調な作業の繰り返しほど退屈なものはありません。コンピュータはプログラムを組んで，通電さえしておけば，人間のように数時間で疲労することもなく忠実に作業を繰り返します。

まずはカウンタを用いて，繰り返し回数を指定するためには，for, thru, do を用います。書式は，

```
for カウンタ名:初期値 step 増分 thru 終了値 do(反復実行手続き);
```

となります。この step は省略することができ，その場合は i:i+1 が指定されたものとなります。そして，do の括弧の中身は，複数の作業を指定するときは，順番にコンマで区切って列挙します。例えば，与えられた行列の 100 乗を，^^ を用いずに求めてみましょう。

```
(%i206)  mtxpow:ident(2);
                                    [ 1  0 ]
(%o206)                             [      ]
                                    [ 0  1 ]


(%i207)  mtx:matrix([1,1],[1,0]);
                                    [ 1  1 ]
(%o207)                             [      ]
                                    [ 1  0 ]


(%i208)  for i:1 thru 100 do(mtxpow:mtxpow.mtx);

(%o208)                                done


(%i209)  mtxpow;
           [ 573147844013817084101  354224848179261915075 ]
(%o209)    [                                             ]
           [ 354224848179261915075  218922995834555169026 ]
```

また，カウンタに関係なく，ある条件を満たし続ける限り反復させるためには，for, while, do を用います。書式は，

```
for カウンタ名:初期値 step 増分 while 条件式 do(反復実行手続き);
```

となります。この step も省略することができ，その場合は i:i+1 が指定されたものとなります。例えば，件数が分かっていないデータを，次々と数列 d_i に入力するとしましょう。ただし，データの入力終わりは，-1 で終わることにします。

```
(%i210)  for i:1 while d[i-1]#-1 do(d[i]:read("No.",i,":"));
```

```
No. 1
2.231;
No. 2
0.965;
No. 3
-1.478;
No. 4
-1;
```

```
(%o210)                                    done
```

```
(%i211)  d[2];
```

```
(%o211)                                    0.965
```

6.4 ブロック化

既存の函数に，分岐構造や反復構造を加えると，様々な処理ができます。それらの手順をまとめて一つの函数にする方法が，このブロック化です。ブロック化するときは，C などプログラミング言語でプログラムするときのような構造で記述されます。 書式は，

```
block([局所変数のリスト], 一連の手続き,return(計算結果));
```

となります。

例えば，先ほどのデータ入力作業に加えて，その算術平均と標準偏差を求める mystatistics という函数に仕立ててみましょう。もっと効率よく計算する方法もありますが，ここでは単純に定義通り，算術平均 μ を

$$\mu = \frac{\sum_{i=1}^{n} d_i}{n}$$

で，標本標準偏差 σ を

$$\sigma = \frac{\sum_{i=1}^{n} (d_i - \mu)^2}{n}$$

で，不偏標準偏差 s を

$$s = \frac{\sum_{i=1}^{n} (d_i - \mu)^2}{n-1}$$

で，それぞれ計算することにします。また，引き数に 0 を与えると標本標準偏差を，1 を与えると不偏標準偏差を返すことにします。

```
(%i212)  mystatistics(flag):=block([d,n,m:0,v:0],
             for i:1 while d[i-1]#-1 do(d[i]:read("No.",i,":"),n:i-2),
             if n<2 then return("false"),
             if (flag#0 and flag#1) then return("false"),
             for j:1 thru n do(m:m+d[j]/n),
             for j:1 thru n do(v:v+(d[j]-m)^2/(n-flag)),
             print("average :  ",float(m),", stdev :  ",float(sqrt(v))),
             return("done"));
```

```
(%o213) mystatistics(flag) := block([d, n, m : 0, v : 0],
        for i while d      # - 1 do (d  : read("No.", i, ":"), n : i - 2),
                    i - 1           i
        if n < 2 then return("false"), if flag # 0 and flag # 1
                                                          d
                                                           j
         then return("false"), for j thru n do m : m + --,
                                                          n
                                       2
                                (d  - m)
                                  j
        for j thru n do v : v + ---------,
                                n - flag
        print("average : ", float(m), ", stdev : ", float(sqrt(v))),
        return("done"))


(%i213)  mystatistics(0);


No. 1
2.231;
No. 2
0.965;
No. 3
-1.478;
No. 4
-1;
average :  1.598 , stdev :  0.633

(%o213)                                    done
```

ここで出てきました局所変数について説明しておきます。局所変数といいますのは，その函数内部だけで有効な臨時の変数であり，それと一般の変数名が重なっても互いの内容に影響を与えません。例えば，先程の mystatistics 函数は，局所変数として d, n, m, v を使っていますが，予めそれらの変数に何を与えていても，mystatistics の計算で変化しません。

```
(%i214)  v:sqrt(-1);

(%o214)                                     %i


(%i215)  mystatistics(0);


No. 1
2.231;
No. 2
```

```
0.965;
No. 3
-1.478;
No. 4
-1;
average :  1.598 , stdev :  0.633
```

```
(%o215)                                    done
```

```
(%i216)  v;
```

```
(%o216)                                     %i
```

言わば，局所変数の v には「mystatistics の」という修飾語が付いていると思ってもらっていいでしょう。

6.5 外部ファイルの利用

Maxima で様々な計算をするのはいいのですが，それらを記録するときに手書きする時代でもありません。かといって，一般的なアプリケーションのようなファイル保存らしきもメニューにはありません。

6.5.1 手順や結果の記録

まず，計算手順やせっかく作った函数などを外部ファイルに記録するには，

```
save("ファイル名",all);
```

で行います。このファイルは，フルパスを省略すると，Maxima と同じディレクトリに保存されます。また，特定の計算セルだけを保存したければ，[4]

```
save("ファイル名",%inn,%onn,...);
```

で保存します。このときの %i で始まるものが入力データ，%o で始まるものが出力データになります。また，書式中の nn は，セルの番号を表します。
保存されたデータを，別のセッションで読み込むには，

```
loadfile("ファイル名");
```

で読み込みます。しかし，読み込んだだけでは，何を読み込んだか分かりません。そこで，何を実行したことになっているのかを確認するために，

```
playback(all);
```

[4] Maxima 5.9.0 以前のバージョンであれば，入力セルが Cnn，出力セルが Dnn となっていますので，それで読み替えてください。

と入力すると良いでしょう。部分的に見たいだけでしたら，

```
playback([開始セル番号, 終了セル番号]);
```

で表示を抑えることができます。

playback 函数で見ると分かりますように，この loadfile で記録を読み込みますと，現在のセッションとセル番号が重なると上書きされてしまいます。ですから，それまでの結果を上書きされないようにするためにも，(%i1) のセルで読み込むことをお勧めします。

6.5.2　データファイルの利用

save 函数で保存される形式は，テキストファイルではありますが Maxima 独自のもの[5]です。ですから，Maxima が生成する計算データを他のアプリケーション，例えば Calc や Microsoft Excel といったプログラムで利用するには，csv 形式にしないといけません。csv 形式とは，Comma Separeted Values の頭文字を並べたもので，テキスト形式のデータを，カンマで区切ったものです。例えば，表計算ソフトウェアで，下の左のような表示があれば，これをcsv 形式にすると，右のようなテキストファイルになるわけです。

	A	B	C	D
1	-0.84	-0.38	2.75	2.51
2	1.58	1.19	-2.27	1.11
3	-0.69	1.64	2.65	2.49
4	2.17	-1.77	1.76	0.28

```
-0.84, -0.38, 2.75, 2.51
1.58, 1.19, -2.27, 1.11
-0.69, 1.64, 2.65, 2.49
2.17, -1.77, 1.76, 0.28
```

つまり，Maxima のデータを，カンマで区切ったファイルに吐き出してやれば，表計算ソフトウェアと連動させることができるわけです。このためには，外部ファイルとの連動用のパッケージを読み込む必要がありますが，パスが通っていないので，ファイル名だけでは探せません。ですから，面倒ですがそのファイルまでのフルパスを指定して読み込みます。ただ，オペレーティングシステムによってフルパスが異なります。Linux と MacOS であれば，

```
load("/usr/share/maxima/5.9.2/share/contrib/numericalio/numericalio.lisp");
```

で読み込みます。Microsoft Windows であれば，

```
load("C:/PROGRA~1/MAXIMA~1.2/share/maxima/5.9.2/share/contrib/numericalio/numericalio.lisp");
```

で読み込みます。 すると，csv 形式ファイルをリストとして読み込む函数

```
read_list("ファイル名",'csv);
```

と，行列として読み込む函数

[5]Common LISP をベースとする，Maxima 専用の LISP です。ですから，Emacs LISP など，LISP 系のプログラミングに慣れている方なら，セーブファイルを直接編集することもできます。

```
read_matrix("ファイル名",'csv);
```

そして，リストや行列の成分を csv 形式でファイルに書き出す函数

```
write_data(行列名,"ファイル名",'csv);
```

が追加されます。

例えば，先ほどのデータが C:¥home¥matrix.csv というファイルに納められているとしましょう。すると，次のようにするとデータをリスト lst として取り込めます。

```
(%i217)  lst:read_list("C:/home/matrix.csv",'csv);

(%o217)    [- 0.84, - 0.38, 2.75, 2.51, 1.58, 1.19, - 2.27, 1.11, - 0.69, 1.64,
                              2.65, 2.49, 2.17, - 1.77, 1.76, 0.28]
```

また，次のようにするとデータを行列 mtx として取り込めます。

```
(%i218)  mtx:read_matrix("C:/home/matrix.csv",'csv);

             [ - 0.84  - 0.38   2.75   2.51 ]
             [                              ]
             [  1.58    1.19   - 2.27  1.11 ]
(%o218)      [                              ]
             [ - 0.69   1.64    2.65   2.49 ]
             [                              ]
             [  2.17   - 1.77   1.76   0.28 ]
```

例えば，この取り込んだ行列 mtx の 10 乗を，C:¥home¥tenth.csv というファイルに書き出してみましょう。

```
(%i219)  write_data(mtx^^10,"C:/home/tenth.csv",'csv);

(%o219)                          (false, false, false, false)
```

そして出来上がったファイルを表計算ソフトウェアで開いてみますと，次のようになっています。

	A	B	C	D
1	264557	91681.79	1292301	973105.6
2	26895.85	12995.93	97926.4	81217.32
3	272930.7	102493.4	1350931	1016299
4	248792.7	83024.34	1220861	920140.4

6.5.3　統計処理への応用例

工学や経済学の研究において，実験や調査データを統計処理するために表計算ソフトを利用することは少なくありません。勿論，それが楽ではありますが，数学的に複雑な処理となると，かなり基本的なところからマクロを組む必要性があります。

一方，Maxima も外部データファイルを読み込んだり書き出したりする能力がありますので，データ処理が可能です。しかも，表計算ソフトにできて Maxima にできないデータ処理はありません。データ処理能力ならば表計算ソフトを大きく引き離しているとはいうものの，グラフィカルユーザインターフェイス回りは著しいまでに弱く，外部ファイルを用いたデータ入出力が結構面倒なので，一長一短ではあります。ですから，有効桁数が 3 桁程度で十分であれば，Excel などを利用することをお勧めします。

まずは基本的なところで，csv 形式の外部からデータを読み込んで，その幾何平均を求めること[6]から説明しましょう。はじめに，予めファイル `C:¥home¥geomean.csv` として，`=RAND()` で $(0,1)$ 区間の一様乱数を発生させ，それをオートフィルで 10 列 ×300 行で 3000 個ほど準備します。試しに Microsoft Excel を使って，このデータで `=GEOMEAN(セル範囲)` によって幾何平均を求めようとすると，高い確率で 0 となります[7]。

そして，それを変数 *lst* にリストとして取り込んでみます。

```
(%i220)  lst:read_list("C:/home/geomean.csv",'csv);


(%o221) [0.524791999, 0.83770885, 0.349946256, 0.033510978, 0.322640717,

                    (途中が長いので省略)

          0.882640526, 0.623933517, 0.862401784, 0.859252114, 0.926337907, 0.775912364,

          0.446657813]
```

次に各データの自然対数を求め，それらの算術平均の真数を求めます。ただし，一個でも 0 以下のデータがあればエラーを返すようにしましょう。

```
(%i221)  mygeomean(d):=block([err:1,i],
             for i:1 thru length(d) do(if d[i]=0 then err:0),
             if err=0 then return("Non-positive data exist.")
                 else return(float(exp(sum(log(d[i]),i,1,length(d))/length(lst)))));


(%o222) mygeomean(d) := block([err : 1, i],
        for i thru length(d) do (if d  <= 0 then err : 0),
                                    i
        if err = 0 then return("Non-positive data exist.")
                               sum(log(d ), i, 1, length(d))
                                       i
```

[6] 勿論，幾何平均を求めることに限って言えば，表計算ソフトウェアの函数を使った方が遥かに早いのは確かです。致命的な問題点がある函数ですが，平均利率の計算など，一般的な使用であれば十分なはずです。

[7] 内部ではおそらく，本当に総積を取ってから累乗根を計算しているようです。計算機の限界から，総積が 0 で近似されてしまうために，このような誤りをすると思われます。ですから，Microsoft Excel でもデータの `LN` を取った後，`=EXP(AVERAGE(セル範囲))` とすると，比較的良い計算をします。

```
        else return(float(exp(-----------------------------))))
                                                  length(lst)

(%i222)  mygeomean(lst);

(%o222)                                  0.36896294490398
```

0 から 1 までの擬似的な一様乱数で生成されたデータですから，追実験しても同じ値にならないと思いますが，高い確率で $\frac{1}{e} \fallingdotseq 0.36787944117144$ 前後の値になったと思います。

6.6 章末演習問題

(1) 気温 t を摂氏で与えると，その気温における音速を返す函数 mysound を作りなさい。ただし，気温 t のときの音速は $331.5 + 0.61t$ で得られます。

(2) 1925 年以後の西暦を与えると，明治通算年，大正通算年，昭和通算年を返す函数 myeras を作りなさい。ただし，西暦 2001 年は，明治 134 年，大正 90 年，昭和 76 年に当たります。

(3) 2 変数関数 $z = f(x, y)$ と独立変数の組 $[x, y]$ を与えると停留点を返す函数 mycritpts を作りなさい。ただし，関数 $z = f(x, y)$ について，$\dfrac{\partial z}{\partial x} = 0$, $\dfrac{\partial z}{\partial y} = 0$ を満たす (x_i, y_i) の組が，z の停留点 $(x_i, y_i, f(x_i, y_i))$ を与えます。

(4) 乾球の温度を c_1, 湿球の温度を c_2 を与えると不快の度合いを示す関数 mydiscomfort を作りなさい。不快の度合いは，不快指数 i を $i = 40.6 + 0.72(c_1 + c_2)$ の式で計算し，

- $i < 70$ の場合，快適。
- $70 \leqslant i < 75$ の場合，一部不快。
- $75 \leqslant i < 80$ の場合，半数不快。
- $80 \leqslant i$ の場合，全員不快。

で決めます。

(5) Maxima に 999 以下の自然数を選ばせて，人がその数を推測して入力するたびにより大きいか小さいかを返す，数当てゲーム mybigsmall を作りなさい。

(6) 2 つのリストを与えると，その積集合を返す函数 myintersection を作りなさい。

(7) 2 つのリストを与えると，その和集合を返す函数 myunion を作りなさい。

(8) csv 形式のファイル名と左から何列目を選ぶかを与えると，その列をリストとして返す函数 mycolumn を作りなさい。

(9) read_matrix 函数は，読み込める行列のサイズに限界があります。そこでサイズに余裕がある read_list 函数を利用して，csv 形式のファイル名と列数を与えると，行列を返す函数 myread_matrix を作りなさい。

(10) 4 列の csv 形式のファイルを与えると，各行を係数とする 3 次方程式の解を実数部列，虚数部列に分けた 6 列の csv 形式のファイルを作成する函数 mysolver を作りなさい。ただし，引数は読み込みファイル名と書き出しファイル名を与えるものとします。

付 録A　Mathematica から Maxima へ

市販されている数式処理ソフトウェアの一種である *Mathematica* の函数にほぼ対応する Maxima の函数の一覧です。この対応は完全互換というわけではなく，一部の函数では上手く行かないこともあります。なお，太字で記されているものは Maxima 5.9.2 のみの対応となります。

A.1　演算と数値

Mathematica	Maxima
a+b	a+b;
a-b	a-b;
a*b	a*b;
a/b	a/b;
a^b	a^b;
Sqrt[a]	sqrt(a);
N[a]	float(a);
N[a,b]	fpprec:n; bfloat(a);

A.2　代数

A.2.1　方程式を解く

Mathematica	Maxima
Solve[f[x]==g[x],x]	solve(f(x)=g(x),x);
Solve[{f==g,h==k},{x,y}]	solve([f=g,h=k],[x,y]);
NSolve[f==g,x]	expand(float(solve(f=g,x)));
FindRoot[f(x)==g(x),{x,a}]	load(newton); newton(f(x)-g(x),x,a);

A.2.2　多項式の操作

Mathematica	Maxima
Expand[f[x]]	expand(f(x));
Factor[f[x]]	factor(f(x));
Together[f[x]]	ratsimp(f(x));
Apart[f[x]]	partfrac(f(x));
Cancel[f[x]]	ratsimp(f(x));

A.2.3 式の簡約化

Mathematica	Maxima
`Simplity[f[x]]`	`ratsimp(f(x));`
`FullSimplify[f[x]]`	**`fullratsimp(f(x));`**

A.2.4 複素数

Mathematica	Maxima
`a+b*I`	`a+b*%i;`
`Re[z]`	`realpart(z);`
`Im[z]`	`imagpart(z);`
`Abs[z]`	`cabs(z);`
`Arg[z]`	`carg(z);`
`Conjugate[z]`	`realpart(z)-imagpart(z)*%i;`

A.3 リストと行列

A.3.1 リストや行列の作成

Mathematica	Maxima
`{a,b,c}`	`[a,b,c];`
`{{a,b},{c,d}}`	リスト:`[[a,b],[c,d]];` 行列:`matrix([a,b],[c,d]);`
`Table[a[i],{i,p,q}]`	`makelist(a(i),i,p,q);`
`Table[a[i],{i,p,q,r}]`	`makelist(a(p+(q-p)*r),i,1,(q-p)/r);`

A.3.2 行列の演算

Mathematica	Maxima
`a.b`	`a.b;`
`Cross[a,b]`	`transpose(adjoint(matrix(a,b,[1,1,1])))[3];`
`Outer[f,a,b]`	**`outermap(f,a,b);`**
`Tr[a]`	`load(nchrpl);` `mattrace(a);`
`Det[a]`	`determinant(a);`
`Inverse[a]`	`invert(a);`
`Transpose[a]`	`transpose(a);`
`Eigenvalues[a]`	`load(eigen);` `eigenvalues(a);`
`Eigenvectors[a]`	`load(eigen);` `eigenvectors(a);`

A.4 三角関数と指数関数

A.4.1 三角関数

Mathematica	Maxima
`Sin[x]`	`sin(x);`
`Cos[x]`	`cos(x);`
`Tan[x]`	`tan(x);`
`ArcSin[x]`	`asin(x);`
`ArcCos[x]`	`acos(x);`
`ArcTan[x]`	`atan(x);`
`TrigExpand[f[x]]`	`trigexpand(f(x));`
`TrigReduce[f[x]]`	`trigreduce(f(x));`

A.4.2 指数と対数

Mathematica	Maxima
`Log[x]`	`log(x);`
`Log[10,x]`	`log(x)/log(10);`
`Exp[x]`	`exp(x);`

A.4.3 双曲線関数

Mathematica	Maxima
`Sinh[x]`	`sinh(x);`
`Cosh[x]`	`cosh(x);`
`Tanh[x]`	`tanh(x);`
`ArcSinh[x]`	`asinh(x);`
`ArcCosh[x]`	`acosh(x);`
`ArcTanh[x]`	`atanh(x);`

A.5 微分積分

A.5.1 通常の演算

Mathematica	Maxima
`D[f[x],x]`	`diff(f(x),x);`
`Integrate[f[x],x]`	`integrate(f(x),x);`
`Integrate[f[x],{x,a,b}]`	`integrate(f(x),x,a,b);`
`Sum[x[k],{k,a,b}]`	`sum(x(k),k,a,b);`
`Product[x[k],{k,a,b}]`	`product(x(k),k,a,b);`
`Limit[f[x],x->a]`	`limit(f(x),x,a);`
`Series[f[x],{x,a,n}]`	`taylor(f(x),x,a,n);`

A.5.2 微分方程式

Mathematica	Maxima
`DSolve[f[y[x]]==0,y[x],x]`	`desolve(f(y(x))=0,y(x));` または `ode2(f(y(x))=0,y(x),x);`
`DSolve[{f==0,g==0},y[x],x]`	`desolve([f=0,g=0],y(x));`
`DSolve[{f==0,y[0]==a},y[x],x]`	`atvalue(y(x),x=0,a);` `desolve(f=0,y(x));`

A.5.3 変換

Mathematica	Maxima
`LaplaceTransform[f[t],t,s]`	`laplace(f(t),t,s);`
`InverseLaplaceTransform[g[s],s,t]`	`ilt(g(s),s,t);`
`FourierTransform[f[t],t,w]`	`load(fft);` `fft(f(t),t,w);`
`InverseFourierTransform[g[w],w,t]`	`load(fft);` `ift(f(t),t,w);`

A.6 その他の関数

A.6.1 整数関数

Mathematica	Maxima
`Round[x]`	`?round(x);`
`Mod[n,p]`	`mod(n,p);`
`GCD[a,b]`	`gcd(a,b);`
`LCM[a,b]`	`lcm(a,b);`
`FactorInteger[n]`	`factor(n);`
`Rationalize[x]`	`ratsimp(x);`

A.6.2 特殊関数

Mathematica	Maxima
`n!`	`n!;`
`BesselI[v,z]`	`bessel_i(v,z);`
`BesselJ[v,z]`	`bessel_j(v,z);`
`BesselK[v,z]`	`bessel_k(v,z);`
`BesselY[v,z]`	`bessel_y(v,z);`
`Erf[x]`	`erf(x);`
`Gamma[x]`	`gamma(x);`
`Zeta[x]`	`zeta(x);`

A.7 グラフィクス

Mathematica	Maxima
`Plot[y,{x,a,b}]`	`plot2d(y,[x,a,b]);`
`Plot[{y1,y2},{x,a,b}]`	`plot2d([y1,y2],[x,a,b]);`
`ParametricPlot[{x,y},{t,a,b}]`	`plot2d([parametric,x,y],[t,a,b]);`
`Plot3D[z,{x,a,b},{y,p,q}]`	`plot3d(z,[x,a,b],[y,p,q]);`
`ParametricPlot3D[{x,y,z},{s,a,b},{t,p,q}]`	`plot3d([x,y,z],[s,a,b],[t,p,q]);`
`ListPlot[x]`	`openplot_curves(x);`

付 録B　インストールについて

B.1　Maxima のインストール

これから Maxima の標準的なインストール方法を説明します。以下は，2005 年 10 月現在の情報で，最新の Maxima のバージョンは 5.9.2 です。

B.1.1　Linux の場合

```
http://sourceforge.net/project/showfiles.php?group_id=4933
```

に rpm 版バイナリがありますので，それをダウンロードして下さい。折角ですから，`GCL 5.9.2` と `XMaxima 5.9.2` も同時にダウンロードすることをお勧めします。後は，通常のパッケージと同じように，三つともインストールします。

```
# rpm -ivh maxima-5.9.2-2.i386.rpm
# rpm -ivh maxima-exec-gcl-5.9.2-2.i386.rpm
# rpm -ivh maxima-xmaxima-5.9.2-2.i386.rpm
```

これで Maxima を利用することができますが，`~/.bashrc` の末尾にでも，

```
alias xmaxima="xmaxima > /dev/null"
```

を加えておくと便利でしょう。

また，CD-ROM 起動の Linux である，KNOPPIX の中で，KNOPPIX/Math という数学に特化したものがあります。これには，Maxima 5.9.1 だけではなく，gnuplot なども一緒にインストールされています。ですから，PC/AT 互換機であれば，BIOS 設定として CD-ROM bootable にしておいて，これを利用するのが最も楽です(1)。

Linux ZAURUS の場合

素晴らしいことに，SHARP 社から発売されている Linux ベースの電子手帳，ZAURUS SL シリーズでもターミナル版の Maxima 5.9.0 を利用することができます。そのためには，

```
http://web.njit.edu/~rxt1077/clisp-maxima-zaurus.html
```

(1) ってゆーか，筆者も一枚噛んでいるんで，できたら KNOPPIX/Math 使ってちょーだい。

からまず

```
clisp-maxima-zaurus.tar.bz2
```

をダウンロードして, 解凍してできた lisp.run を /usr/local/bin に, maxima-clisp.mem を /home/zaurus にそれぞれ置いて下さい(2)。これで Maxima を利用することができますが, /home/zaurus/.bashrc の末尾にでも,

```
alias maxima='lisp.run -M /home/zaurus/maxima-clisp.mem'
```

を書き加えておけば(3), ターミナルを立ち上げた後, maxima とタイプインするだけで Maxima を利用することができます。

ただし, ターミナル版なので, plot2d や plot3d の類いは, 必ずオプションとして [plot_format,gnuplot] を指定し, 表示は gnuplot で行います。また, ヘルプがついていませんので, ご注意ください。

B.1.2　MacOS X の場合

現在 MacOS X 用の初心者向けインストーラは供給されていません(4)。ですが, 必要なライブラリをダウンロードすれば, Linux 用の Maxima のソースをコンパイルすることで利用することができます。なお, MacOSX で Maxima 5.9.2 のインストールは, まだ筆者は成功していませんので, 以下で述べる情報は Maxima 5.9.1 のものです。

まず, ターミナルから

```
% which make
```

と入力して, C コンパイラが準備できているかどうかを確認します。

```
/usr/bin/make
```

と出てきたら準備ができています。次に, Maxima を動かす心臓部である CLISP をインストールします。

そのためには下準備として, libsigsegv 2.1, readline 5.0, gettext 0.14.1 をインストールしなければなりません。

```
ftp://ftp.gnu.org/pub/gnu/libsigsegv/
```

(2)本体メモリ残量が少なければ, 他の SD メモリやコンパクトフラッシュメモリのどこかに置いても利用できますが, 実行速度に少々問題があります。本体メモリに 3MB 以上の余裕があれば, /home/zaurus に置くのがお勧めです。

(3)SD メモリやコンパクトフラッシュメモリに置いた場合は, /mnt/card/maxima-clisp.mem や /mnt/cf/maxima-clisp.mem のようにパスを書き換えて下さい。

(4)筆者が見つけられなかっただけなのかも知れません。情報をお持ちの方がおられましたら, 是非教えて下さい。

から libsigsegv-2.1.tar.gz をダウンロードして，解凍した後，ターミナルから，

```
% ./configure

% make
% sudo make install
```

で libsigsegv 2.1 をインストールします。次に，

ftp://ftp.gnu.org/pub/gnu/readline/

から readline-5.0.tar.gz をダウンロードして，解凍した後，ターミナルから，

```
% ./configure

% make
% sudo make install
```

で readline 5.0 をインストールします。さらに，

ftp://ftp.gnu.org/pub/gnu/gettext/

から gettext-0.14.1.tar.gz をダウンロードして，解凍した後，ターミナルから，

```
% ./configure

% make
% sudo make install
```

で gettext 0.14.1 をインストールします。

そして，CLISP 本体のインストールです。

ftp://ftp.gnu.org/pub/gnu/clisp/release/2.33.2/

から clisp-2.33.2.tar.gz をダウンロードし，続いて，

http://darwinports.opendarwin.org/darwinports/dports/lang/clisp/files/

から patch-src-stream.d をダウンロードします。clisp-2.33.2.tar.gz を解凍した後，ターミナルから，

```
% cd clisp-2.33.2

% patch -p0 < ../patch-src-stream.d
% ./configure
% cd src
% ./makemake --with-dynamic-ffi --with-readline --with-gettext --with-module=regexp
 --with-export-syscalls > Makefile
```

```
% make config.lisp
% make
% sudo make install
```
で clisp 2.33.2 をインストールします。

最後に Maxima のインストールです。折角ですから，Tcl/Tk 8.4.9 もインストールして，XMaxima を使えるようにしましょう。

```
http://tcltkaqua.sourceforge.net/
```

から TclTkAqua-8.4.9.dmg をダウンロードし，出てきたディスクイメージのアイコンをダブルクリックしてインストールします。そして，

```
http://maxima.sourceforge.net/download.shtml
```

から Maxima のソースコードをダウンロードして，ターミナルから
```
% ./configure

% make
% sudo make install
```
と入力することで，すべてのインストールが完了です。これで XMaxima を利用することができますが，

~/.bashrc の末尾にでも，

```
alias xmaxima="xmaxima > /dev/null"
```

を加えておくと便利でしょう。

B.1.3　Microsoft Windows の場合

Microsoft Windows 用の Maxima は，英語版 gnuplot まで含めたインストーラがありますので，それを利用するのが楽でしょう。

```
http://maxima.sourceforge.net/download.shtml
```

から Windows 用インストーラをダウンロードします。

コンピュータ本体にインストールするならば，ライセンス確認のときに I accept the agreement を選ぶということ以外は，単純に Next をクリックし続けるだけでいいでしょう。また，USB フラッシュメモリに 70MB 以上の空きがあれば，起動が少々遅くなりますが，そちらにインストールすることもできます。そうすると，Maxima と gnuplot をセットで持ち歩けます。

B.2　gnuplot のインストール

以下は，2005 年 10 月現在の情報で，最新の gnuplot のバージョンは 4.0.0 です。

B.2.1 Linux の場合

```
ftp://ftp.pbone.net/mirror/www.arklinux.org/1.0-0.alpha12.2/i586/
```

から，gnuplot-4.0.0-1ark.i586.rpm をダウンロードして，通常のパッケージと同じようにインストールします。つまり，# rpm -ivh gnuplot-4.0.0-1ark.i586.rpm

で，インストールできます。

Linux ZAURUS の場合

```
http://www.mneuroth.de/privat/zaurus/gnuplot.html
```

から，qtplot_0.2_arm.ipk をダウンロードして，ZAURUS のインストーラで普通にインストールしてください。

B.2.2 MacOS X の場合

```
http://sourceforge.net/project/showfiles.php?group_id=2055&package_id=1996
```

から，Gnuplot-4.0.0.dmg をダウンロードして，出てきたディスクイメージのアイコンをダブルクリックしてインストールします。

付 録C　トラブルシューティング

C.1　システム絡みのトラブル

Q.1. Maxima って何ですか。

Maxima とは，数式処理ソフトウェアと呼ばれるアプリケーションの一種です。といいますのか，このノートはそれを説明するために書いたものです。このノートを最初からお読み下さい。

Q.2. Maxima が見当たりません。

各オペレーティングシステムで次のように対処します。

Linux や MacOSX の場合。

シェルから

```
which xmaxima
```

と入力します。このとき，/usr/local/bin/xmaxima などパス名が現れるようでしたら，インストールはできていますから，そのままシェルで

```
xmaxima &
```

とタイプインすればよいでしょう。

もしインストールした記憶があるのに，which xmaxima に対して無反応であれば，オペレーティングシステムに Maxima を探索する場所[1]が登録されていない可能性があります。まずは，どこにインストールされているか，しらみつぶしに探索してもらうために，シェルで

```
find / -name xmaxima
```

と入力します。しばらくすると，/usr/math/bin/xmaxima などフルパス名がでてきます。それをフルパスのまま入力すればいいのですが，~/.bashrc の末尾にでも，

```
alias xmaxima="/usr/math/bin/xmaxima > /dev/null"
```

のように加えておくと便利でしょう。

また，MacOSX であれば，

Linux でも同様のアイコン作成ができますが，具体的にどうするかは利用しているウィンドウマネージャに依存しますので，そちらのヘルプを参照してください。

[1]パス (path) といいます。

Microsoft Windows の場合。

スタートボタンから プログラム → Maxima-(バージョン名) → XMaxima-(バージョン名) と選びます。

このスタートメニューが見当たらない時は，インストール時にスタートメニューへの登録を選択し忘れたか，スタートメニューを編集してしまったかのいずれかです。このときは，スタートメニューから，ファイルやフォルダの検索を選び，xmaxima.exe を検索します。検索されたら，それをダブルクリックすると起動できます。ですが，折角ですから右クリックからショートカットの作成を選んで，デスクトップにアイコンを作成しておくと便利でしょう。

Q.3. Maxima をインストールするときに容量不足の警告が出ます。

Maxima 5.9.2 をインストールするには，ハードディスクを 60 MB ほど必要とします。従って，普通のアプリケーションインストールのトラブル同様，不要なファイルを削除[(2)]してください。

Q.4. Maxima を立ち上げるとメモリ不足の警告が出ます。

Maxima 5.9.2 は，計算の心臓部である `maxima` とグラフィカルなインターフェイスを含めて，起動時にメモリを 15 MB ほど要求します。従って，普通のアプリケーション立ち上げ時のトラブル同様，他のアプリケーションを終了させてメモリを開放[(3)]してください。

Q.5. Maxima を立ち上げると Out of environment space というエラーが出ます。

Microsoft Windows 95/98/98SE にインストールした場合，MS-DOS の環境変数に十分なメモリが割り当てられていないときに起こります。Maxima-5.9.1 のフォルダの readme.txt にも英語で書いてありますが，解決するには，

(1) スタートボタンから，[スタート] → [ファイル名を指定して実行 (R)] を選びます。
(2) 出てきたパネルに `sysedit` と打ち込み，[OK] をクリックします。
(3) システム環境を決定する重要なファイルが次々出てきますが，その中で `CONFIG.SYS` というファイルをクリックしてアクティブにします。
(4) その中に，`SHELL=` で始まる行があったら，冒頭に `REM` を書き込んでコメントアウトします。もちろん，`REM` の直後の半角スペースも忘れないようにして下さい。
(5) `CONFIG.SYS` ファイルの冒頭に，

```
SHELL=C:¥COMMAND.COM /E:4096 /P
```

という行を書き加えて下さい。
(6) メニューから [ファイル (F)] → [上書き保存 (S)] を選びます。
(7) 続けて，メニューから [ファイル (F)] → [システムエディタの終了 (X)] を選びます。
(8) コンピュータを再起動して下さい。

[(2)] ハードディスクを圧迫している「不要なファイル」とは，大抵の場合，怪しい画像だとか音楽ファイルだとか，いわゆる「お宝ファイル」ではないかと思われますが。

[(3)] Microsoft Windows において，最もメモリを圧迫する悪質なプログラムは，大抵の場合，オペレーティングシステムそのものであることが多いので，誤ってオペレーティングシステムを終了させないようにしましょう。

Q.6. Maxima で発動するコンピュータウイルスはありますか。

少なくとも現在のところ，筆者は聞いたことがありません。Maxima そのものにはディレクトリやファイルの消去や書き換える能力はありません。しかし，Maxima の背景にある CLISP という言語にはその能力があるので，原理的には破壊活動が可能です。そんな Maxima にまで対応しているウイルス対策ソフトウェアはないと思いますから，将来的には可能性がないとも言い切れません。

C.2 操作方法に関するトラブル

Q.1. 計算実行したところ，実行中のまま計算終了してくれません。

まずは， Maxima の起動と終了で説明した，Ctrl + g を試してください。場合によっては，停止するまで少々時間がかかることがあります。それで駄目なら,「1.1 Maxima の起動と終了」を参考に，オペレーティングシステムレベルで，maxima のプロセス，xmaxima のプロセスの順に強制終了してください。

Q.2. Maxima を終了させたのに，マシンが急に重くなりました。

おそらく計算実行中に xmaxima を強制終了させた可能性が高いです。Maxima は，それ自身をシェルで立ち上げることもあれば，xmaxima の子プロセスとして立ち上げることもあります。Maxima は OS によっては

```
quit();
```

で終了させるまでプロセスが残っていますから，Maxima のプロセスが残っているのに親プロセスの xmaxima を終了すると，単独で暴走する可能性があります。このときは，オペレーティングシステムレベルで，Maxima のプロセスを選んで強制終了してください。またこれは，初心者が起こしやすいミスで，重い計算の停止方法が分からずに，強制的に xmaxima のウィンドウを終了させたときに起こやすい現象です。

Q.3. もう少し大きなフォントになりませんか。

まず，XMaxima のメニューにある，[Option] → [Preferences] でパネルを出します。

下半分のブラウザの文字フォントを変えるには，The proportional font is の後の [Arial] をクリックして読みやすいものを選びましょう。変えるとしたら，筆者のお勧めは [Times New Roman] です。その大きさを決めるのは，続く with a size adjustment of の後の [0] をクリックしてサイズを決めます。数値が大きいほど大きなフォントになります。

上半分の入力および出力部分の文字フォントを変えるには，The fixed font is の後の [Courier NEW] をクリックして読みやすい固定幅フォントを選びましょう。変えるとしたら，筆者のお勧めは [Terminal] です。その大きさを決めるのは，続く with a size adjustment of の後の [0] をクリックしてサイズを決めます。数値が大きいほど大きなフォントになりますので，授業で学生に見せるときに使うなら，[5] にしましょう。

その後，[Apply and Quit] を押すと変更完了です。

Q.4. XMaxima のウィンドウサイズが不便です。

XMaxima は Maxima に Tcl/Tk[4]でウィンドウを被せたものですから，他のアプリケーションのようにウィンドウのリサイズができます。ウィンドウ全体の大きさも変えられますし，上の入出力部と下のブラウザとの境界もマウスでリサイズできます。

ただ，横幅を狭くしたとき，79 文字幅以下にしてしまうと，一行に長い出力が出たときに出力が乱れます。そのときは，

```
linel:文字数;
```

で適切な文字数にして下さい。

C.3 函数に関するトラブル

Q.1. 「函数」とは何ですか。

まず，漢字のお勉強から。「函数」は「かんすう」と読みます[5]。英語で言うところの function のことですが，このノートでは数学の function とプログラムの function を区別するために，数学の方[6]を「関数」で，プログラムの方[7]を「函数」と記述してあります。

Q.2. 使いたい函数名の見当がつきません。

Maxima は，1000 個を超える函数や変数が組み込まれています。折角あるのですから，それらを使わない手はありません。XMaxima は，上下のウィンドウに別れており，下がヘルプになっています。Maxima Primer の 2 行目にある “documentation” のリンクをクリックしますと，機能別に整理されたヘルプ ただし，思いっきり英語ですので，英語が苦手な人は泣きましょう。なお，筆者も心ひそかに泣いている一人ですから。が出てきます。そこから系統的に推定することができるでしょう。なお，すべて印刷しないと気が済まないお方は，

```
http://maxima.sourceforge.net/docs/manual/en/maxima.pdf
```

に pdf 形式のファイルが提供されています。A4 で 460 ページを超える大作ですので，紙資源と相談しながら印刷してください。また，Maxima のバージョンが，5.4 の古いものでもよければ，

```
http://www.bekkoame.ne.jp/~ponpoko/Math/maxima/maxima_toc.html
```

に，ぽんぽこ氏による日本語訳が提供されています。

---

[4] ティックルティーケーと読みます。
[5] 驚くなかれ,「その漢字は，何と読むんですか」というのは，大学生から本当にあった質問です。その後，私「何と読むと思いますか」学生「えっと‥‥キョクスウ？」という会話があったことは内緒。
[6] 集合から集合への写像という意味で用いました。
[7] 与えられた引数から一連の手順に従ってデータ処理を実行させるものという意味で用いました。

Q.3. 函数のオプションには何があるのでしょうか。

オンラインヘルプでも説明しましたように，? に続けて函数名を入力すれば，ヘルプの一部が出てきます。そこに，それぞれの函数で利用されるオプションの説明があります。

Q.4. Maxima の計算結果がおかしいです。

Maxima はセッション内で利用した変数の値のすべてを記憶しています。ですから，すでに利用した変数名や定義した函数名を (利用していたことを忘れて)，再利用してしまうと以前の値を代入されてしまいます。一般的には，そこで使いたい変数や函数を全部 kill で消去してから，再定義させるといいでしょう。

それでもおかしいときは，バグの可能性があります。是非，

```
http://sourceforge.net/tracker/?atid=104933&group_id=4933&func=browse
```

からバグを**英語で**[8]報告してください。また，報告のときに

```
bug_report();
```

として出てくる Maxima 自身の情報，例えば，

```
Maxima version: 5.9.2
Maxima build date: 9:5 10/12/2005
host type: i686-pc-mingw32
lisp-implementation-type: GNU Common Lisp (GCL)
lisp-implementation-version: GCL 2.6.7
```

という情報も一緒に報告してください。

Q.4. 変数を利用しようとするとエラーがでました。

Maxima には 1000 個以上の函数が組み込まれており，一部の函数や文字定数は保護されています。なお，変数名として日本語のローマ字を使うと，大抵の場合，引っかからないので初心者にはそれがお勧めです。

C.4 その他のトラブル

Q.1. 個人的に Maxima が欲しくなってきました。

Microsoft Windows にこだわらないのでしたら，多少古いコンピュータ[9]でも，Maxima は動きます。筆者の確認したところでは，

- CPU クロック数：266 MHz
- メモリ：64MB

[8] Maxima は，世界中の有志がバグ修正に協力しています。
[9] え。コンピュータをお持ちでない，ですと。それは流石にお金を出して買って下さい。コンピュータまでは無料ではありません。

- ハードディスク：2 GB
- オペレーティングシステム：Vine Linux 2.6[10]

で，Tcl/Tk と CLISP をバージョンアップしただけで問題なく利用できています[11]。それ以上の能力のマシンであれば，付録の「インストールについて」を参考に，Maxima をダウンロードして，インストールしてください。

Q.2. Maxima を使っていると気分が悪くなってきました。

原因は二つ考えられます。一つは，ドライアイなどの眼精疲労です。Maxima は指数や分数などを何行にもわたって表示するため，他のアプリケーションに比べても，テキストのスクロール量は多い方です。コンピュータ画面のことですから，最悪，網膜剥離の危険性があるため，適切に休息するように指示してください。もしも顔色が悪いよう[12]でしたら，医師に相談してください。

もう一つは純粋にメンタルなもので，単なる数学嫌いか，コンピュータ嫌いか，勉強嫌いくらいなものです。二，三発，天誅を加えてあげますから，かかってきなさい。

Q.3. 計算結果やグラフィクスを TEX で利用したいです。

計算結果のみを利用したいのであれば，計算結果を引数として，

```
tex(計算式);
```

に引き渡せば，plain TEX の数式環境の コマンドで記述された結果が得られます。勿論，LATEX でもそのまま使えますので，それを TEX ソースの必要な個所にコピー，ペーストすればよいでしょう。例えば，$\int \frac{1}{x^4+1}\,dx$ の計算結果を，TEX に取り込みたければ次の様に入力します。

```
(%i223)  tex(integrate(1/(x^4+1),x));
```

```
$${{\log \left(x^2+\sqrt{2}\,x+1\right)}\over{4\,\sqrt{2}}}-{{\log
 \left(x^2-\sqrt{2}\,x+1\right)}\over{4\,\sqrt{2}}}+{{\arctan \left(
 {{2\,x+\sqrt{2}}\over{\sqrt{2}}}\right)}\over{2\,\sqrt{2}}}+{{
 \arctan \left({{2\,x-\sqrt{2}}\over{\sqrt{2}}}\right)}\over{2\,
 \sqrt{2}}}$$
```

```
(%o223)                                  FALSE
```

グラフィクスを利用したいのであれば，このノートの「2.5 gnuplot との連動」を参考にして下さい。

[10] 昔の UNIX Magazine に付属していた CD-ROM を利用しました。
[11] たまたま，職場で Windows 98SE マシンとしては能力不足ということで廃棄寸前だったものを蘇らせただけです。というのか，Microsoft Office 2000 ですら苦しいマシンですから。
[12] あくまでも顔色ですよ。顔の造作についてはご両親とご相談ください。

付 録D　演習問題の解答例

D.1　Maxima の基本

1. オンラインヘルプ

函数 solve とは何か，オンラインヘルプで調べなさい。

```
(%i1)  ?  solve


 0: desolve :(maxima.info)Definitions for Differential Equations.
 1: fast_linsolve :Definitions for Affine.
 2: funcsolve :Definitions for Equations.
 3: globalsolve :Definitions for Equations.
 4: linsolve :Definitions for Equations.
 5: linsolve_params :Definitions for Equations.
 6: linsolvewarn :Definitions for Equations.
 7: solve :Definitions for Equations.
 8: solve_inconsistent_error :Definitions for Equations.
 9: solvedecomposes :Definitions for Equations.
 10: solveexplicit :Definitions for Equations.
 11: solvefactors :Definitions for Equations.
 12: solvenullwarn :Definitions for Equations.
 13: solveradcan :Definitions for Equations.
 14: solvetrigwarn :Definitions for Equations.
Enter space-separated numbers, 'all' or 'none':
Info from file C:/PROGRA~1/MAXIMA~1.2/info/maxima.info:
 -- Function: solve (<expr>, <x>)
 -- Function: solve (<expr>)
 -- Function: solve ([<eqn_1>, ..., <eqn_n>], [<x_1>, ..., <x_n>])
     Solves the algebraic equation <expr> for the variable <x> and
     returns a list of solution equations in <x>.  If <expr> is not an
     equation, the equation '<expr> = 0' is assumed in its place.  <x>
     may be a function (e.g. 'f(x)'), or other non-atomic expression
     except a sum or product. <x> may be omitted if <expr> contains
     only one variable.  <expr> may be a rational expression, and may
     contain trigonometric functions, exponentials, etc.

     The following method is used:
```

```
Let <E> be the expression and <X> be the variable.  If <E> is
linear in <X> then it is trivially solved for <X>.  Otherwise if
<E> is of the form 'A*X^N + B' then the result is '(-B/A)^1/N)'
times the 'N''th roots of unity.

If <E> is not linear in <X> then the gcd of the exponents of <X>
in <E> (say <N>) is divided into the exponents and the
multiplicity of the roots is multiplied by <N>.  Then 'solve' is
called again on the result.  If <E> factors then 'solve' is called
on each of the factors.  Finally 'solve' will use the quadratic,
cubic, or quartic formulas where necessary.
```

(途中省略)

```
(%i8) ev (x^6 - 1, %[1]);
                                        6
                          (sqrt(3) %i + 1)
(%o8)                     ---------------- - 1
                                 64
(%i9) expand (%);
(%o9)                              0
(%i10) x^2 - 1;
                                   2
(%o10)                            x  - 1
(%i11) solve (%, x);
(%o11)                     [x = - 1, x = 1]
(%i12) ev (%th(2), %[1]);
(%o12)                             0

(%o1)                              false
```

2. 四則演算と指数乗

(1) $2735 + 1846 - 3099$

```
(%i2)  2735+1846-3099;

(%o2)                                  1482
```

(2) $2.35 \times 3.91 - 1.83 \times 7.01$

```
(%i3)  2.35*3.91-1.83*7.01;
```

```
(%o3)                              - 3.639799999999999
```

(3) $\dfrac{6174}{58+89}$

```
(%i4)  6174/(58+59);
                                                686
(%o4)                                           ----
                                                13
```

(4) $2.21^{\frac{1}{3}} \times 3.44^{1.25}$

```
(%i5)  2.21^(1/3)*3.44^1.25;

(%o5)                              6.102327023772617
```

(5) $\left(\dfrac{1}{3^{\frac{1}{2}}+1}\right)^{\frac{1}{3}}$

```
(%i6)  (1/(3^(1/2)+1))^(1/3);
                                                1         1/3
(%o6)                                   ------------------
                                          (sqrt(3) + 1)
```

3. 近似計算

(1) $\sqrt{\dfrac{\pi}{2}}$

```
(%i7)  fpprec:30;

(%o7)                                           30


(%i8)  bfloat(sqrt(%pi/2));

(%o8)                  1.25331413731550025120788264241B0
```

(2) $\cos(\sin(2.63^3))$

```
(%i9)  fpprec:30;

(%o9)                                           30


(%i10)  bfloat(cos(sin(2.63^3)));

  Warning:  Float to bigfloat conversion of 0.81871785281177967
```

```
(%o10)                    8.187178528117796268465595210 6B-1
```

(3) $\log_{10}(42)$

```
(%i11) fpprec:30;
(%o11)                                    30

(%i12) bfloat(log(42)/log(10));
(%o12)                    1.623249290397900463220983056 57B0
```

(4) $16\arctan\left(\frac{1}{5}\right) - 4\arctan\left(\frac{1}{239}\right)$

```
(%i13) fpprec:30;
(%o13)                                    30

(%i14) bfloat(16*atan(1/5)-4*atan(1/239));
(%o14)                    3.141592653589793238462643383 28B0
```

(5) $e^{\sqrt{163}\pi}$

```
(%i15) fpprec:30;
(%o15)                                    30

(%i16) bfloat(exp(sqrt(163)*%pi));
(%o16)                    2.625374126407687439999999999 999B17
```

4. 代入計算

(1) 文字変数 t に $\cos(30°)$ の値を代入しなさい。

```
(%i17) t:cos(30*%pi/180);
                                        sqrt(3)
(%o17)                                  -------
                                          2
```

(2) 文字変数 c に $\sqrt{\dfrac{1+t}{2}}$ の値を代入しなさい。

```
(%i18)  c:ratsimp(sqrt((1+t)/2));
                                      sqrt(sqrt(3) + 2)
(%o18)                               ------------------
                                             2
```

(3) 文字変数 s に $\sqrt{\dfrac{1-t}{2}}$ の値を代入しなさい。

```
(%i19)  s:ratsimp(sqrt((1-t)/2));
                                       sqrt(sqrt(3) - 2)
(%o19)                              - ------------------
                                              4
```

(4) $2cs$ の値を求めなさい。

```
(%i20)  ratsimp(2*c*s);
                             sqrt(2 - sqrt(3)) sqrt(sqrt(3) + 2)
(%o20)                       -----------------------------------
                                             2
```

(5) 文字変数 t, c, s と値の関係を消去しなさい。

```
(%i21)  kill(t,c,s);

(%o21)                                  DONE
```

5. 関数定義

(1) 引き数 x を与えると，それを度数法の角度として，対応する弧度法の角度 $\dfrac{x\pi}{180}$ を返す関数 $p(x)$ を定義しなさい。

```
(%i22)  p(x):=x*%pi/180;
                                          x %pi
(%o22)                             p(x) := ------
                                           180
```

(2) 引き数 x を与えると，それを華氏の温度として，対応する摂氏の温度 $\dfrac{5}{9}(x-32)$ を返す関数 $q(x)$ を定義しなさい。

```
(%i23)  q(x):=float(5/9*(x-32));
                                            5
(%o23)                      q(x) := float(--(x - 32))
                                            9
```

(3) 引き数 x, y, z を与えると，それらを成分にもつ 3 次元ベクトル (x, y, z) の大きさ $\sqrt{x^2+y^2+z^2}$ を返す関数 $r(x, y, z)$ を定義しなさい。

```
(%i24)  r(x,y,z):=sqrt(x^2+y^2+z^2);
                                                  2    2    2
(%o24)                          r(x, y, z) := sqrt(x  + y  + z )
```

(4) 引き数 a, b, c, x, y, z を与えると，それらを成分にもつ二つの 3 次元ベクトル (a, b, c), (x, y, z) がなす角度 $\cos^{-1}\left(\dfrac{ax+by+cz}{r(a,b,c)r(x,y,z)}\right)$ を返す関数 $s(a, b, c, x, y, z)$ を定義しなさい。

```
(%i25)  s(a,b,c,d,e,f):=float(acos((a*x+b*y+c*z)/(r(a,b,c)*r(x,y,z))));
                                                      a x + b y + c z
(%o25)          s(a, b, c, d, e, f) := float(acos(----------------------))
                                                   r(a, b, c) r(x, y, z)
```

(5) 関数 $p(x)$, $q(x)$, $r(x, y, z)$, $s(a, b, c, x, y, z)$ と関数式の関係を消去しなさい。

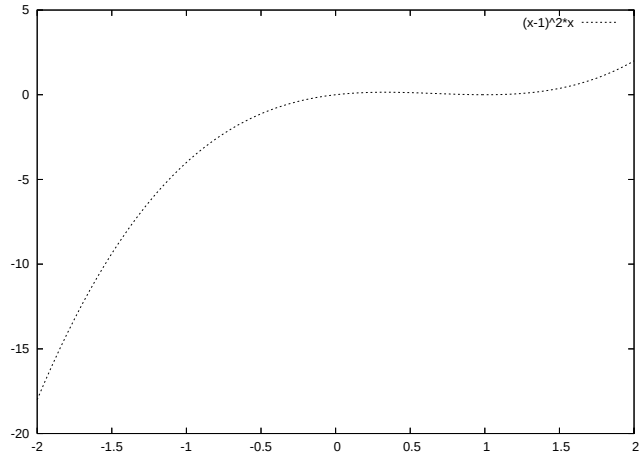
```
(%i26)  kill(p,q,r,s);

(%o26)                                  DONE
```

D.2 グラフの描画

1. $y = f(x)$ のグラフ

(1) 関数 $y = x(x-1)^2$ のグラフを $-2 \leqslant x \leqslant 2$ の範囲で描画しなさい。

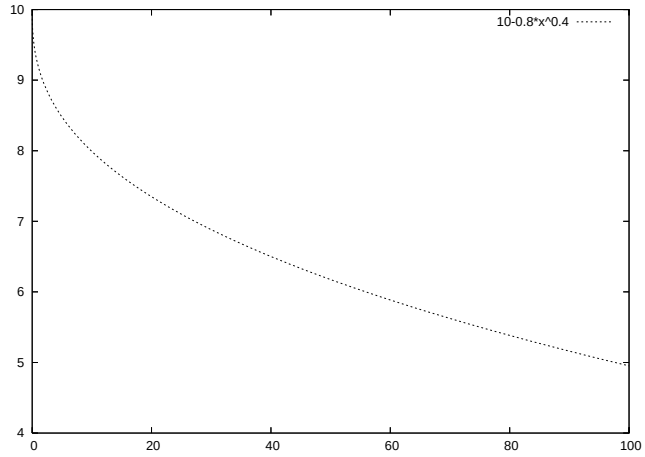
```
(%i27)  plot2d(x*(x-1)^2,[x,-2,2]);
```



(%o27)

(2) 需要曲線 $y = 10 - 0.8x^{0.4}$ のグラフを $0 \leqslant x \leqslant 100$ の範囲で描画しなさい。

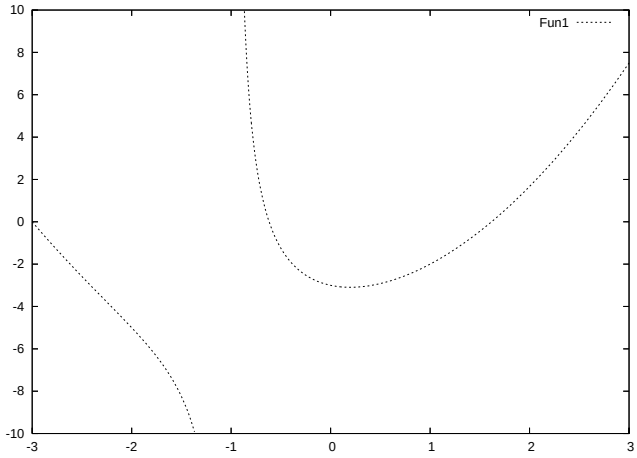
```
(%i28)  plot2d(10-0.8*x^0.4,[x,0,100]);
```




```
(%o28)
```

(3) 有理関数 $y = \dfrac{x^3 + 2x^2 - 4x - 3}{x + 1}$ のグラフを $-3 \leqslant x \leqslant 3$ の範囲で描画しなさい。

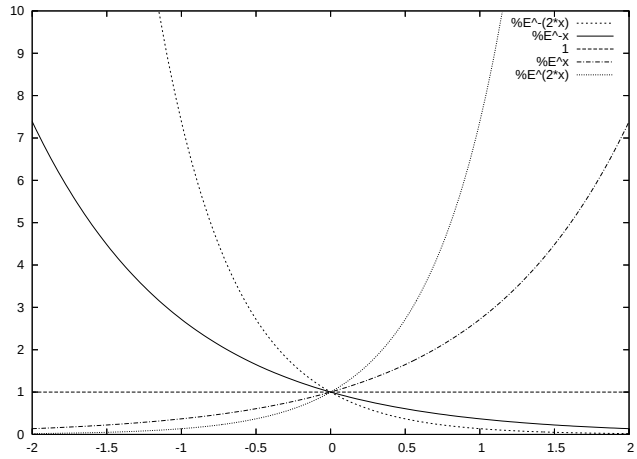
```
(%i29)  plot2d((x^3+2*x^2-4*x-3)/(x+1),[x,-3,3],[y,-10,10]);
```




```
(%o29)
```

(4) 次の 5 つの指数函数 $y = e^{kx}, (k = -2, -1, 0, 1, 2)$ のグラフを $-2 \leqslant x \leqslant 2$ の範囲で一緒に描画しなさい。

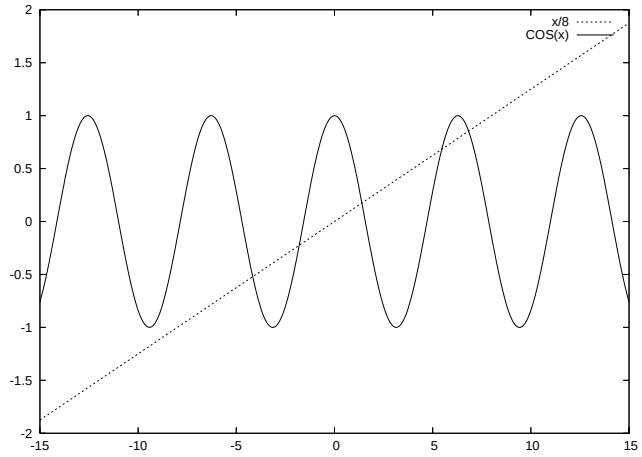
```
(%i30)  plot2d([exp(-2*x),exp(-x),1,exp(x),exp(2*x)],[x,-2,2],[y,0,10]);
```




```
(%o30)
```

(5)　方程式 $\dfrac{x}{8} = \cos(x)$ が実数解をいくつ持つか求めなさい。

```
(%i31)  plot2d([x/8,cos(x)],[x,-15,15]);
```



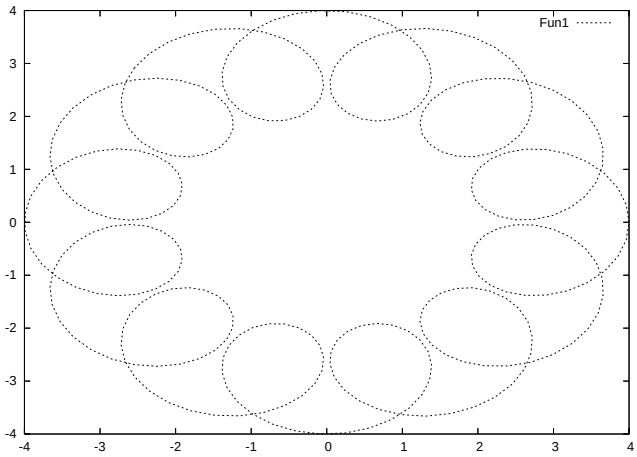

```
(%o31)
```

上のグラフより，5 つの実数解を持つことが分かります。

2. $x = f(t)$, $y = g(t)$ のグラフ

(1)　関数 $x = 3\cos(t) + \cos(13t)$, $y = 3\sin(t) + \sin(13t)$ のグラフを $0 \leqslant t \leqslant 2\pi$ の範囲で描画しなさい。

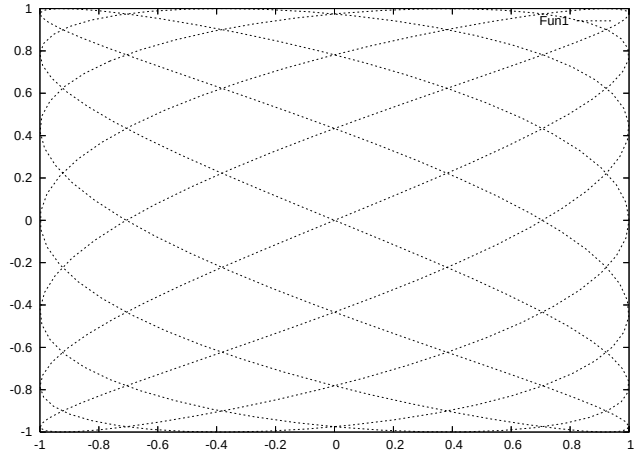
```
(%i32)  plot2d([parametric,3*cos(t)+cos(13*t),3*sin(t)+sin(13*t)],[t,0,2*%pi]
        ,[nticks,300]);
```




```
(%o32)
```

(2) リサージュ曲線 $x = \cos(7t)$, $y = \sin(4t)$ のグラフを $0 \leqslant t \leqslant 2\pi$ の範囲で描画しなさい。

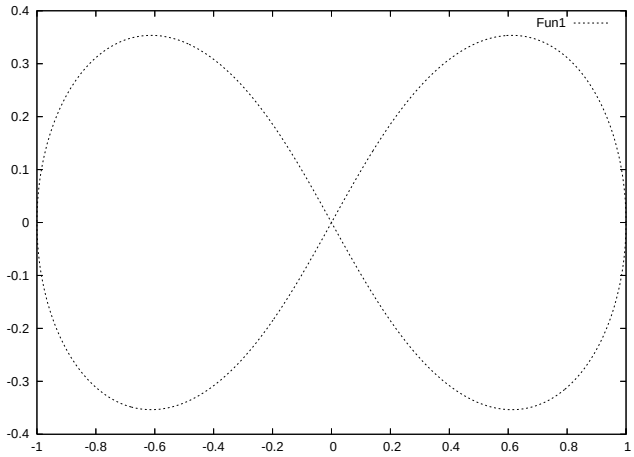
```
(%i33)  plot2d([parametric,cos(7*t),sin(4*t)],[t,0,2*%pi],[nticks,300]);
```




```
(%o33)
```

(3) ベルヌーイのレムニスケート $x = \dfrac{\cos(t)}{1+\sin^2(t)}$, $y = \dfrac{\sin(t)\cos(t)}{1+\sin^2(t)}$ のグラフを $0 \leqslant t \leqslant 2\pi$ の範囲で描画しなさい。

```
(%i34)  plot2d([parametric,cos(t)/(1+sin(t)^2),(sin(t)*cos(t))/(1+sin(t)^2)]
        ,[t,0,2*%pi],[nticks,1000]);
```

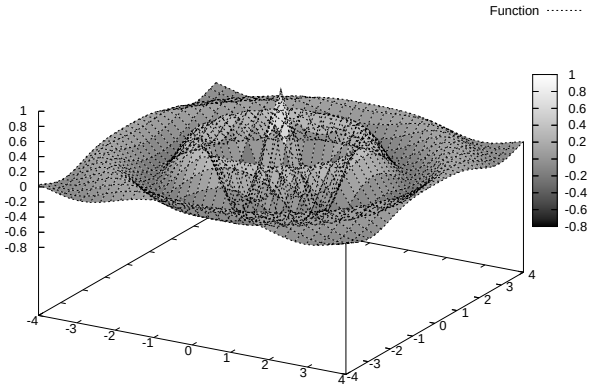


(%o34)

3. $z = f(x, y)$ のグラフ

(1) 関数 $z = \exp\left(-\dfrac{\sqrt{x^2+y^2}}{2}\right)\cos\left(\pi\sqrt{x^2+y^2}\right)$ のグラフを，$-4 \leqslant x, y \leqslant 4$ の範囲で描画しなさい。

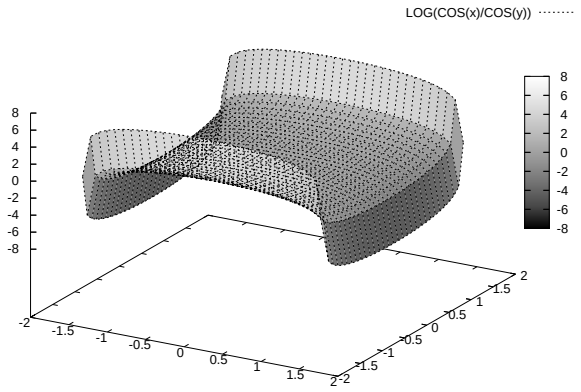
```
(%i35)  plot3d(exp(-sqrt(x^2+y^2)/2)*cos(%pi*sqrt(x^2+y^2)),[x,-4,4],[y,-4,4]);
```



(%o35)

(2) シャークの極小曲面 $z = \log\left(\dfrac{\cos(x)}{\cos(y)}\right)$ のグラフを $-\dfrac{\pi}{2} < x, y < \dfrac{\pi}{2}$ の範囲で描画しなさい。

```
(%i36)  plot3d(log(cos(x)/cos(y)),[x,-1.57,1.57],[y,-1.57,1.57]);
```



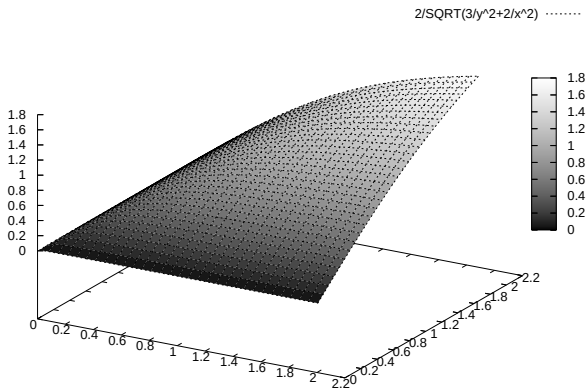

```
(%o36)
```

(3) 2 財に対する CES 型効用関数 (生産函数) $u(a,x,y,p,q,r) = a\left(px^{-r} + qy^{-r}\right)^{-\frac{1}{r}}$ を定義して，それを用いて $a = 2$, $p = 2$, $q = 3$, $r = 2$ のグラフを，$0 < x, y \leqslant 2$ の範囲で描画しなさい。

```
(%i37)  u(a,x,y,p,q,r):=a*(p*x^(-r)+q*y^(-r))^(-1/r);
```

$$u(a, x, y, p, q, r) := a\left(p\,x^{-r} + q\,y^{-r}\right)^{-\frac{1}{r}} \qquad \text{(\%o37)}$$

```
(%i38)  plot3d(u(2,x,y,2,3,2),[x,0.01,2],[y,0.01,2]);
```



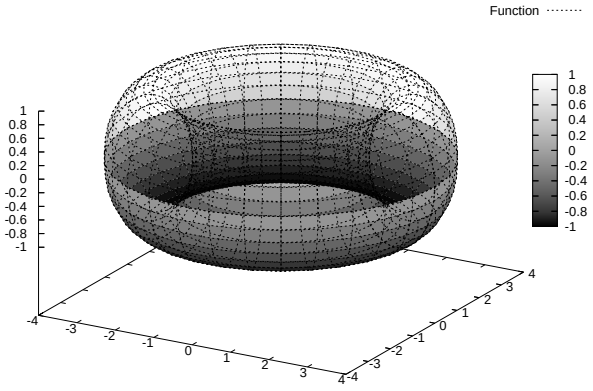

```
(%o38)
```

```
(%i39)  kill(u);

(%o39)                                DONE
```

4. $x = f(s,t)$, $y = g(s,t)$, $z = h(s,t)$ のグラフ

(1) トーラス $x = \cos(s)\,(3 + \cos(t))$, $y = \sin(s)\,(3 + \cos(t))$, $z = \sin(s)$ のグラフを $0 \leqslant s, t \leqslant 2\pi$ の範囲で描画しなさい。

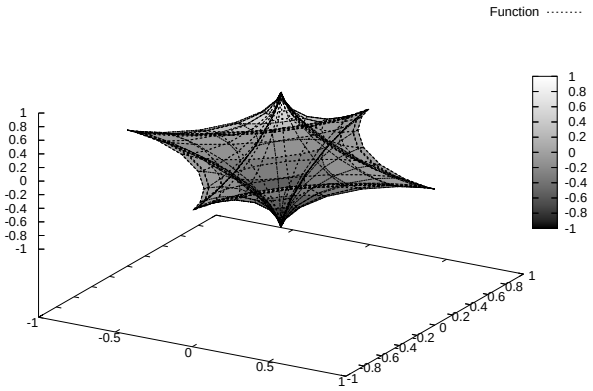
```
(%i40)  plot3d([cos(s)*(3+cos(t)),sin(s)*(3+cos(t)),sin(t)],[s,0,2*%pi],[t,0,2*%pi]);
```




```
(%o40)
```

(2) アステロイド的球面 $x = \cos^3(s)\cos^3(t)$, $y = \sin^3(s)\cos^3(t)$, $z = \sin^3(t)$ のグラフを $0 \leqslant s, t \leqslant 2\pi$ の範囲で描画しなさい。

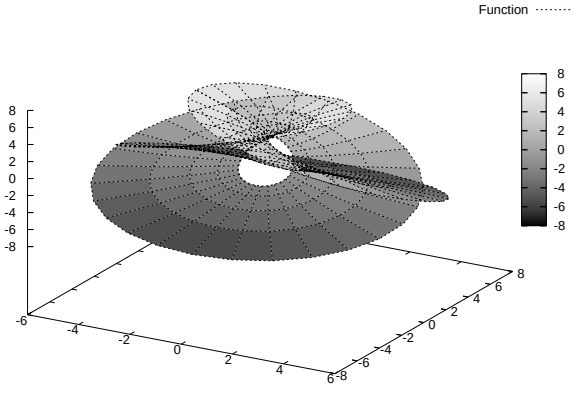
```
(%i41)  plot3d([cos(s)^3*cos(t)^3,sin(s)^3*cos(t)^3,sin(t)^3],[s,0,2*%pi],[t,0,2*%pi]);
```




```
(%o41)
```

(3) ヘンネベルグの極小曲面 $x = 2\sinh(s)\cos(t) - \frac{2}{3}\sinh(3s)\cos(3t)$, $y = 2\sinh(s)\sin(t) - \frac{2}{3}\sinh(3s)\sin(3t)$, $z = 2\cosh(2s)\cos(2t)$ のグラフを $0.3 \leqslant s \leqslant 0.9$, $0 \leqslant t \leqslant 2\pi$ の範囲で描画しなさい。

```
(%i42)  plot3d([2*sinh(s)*cos(t)-2/3*sinh(3*s)*cos(3*t),
        2*sinh(s)*sin(t)-2/3*sinh(3*s)*sin(3*t),2*cosh(2*s)*cos(2*t)]
        ,[s,0.3,0.9],[t,0,2*%pi],[grid,4,72]);
```

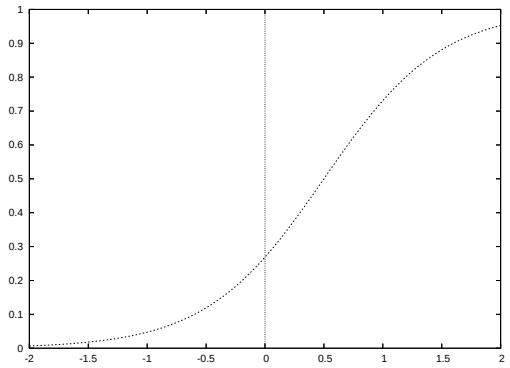



```
(%o42)
```

5. gnuplot との連動

(1) Maxima で $y = \dfrac{1}{1 + e^{-2x+1}}$ のグラフを，$-2 \leqslant x \leqslant 2$ の範囲で作成し，それを gnuplot に取り込みなさい。

```
(%i43)  plot2d(1/(1+exp(-2*x+1)),[x,-2,2],[plot_format,gnuplot]);

(%o43)


gnuplot> load 'maxout.gnuplot' with lines
```



(2)　gnuplot を用いて，$z=\sqrt{x^2+y^2}$ と $z=x+1$ のグラフを $-1 \leqslant x, y \leqslant 1$ の範囲で一緒に描画しなさい。このとき，交わりである放物線が分かるような視点を探しなさい。

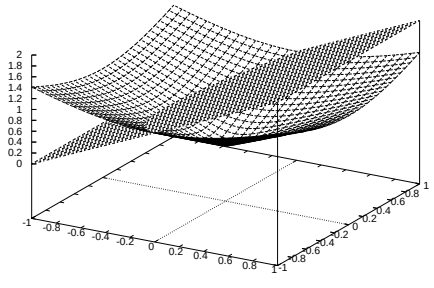
```
(%i44)  plot3d(sqrt(x^2+y^2),[x,-1,1],[y,-1,1],[plot_format,gnuplot]);

(%o44)
```

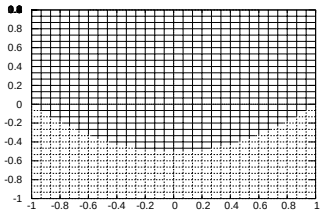
```
% tail +4 maxout.gnuplot > maxout1
```
(1)

```
(%i45)  plot3d(x+1,[x,-1,1],[y,-1,1],[plot_format,gnuplot]);

(%o45)
```

```
% tail +4 maxout.gnuplot > maxout2
```
(2)

```
gnuplot> set hidden3d

gnuplot> splot 'maxout1' with lines, 'maxout2' with lines
```





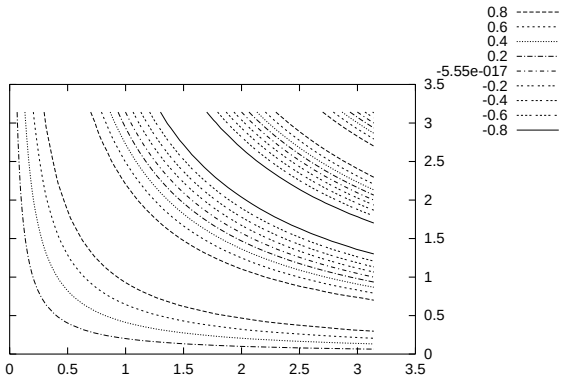
(1) これは Linux や MacOSX でのやり方です。Microsoft Windows ならエディタで行頭 4 行分を消去して，別名で保存してください。

(2) これは Linux や MacOSX でのやり方です。Microsoft Windows ならエディタで行頭 4 行分を消去して，別名で保存してください。

(3) gnuplot を用いて，$z = \sin(xy)$ の等高線のグラフを $0 \leqslant x, y \leqslant \pi$ の範囲で描画しなさい。このとき，高さレベルを 9 段階用意しなさい。

```
(%i46)  plot3d(sin(x*y),[x,0,%pi],[y,0,%pi],[plot_format,gnuplot]);
```

```
% tail +4 maxout.gnuplot > contour.gnuplot

gnuplot> set contour

gnuplot> set cntrparam levels auto 9

gnuplot> unset surface

gnuplot> set view 0,0,1

gnuplot> splot 'contour.gnuplot' with lines
```



(4) 上で作成したグラフを，TEX で取り込める eps 形式で出力しなさい。また，gif 形式で出力しなさい。

```
gnuplot> set output 'foo.eps'

gnuplot> set terminal postscript

gnuplot> splot 'contour.gnuplot' with lines

gnuplot> set output 'foo.gif'

gnuplot> set terminal gif

gnuplot> splot 'contour.gnuplot' with lines

gnuplot> set output

gnuplot> set terminal x11
```

D.3 数式の操作

1. 多項式の整理

(1) 多項式 $(-a+b+c)^3(a-b+c)^3(a+b-c)^3$ を展開しなさい。

```
(%i47)  expand((-a+b+c)^3*(a-b+c)^3*(a+b-c)^3);

          9        8        8           7      3  6         2  6
(%o47) - c  + 3 b c  + 3 a c  - 12 a b c  - 8 b  c  + 12 a b  c
       2    6      3  6      4  5         3  5       2  2  5
 + 12 a  b c  - 8 a  c  + 6 b  c  + 12 a b  c  - 36 a  b  c
       3    5      4  5      5  4         4  4       2  3  4
 + 12 a  b c  + 6 a  c  + 6 b  c  - 30 a b  c  + 24 a  b  c
       3  2  4       4    4      5  4      6  3         5  3
 + 24 a  b  c  - 30 a  b c  + 6 a  c  - 8 b  c  + 12 a b  c
       2  4  3       3  3  3       4  2  3       5    3      6  3
 + 24 a  b  c  - 56 a  b  c  + 24 a  b  c  + 12 a  b c  - 8 a  c
         6  2       2  5  2       3  4  2       4  3  2       5  2  2
 + 12 a b  c  - 36 a  b  c  + 24 a  b  c  + 24 a  b  c  - 36 a  b  c
       6    2      8           7         2  6         3  5
 + 12 a  b c  + 3 b  c - 12 a b  c + 12 a  b  c + 12 a  b  c
       4  4         5  3         6  2         7          8      9
 - 30 a  b  c + 12 a  b  c + 12 a  b  c - 12 a  b c + 3 a  c - b
        8      3  6      4  5      5  4      6  3      8      9
 + 3 a b  - 8 a  b  + 6 a  b  + 6 a  b  - 8 a  b  + 3 a  b - a
```

(2) 多項式 $(x+1)^{36}(x-2)^{64}$ の x^{50} の係数を求めなさい。

```
(%i48)  ratcoef((x+1)^36*(x-2)^64,x^50);

(%o48)                          - 56567652670376600977539072
```

(3) 多項式 $-bc^4-ac^4+b^2c^3+abc^3+a^2c^3+b^3c^2+a^3c^2-b^4c+ab^3c+a^3bc-a^4c-ab^4+a^2b^3+a^3b^2-a^4b$ を因数分解しなさい。

```
(%i49)  factor(-b*c^4-a*c^4+b^2*c^3+a*b*c^3+a^2*c^3+b^3*c^2+a^3*c^2-b^4*c+a*b^3*c
        +a^3*b*c-a^4*c-a*b^4+a^2*b^3+a^3*b^2-a^4*b);

(%o49)          - (c - b - a) (c - b + a) (c + b - a) (b c + a c + a b)
```

(4) 193480610806663641976612700346380 81 を素因数分解しなさい。

```
(%i50)  factor(193480610806663641976612700346380 81);

                                          2        3
(%o50)                              7199911  7199921
```

2. 有理式の整理

(1) 有理式 $x-1+\dfrac{1}{3(x+1)-\dfrac{x-2}{3(x^2-x+1)}}$ を通分しなさい。

```
(%i51)  ratsimp(x-1+1/(3*(x+1)-(x-2)/(3*(x^2-x+1))));
```

$$\text{(\%o51)}\quad \frac{9x^4-9x^3+2x^2+9x-8}{9x^3-x+11}$$

(2) 有理式 $\dfrac{1+\dfrac{1}{2+\dfrac{i}{3+\dfrac{i}{x}}}}{1+\dfrac{1}{3+\dfrac{i}{2+\dfrac{i}{1+\dfrac{i}{x}}}}}$ の分子，分母をそれぞれ求めなさい。

```
(%i52)  q:ratsimp((1/(1+%i/(2+%i/(3+%i/x))))/(1/(3+%i/(2+%i/(1+%i/x)))));
```

$$\text{(\%o52)}\quad \frac{(30\,\%i+32)\,x^2+(47\,\%i-20)\,x-2\,\%i-12}{(14\,\%i+8)\,x^2+(15\,\%i-12)\,x-2\,\%i-4}$$

```
(%i53)  num(q);
```

$$\text{(\%o53)}\quad (30\,\%i+32)\,x^2+(47\,\%i-20)\,x-2\,\%i-12$$

```
(%i54)  denom(q);
```

$$\text{(\%o54)}\quad (14\,\%i+8)\,x^2+(15\,\%i-12)\,x-2\,\%i-4$$

(3)　有理式 $\dfrac{1}{x^9+1}$ を部分分数展開しなさい。

```
(%i55)  partfrac(1/(x^9+1),x);
```

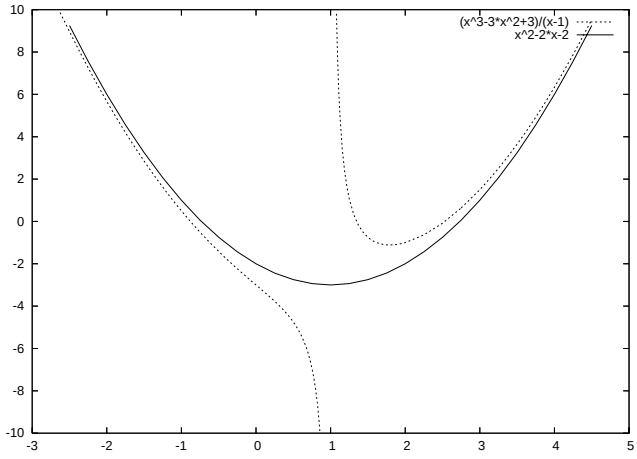
```
                                  3
                               x  - 2            x - 2             1
(%o55)          - --------------- - ------------- + ---------
                        6     3          2          9 (x + 1)
                  3 (x  - x  + 1)   9 (x  - x + 1)
```

(4)　有理式 $\dfrac{x^3-3x^2+3}{x-1}$ の無限大における漸近線を求め，それを図示しなさい。

```
(%i56)  partfrac((x^3-3*x^2+3)/(x-1),x);
```

```
                                    2          1
(%o56)                             x  - 2 x + ------ - 2
                                              x - 1
```

```
(%i57)  plot2d([(x^3-3*x^2+3)/(x-1),x^2-2*x-2],[x,-5,5],[y,-10,10]);
```




```
(%o57)
```

3. 方程式の厳密解

(1)　方程式 $x^3+px+q=0$ を x について解きなさい。

```
(%i58)  solve(x^3+p*x+q=0,x);
```

```
                                                2      3
                    sqrt(3) %i   1   sqrt(27 q  + 4 p )   q 1/3
(%o58)  [x = (- ---------- - -) (------------------ - -)
                         2        2       6 sqrt(3)        2
                          sqrt(3) %i   1
                          (---------- - -) p
                               2        2
                     - ----------------------------,
                                    2      3
                               sqrt(27 q  + 4 p )   q 1/3
                           3 (------------------ - -)
                                  6 sqrt(3)        2
                                                   2      3
                           sqrt(3) %i   1   sqrt(27 q  + 4 p )   q 1/3
                     x = (---------- - -) (------------------ - -)
                                2        2       6 sqrt(3)        2
                                    sqrt(3) %i   1
                                   (- ---------- - -) p
                                         2        2
                     - ----------------------------,
                                    2      3
                               sqrt(27 q  + 4 p )   q 1/3
                           3 (------------------ - -)
                                  6 sqrt(3)        2
                                     2      3
                           sqrt(27 q  + 4 p )   q 1/3                 p
                     x = (------------------ - -)    - ----------------------------]
                                  6 sqrt(3)        2                   2      3
                                                                  sqrt(27 q  + 4 p )   q 1/3
                                                                3 (------------------ - -)
                                                                       6 sqrt(3)        2
```

(2) 連立方程式

$$\begin{cases} x^3+y^3+z^3=8 \\ x^2+y^2+z^2=27 \\ x+y+z=1 \end{cases}$$

を x, y, z について解きなさい。

```
(%i59)  solve([x^3+y^3+z^3=8,x^2+y^2+z^2=27,x+y+z=1],[x,y,z]);


(%o59) [[x = 3.722506393861893, y = 0.81094919259551, z = - 3.533455545371219],
```

```
[x = 3.722506393861893, y = - 3.533455545371219, z = 0.81094919259551],

[x = - 3.533455545371219, y = 0.81094919259551, z = 3.722506393861893],

[x = - 3.533455545371219, y = 3.722506393861893, z = 0.81094919259551],

[x = 0.81094919259551, y = - 3.533455545371219, z = 3.722506393861893],

[x = 0.81094919259551, y = 3.722506393861893, z = - 3.533455545371219]]
```

(3)　連立方程式

$$\begin{cases} xy + yz + zx = 1 \\ x + y + z = 1 \end{cases}$$

を x, y, z について解きなさい。

変数が 3 つなのに方程式が 2 つしかありませんので，不定方程式ではないかと予想されます。

```
(%i60)  solve([x*y+y*z+z*x=1,x+y+z=1],[x,y,z]);


  Maxima encountered a Lisp error:

   Error in MACSYMA-TOP-LEVEL [or a callee]: 0 is not of type LIST.

  Automatically continuing.
  To reenable the Lisp debugger set *debugger-hook* to nil.
```

となり，不定方程式であることが分かります。そこで，変数の数を方程式の数にまで落としてみましょう。

```
(%i61)  solve([x*y+y*z+z*x=1,x+y+z=1],[x,y]);

                          2
                sqrt(- 3 z  + 2 z - 3) + z - 1
(%o61) [[x = - ------------------------------,
                             2

                          2                                 2
                sqrt(- 3 z  + 2 z - 3) - z + 1    sqrt(- 3 z  + 2 z - 3) - z + 1
          y = ------------------------------], [x = ------------------------------,
                             2                                 2
```

```
                                  2
                       sqrt(- 3 z  + 2 z - 3) + z - 1
                 y = - ------------------------------]]
                                    2
```

となって，x, y を z の式で記述可能であることが分かります。

(4) 連立方程式

$$\begin{cases} x^3 + y^3 = 5 \\ x^2 + y^2 = 3 \\ x + y = 1 \end{cases}$$

を x, y, z について解きなさい。

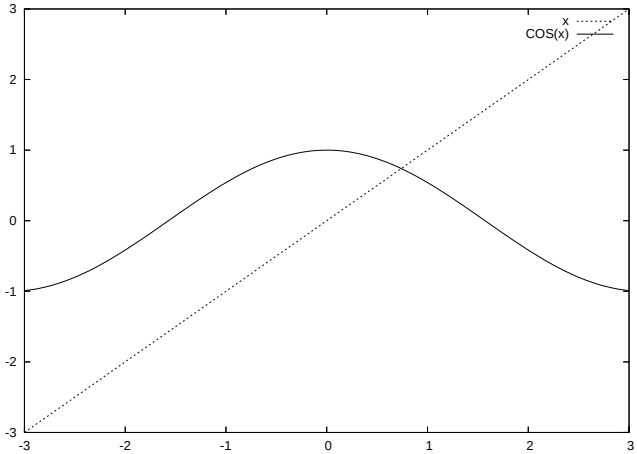
```
(%i62)  solve([x^3+y^3=5,x^2+y^2=3,x+y=1],[x,y,z]);

(%o62)                                    []
```

となって，不能方程式であることが分かります。

4. 実数解の数値近似

(1) 方程式 $x = \cos(x)$ の実数解を数値近似しなさい。

```
(%i63)  plot2d([x,cos(x)],[x,-3,3]);
```




```
(%o63)
```

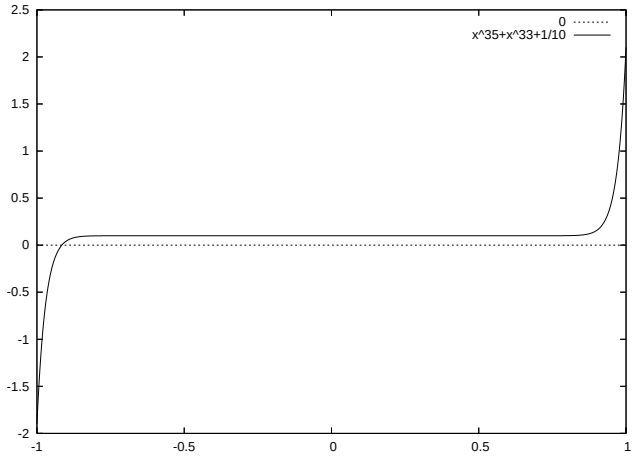
グラフの交点にマウスカーソルを重ねて読み取れる数値より，およそ $x = 0.734820$ くらいで交点を持つようですから，それを初期値としてニュートン近似を行います。

```
(%i64)  load(newton);
```

```
(%o64)     C:/PROGRA~1/MAXIMA~1.2/share/maxima/5.9.2/share/numeric/newton.mac

(%i65)  newton(x-cos(x),0.734820);

  Warning:  Float to bigfloat conversion of 0.73482000000000003

(%o65)                                7.390851332187552B-1
```

(2)　方程式 $x^{35}+x^{33}+\dfrac{1}{10}=0$ の実数解を数値近似しなさい。

```
(%i66)  plot2d([0,x^35+x^33+1/10],[x,-1,1]);
```



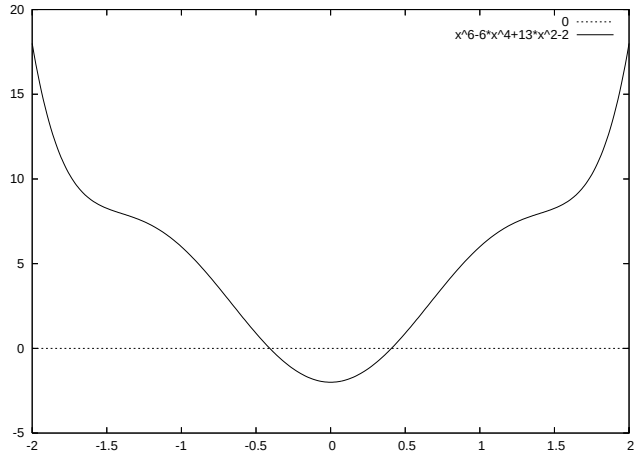

```
(%o66)
```

グラフの交点にマウスカーソルを重ねて読み取れる数値より，およそ $x=-0.915347$ くらいで交点を持つようですから，それを初期値としてニュートン近似を行います。

```
(%i67)  newton(x^35+x^33+1/10,-0.915347);

  Warning:  Float to bigfloat conversion of -0.91534700000000002

(%o67)                              - 9.155556890178944B-1
```

(3) 方程式 $x^6 - 6x^4 + 13x^2 - 2 = 0$ の実数解を数値近似しなさい。

```
(%i68)  plot2d([0,x^6-6*x^4+13*x^2-2],[x,-2,2]);
```




```
(%o68)
```

グラフの交点にマウスカーソルを重ねて読み取れる数値より，およそ $x = 0.407034$ と $x = -0.409536$ くらいで交点を持つよう[3]ですから，それを初期値としてニュートン近似を行います[4]。

```
(%i69)  newton(x^6-6*x^4+13*x^2-2,0.394857);

  Warning:  Float to bigfloat conversion of 0.39485700000000001

(%o69)                              4.077364863224771B-1

(%i70)  newton(x^6-6*x^4+13*x^2-2,-0.394857);

  Warning:  Float to bigfloat conversion of -0.39485700000000001

(%o70)                             - 4.077364863224771B-1
```

[3]関数 $f(x) = x^6 - 6x^4 + 13x^2 - 2$ は，**偶関数**と呼ばれる性質 $f(x) = f(-x)$ を満たしますから，解も対称性を持ちます。ただ，アプリケーション操作の限界から，多少のずれが発生しています。

[4]$t = x^2$ と置けば，$f(x) = t^3 - 6t^2 + 13t - 2$ となりますので，3 次方程式ですから厳密に解けることになります。ですから，普通に `solve` 函数と `float` 函数に引き渡しますと，$x \fallingdotseq -0.4077364863225$, $x \fallingdotseq 0.4077364863225$ であることが分かりますが，ここではあえてニュートン近似で求めてみます。

1. 極限値

(1)　極限値 $\lim_{x\to 0}\dfrac{\cos(2x)-1}{\cosh(3x)-1}$ を求めなさい。

```
(%i71)  limit((cos(2*x)-1)/(cosh(3*x)-1),x,0);

                                        4
(%o71)                                - --
                                        9
```

(2)　極限値 $\lim_{x\to 0+0}\dfrac{\cos(x)-1}{x\,|x|}$ を求めなさい。

```
(%i72)  limit((cos(x)-1)/(x*abs(x)),x,0,plus);

                                        1
(%o72)                                - --
                                        2
```

(3)　極限値 $\lim_{n\to\infty}\left(1-\dfrac{2}{3n}\right)^{4n}$ を求めなさい。

```
(%i73)  limit((1-2/(3*n))^(4*n),n,inf);

                                      - 8/3
(%o73)                               %E
```

(4)　関数 $y=\dfrac{x^4-x^2+x+5}{x^2-1}$ は，x が正負の無限大において漸近線 $y=x^2-x+1$ を持つことを確認しなさい。

```
(%i74)  limit((x^4-x^3+x+5)/(x^2-1)-(x^2-x+1),x,inf);

(%o74)                                  0


(%i75)  limit((x^4-x^3+x+5)/(x^2-1)-(x^2-x+1),x,minf);

(%o75)                                  0
```

2. 関数の微分

(1)　関数 $f(x)=\cos\left(\cos\left(\cos(x^3)\right)\right)$ の導関数を求めなさい。また，3 階導関数を求めなさい。

```
(%i76)  diff(cos(cos(cos(x^3))),x);

                          2      3           3                 3
(%o76)              - 3 x  sin(x ) sin(cos(x )) sin(cos(cos(x )))


(%i77)  diff(cos(cos(cos(x^3))),x,3);
```

```
            6    3  3     3      3                3
(%o77) 27 x  sin (x ) sin (cos(x )) sin(cos(cos(x )))
             6    3  3           3                3
       + 27 x  sin (x ) sin(cos(x )) sin(cos(cos(x )))
             6      3           3                3
       + 27 x  sin(x ) sin(cos(x )) sin(cos(cos(x )))
              3           3                3
       - 6 sin(x ) sin(cos(x )) sin(cos(cos(x )))
             3      3           3                3
       - 54 x  cos(x ) sin(cos(x )) sin(cos(cos(x )))
             3    2  3           3                3
       + 54 x  sin (x ) cos(cos(x )) sin(cos(cos(x )))
             6      3       3           3                3
       + 81 x  cos(x ) sin(x ) cos(cos(x )) sin(cos(cos(x )))
             3    2  3     2      3                3
       - 54 x  sin (x ) sin (cos(x )) cos(cos(cos(x )))
             6      3       3     2      3                3
       - 81 x  cos(x ) sin(x ) sin (cos(x )) cos(cos(cos(x )))
             6    3  3           3            3                3
       + 81 x  sin (x ) cos(cos(x )) sin(cos(x )) cos(cos(cos(x )))
```

(2) 関数 $f(x)$ の極大値や極小値，いわゆるグラフの山の頂きや谷の底を与える x の値は，$f'(x) = 0$ を満たします。そして，その x の値を $f''(x)$ に代入して，負であれば極大値 (山の頂き)，正であれば極小値 (谷の底) を与えることになります。これを用いて，$f(x) = x^x$, $(x > 0)$ の極小値を求めなさい。

```
(%i78)  solve(diff(x^x,x)=0,x);
                                              - 1    x
(%o78)                                [x = %e    , x  = 0]

(%i79)  at(diff(x^x,x,2),x=float(1/%e));

(%o79)                                 1.881596387531646
```

(3) 関数 $f(x,y) = \dfrac{x^2 y}{x^4 + y^2}$ の x による偏導関数 $f_x(x,y)$ および，y による偏導関数 $f_y(x,y)$ を求めなさい。また，$f(x,y)$ の全導関数を求めなさい。

```
(%i80)  diff((x^2*y)/(x^4+y^2),x);

                                          5
                                 2 x y      4 x  y
(%o80)                           ------- - ----------
                                 2    4     2    4 2
                                y  + x     (y  + x )
```

```
(%i81)  diff((x^2*y)/(x^4+y^2),y);

                            2          2  2
                           x        2 x  y
(%o81)                    ------- - ----------
                           2    4     2    4 2
                          y  + x     (y  + x )
```

```
(%i82)  diff((x^2*y)/(x^4+y^2));

            2          2  2                              5
           x        2 x  y                 2 x y      4 x  y
(%o82) (------- - ----------) del(y) + (------- - ----------) del(x)
          2    4     2    4 2             2    4     2    4 2
         y  + x     (y  + x )            y  + x     (y  + x )
```

(4)　関数 $\log(\cos(\pi x))$ の，$x=1$ を展開の中心とするテイラー展開を 6 次まで表示しなさい。

```
(%i83)  taylor(log(cos(%pi*x)),x,1,6);

                          2        2      4        4      6        6
                       %pi  (x - 1)    %pi  (x - 1)    %pi  (x - 1)
(%o83)/T/ log(- 1) - ------------ - ------------ - ------------
                          2              12              45

                                                              + . . .
```

3. 級数和と級数積

(1)　級数和 $\log(10000) - \displaystyle\sum_{k=1}^{10000} \frac{1}{k}$ を数値近似しなさい。

```
(%i84)  float(log(10000)-sum(1/k,k,1,10000));

(%o84)                              - 0.5772656640682
```

(2)　関数 $\arctan(x)$ の，$x=1$ を展開の中心とする 10 次のテイラー多項式を求めなさい。

```
(%i85)  sum(at(diff(atan(x),x,k),x=1)/k!*(x-1)^k,k,0,10);
```

```
                  10          9          7          6          5
           (x - 1)     (x - 1)    (x - 1)    (x - 1)    (x - 1)
(%o85)  - --------- + -------- - -------- + -------- - --------
             320         288        112         48         40

                                                3          2
                                          (x - 1)    (x - 1)    x - 1   %pi
                                        + -------- - -------- + ----- + ---
                                             12         4         2      4
```

(3) 級数和 $\displaystyle\sum_{k=1}^{n} k^2 e^k$ を求めなさい。

```
(%i86)  load(nusum);

(%o86)     C:/PROGRA~1/MAXIMA~1.2/share/maxima/5.9.2/share/algebra/nusum.mac


(%i87)  nusum(k^2*exp(k),k,1,n);

          2  2         2    2                              n + 1
       (%e  n  - 2 %e n  + n  - 2 %e n + 2 n + %e + 1) %e
(%o87) ------------------------------------------------------
                                    3
                           (%e - 1)
                                                    %e (%e + 1)
                                                  - -----------
                                                             3
                                                    (%e - 1)
```

(4) 級数積 $\displaystyle\prod_{k=1}^{100}\left(1-\frac{\sin^2(k)}{k}\right)$ を数値近似しなさい。

```
(%i88)  float(product(1-sin(k)^2/k,k,1,100));

(%o88)                          0.0285262359834
```

(5) 級数積 $\displaystyle\prod_{k=1}^{10000}\left(1-\frac{\sin^2(k)}{k}\right)$ を数値近似しなさい。

```
(%i89)  float(product(1-sin(k)^2/k,k,1,10000));
```

このままでは，ちょっと終わってくれません。そこで，一度停止させましょう。

```
Maxima encountered a Lisp error:

 Console interrupt.

Automatically continuing.
To reenable the Lisp debugger set *debugger-hook* to nil.
```

仕方がないので，誤差が大きくなることを覚悟の上で，100 個ずつにデータを区切ることにします。

```
(%i90)  lst:makelist(float(product(1-sin(k)^2/k,k,100*n+1,100*n+100)),n,0,99);

(%o90) [0.0285262359834, 0.70651698296465, 0.81701560512074,

        0.8668398694711, 0.89487347730142, 0.91274778781261,

          (途中省略)

        0.99489816322068, 0.99496707323302, 0.99500380663043]


(%i91)  product(lst[k],k,1,100);

(%o91)                              0.00285502480784
```

4. 関数の積分

(1)　不定積分 $\displaystyle\int \frac{1}{x^5+1}\,dx$ を求めなさい。

```
(%i92)  integrate(1/(x^5+1),x);

                                2
       (sqrt(5) - 3) log(2 x  + (sqrt(5) - 1) x + 2)
(%o92) ---------------------------------------------
                     10 - 10 sqrt(5)

                                2
         (sqrt(5) + 3) log(2 x  + (- sqrt(5) - 1) x + 2)
       - -----------------------------------------------
                10 sqrt(5) + 10

                           4 x + sqrt(5) - 1
```

```
                  (sqrt(5) + 1) atan(--------------------)
                                    sqrt(2 sqrt(5) + 10)
                + ----------------------------------------
                         sqrt(5) sqrt(2 sqrt(5) + 10)

                                     4 x - sqrt(5) - 1
                  (sqrt(5) - 1) atan(--------------------)
                                    sqrt(10 - 2 sqrt(5))    log(x + 1)
                + ---------------------------------------- + ----------
                         sqrt(5) sqrt(10 - 2 sqrt(5))            5
```

(2) 積分方程式 $\int_{-\infty}^{\infty} Ce^{-\frac{x^2}{2}}\,dx = 1$ を満たす C を求めなさい。

```
(%i93)  solve(integrate(C*exp(-x^2/2),x,minf,inf)=1,C);

                                          1
(%o93)                          [C = ------------------]
                                     sqrt(2) sqrt(%pi)
```

(3) グラフ $y = f(x)$ に沿った曲線 $(a, f(a)) \sim (b, f(b))$ の長さは $\int_a^b \sqrt{1 + (f'(x))^2}\,dx$ で与えられます。これを用いて，関数 $y = \cosh(x)$ に沿った曲線 $(\log(2 - \sqrt{3}), 2) \sim (\log(2 + \sqrt{3}), 2)$ の長さを求めなさい。

```
(%i94)  integrate(sqrt(1+(diff(cosh(x),x))^2),x,log(2-sqrt(3)),log(2+sqrt(3)));

(%o94)                                 2 sqrt(3)
```

(4) 関数 $y = f(x),\ (a \leqslant x \leqslant b, f(x) > 0)$ を x 軸を中心に回してできた，帯状の回転体の面積は，$2\pi \int_a^b f(x)\sqrt{1 + (f'(x))^2}\,dx$ で与えられます。
これを用いて，楕円 $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1,\ (a > b)$ を x 軸を中心に回してできた回転体の表面積を求めなさい。また，y 軸を中心に回してできた回転体の表面積を求めなさい。

```
(%i95)  solve(x^2/a^2+y^2/b^2=1,y);

                                 2    2               2    2
                          b sqrt(a  - x )       b sqrt(a  - x )
(%o95)           [y = - ---------------, y = ---------------]
                                a                     a


(%i96)  f(x):=b*sqrt(a^2-x^2)/a;
```

```
                                          2    2
                                   b sqrt(a  - x )
(%o96)                  f(x) := ---------------
                                        a
```

```
(%i97)  2*%pi*integrate(f(x)*sqrt(1+diff(f(x),x)^2),x,-a,a);

Is  a  positive or negative?
positive;
                     a
                    /                            2  2
                    [          2    2          b  x
            2 %pi b I    sqrt(a  - x ) sqrt(------------ + 1) dx
                    ]                         2   2    2
                    /                        a  (a  - x )
                     - a
(%o97)      ------------------------------------------------------
                                   a
```

何やら積分計算ができていません。ですが，よく眺めますと sqrt 関数が上手く処理できていないことが分かります。ですから，ここだけ人力で展開してやります。

```
(%i98)  2*%pi*integrate(b/a*sqrt(b^2*x^2/a^2+a^2-x^2),x,-a,a);

Is  a  positive or negative?
positive;
Is  (b - a) (b + a)  positive or negative?
negative;
Is  b  positive or negative?
positive;
```

```
                                                 2    2
                    3       2    2       sqrt(a  - b )       3    3
          4 %pi b (a  sqrt(a  - b ) asin(-------------) - a b  + a  b)
                                                 a
(%o98)  - ------------------------------------------------------------
                                      2      2
                                 a (2 b  - 2 a )
```

```
(%i99)  solve(x^2/a^2+y^2/b^2=1,x);
```

```
                                2    2               2    2
                          a sqrt(b  - y )      a sqrt(b  - y )
(%o99)           [x = - ---------------, x = ---------------]
                                 b                    b
```

```
(%i100)  g(y):=a*sqrt(b^2-y^2)/b;
```

```
                                          2    2
                                   a sqrt(b  - y )
(%o100)                  g(y) := ---------------
                                          b
```

```
(%i101)  g(y)*sqrt(1+diff(g(y),y)^2);
```

```
                                          2  2
                          2    2         a  y
                  a sqrt(b  - y ) sqrt(------------ + 1)
                                        2   2    2
                                       b  (b  - y )
(%o101)           ---------------------------------------
                                    b
```

```
(%i102)  2*%pi*integrate(a/b*sqrt(a^2*y^2/b^2+b^2-y^2),y,-b,b);

Is  b  positive or negative?
positive;
Is  (b - a) (b + a)  positive or negative?
```

```
negative;
Is  a  positive or negative?
positive;
                                                2    2
                    3       2    2         sqrt(a  - b )       3    3
          4 %pi a (b  sqrt(a  - b ) asinh(-------------) - a b  + a  b)
                                                 b
(%o102) - ------------------------------------------------------------
                                      2      2
                                 b (2 b  - 2 a )
```

5. 微分方程式

(1)　微分方程式 $v'(t) = 9.8 - 4.9v(t)$ を，初期条件 $v(0) = 0$ について解きなさい。

```
(%i103)  atvalue(v(t),t=0,0);

(%o103)                                  0


(%i104)  desolve(diff(v(t),t)=9.8-4.9*v(t),v(t));


RAT replaced -9.8 by -49//5 = -9.8
RAT replaced 4.9 by 49//10 = 4.9
                                                 49 t
                                               - ----
                                                  10
(%o104)                    v(t) = 2 - 2 %e
```

(2)　微分方程式 $x'''(t) - 2x''(t) + 3x'(t) - 2x(t) = \sin(x)$ を，初期条件 $x(0) = 1$, $x'(0) = 0$, $x''(0) = -1$ について解きなさい。

```
(%i105)  atvalue(x(t),t=0,1);

(%o105)                                  1


(%i106)  atvalue(diff(x(t),t),t=0,0);

(%o106)                                  0


(%i107)  atvalue(diff(x(t),t,2),t=0,-1);
```

```
(%o107)                                  - 1
```

```
(%i108)  desolve(diff(x(t),t,3)-2*diff(x(t),t,2)+3*diff(x(t),t)-2*x(t)=sin(t),x(t));
```

$$(\%o108)\quad x(t) = \%e^{t/2}\left(\frac{3\cos\left(\frac{\sqrt{7}\,t}{2}\right)}{4} - \frac{9\sin\left(\frac{\sqrt{7}\,t}{2}\right)}{4\sqrt{7}}\right) - \frac{\cos(t)}{2} + \frac{3\,\%e^{t}}{4}$$

(3) 微分方程式 $p'(x) = -xp(x)$ を，初期条件 $p(0) = \dfrac{1}{\sqrt{2\pi}}$ について解きなさい。

```
(%i109)  atvalue(p(x),x=0,1/sqrt(2*%pi));
```

$$(\%o109)\quad \frac{1}{2\,\%PI}$$

```
(%i110)  desolve(diff(p(x),x)=-x*p(x),p(x));
```

$$(\%o110)\quad p(x) = \mathrm{ilt}\left(\frac{\sqrt{2}\,\sqrt{\%pi}\left(\frac{d}{d\,lvar}\left(\mathrm{laplace}(p(x), x, lvar)\right)\right) + 1}{\sqrt{2}\,\sqrt{\%pi}\;lvar},\ lvar,\ x\right)$$

どうも desolve 函数では解けなかったようです。ですから ode2 函数に掛けてみます。

```
(%i111)  ode2(diff(p(x),x)=-x*p(x),p(x),x);
```

$$(\%o111)\quad p(x) = \%c\ \%e^{-\frac{x^2}{2}}$$

こちらでは解けました。

(4)　連立微分方程式

$$\begin{cases} x'(t) = 4x(t) + 2y(t) + \sin(t) \\ y'(t) = 3x(t) + 3y(t) - \cos(t) \end{cases}$$

を，初期条件 $x(0) = 1$, $y(0) = -1$ について解きなさい。

```
(%i112)  atvalue(x(t),t=0,1);

(%o112)                                  1


(%i113)  atvalue(y(t),t=0,-1);

(%o113)                                 - 1


(%i114)  desolve([diff(x(t),t)=4*x(t)+2*y(t)+sin(t),
         diff(y(t),t)=3*x(t)+3*y(t)-cos(t)],[x(t),y(t)]);

                                                6 t       t
                    4 sin(t)   13 cos(t)   28 %e      6 %e
(%o114) [x(t) = - -------- - --------- + -------- + -----,
                      37         37         185        5

                                                    6 t       t
                           4 sin(t)   24 cos(t)   28 %e      9 %e
                  y(t) = - -------- + --------- + -------- - -----]
                              37         37         185        5
```

D.4　リストの扱い

データのリスト

(1)　リスト $[1-t, 1+t, 1-t^2, 1+t^2]$ を変数 p に定義しなさい。また，その第 3 項を抽出しなさい。

```
(%i115)  p:[1-t,1+t,1-t^2,1+t^2];

                                              2   2
(%o115)                  [1 - t, t + 1, 1 - t , t  + 1]


(%i116)  p[3];

                                           2
(%o116)                                1 - t
```

(2) リスト p から第 3 項を削除しなさい。さらに，p の末尾に $1+t^4$ という項を追加しなさい。

```
(%i117)  p:delete(p[3],p,1);
                                                 2
(%o117)                            [1 - t, t + 1, t  + 1]


(%i118)  p:endcons(1+t^4,p);
                                               2       4
(%o118)                        [1 - t, t + 1, t  + 1, t  + 1]
```

(3) リスト $[1, 1+t, 1+t+t^2, 1+t+t^2+t^3]$ を変数 q に定義しなさい。そして，リスト p とリスト q を連結しなさい。

```
(%i119)  q:[1,1+t,1+t+t^2,1+t+t^2+t^3];
                                      2           3    2
(%o119)                  [1, t + 1, t  + t + 1, t  + t  + t + 1]


(%i120)  append(p,q);
                           2       4                 2           3    2
(%o120)   [1 - t, t + 1, t  + 1, t  + 1, 1, t + 1, t  + t + 1, t  + t  + t + 1]
```

(4) makelist を用いて，掛け算の九々の表を作りなさい。そして，それを行列に変換して，正方形状に表示しなさい。

```
(%i121)  p:makelist(makelist(x*y,y,1,9),x,1,9);


(%o121) [[1, 2, 3, 4, 5, 6, 7, 8, 9], [2, 4, 6, 8, 10, 12, 14, 16, 18],
         [3, 6, 9, 12, 15, 18, 21, 24, 27], [4, 8, 12, 16, 20, 24, 28, 32, 36],
         [5, 10, 15, 20, 25, 30, 35, 40, 45], [6, 12, 18, 24, 30, 36, 42, 48, 54],
         [7, 14, 21, 28, 35, 42, 49, 56, 63], [8, 16, 24, 32, 40, 48, 56, 64, 72],
         [9, 18, 27, 36, 45, 54, 63, 72, 81]]



(%i122)  apply(matrix,p);

                    [ 1  2   3   4   5   6   7   8   9  ]
                    [                                  ]
                    [ 2  4   6   8   10  12  14  16  18 ]
                    [                                  ]
                    [ 3  6   9   12  15  18  21  24  27 ]
```

```
                    [                                    ]
                    [ 4  8   12  16  20  24  28  32  36 ]
                    [                                    ]
(%o122)             [ 5  10  15  20  25  30  35  40  45 ]
                    [                                    ]
                    [ 6  12  18  24  30  36  42  48  54 ]
                    [                                    ]
                    [ 7  14  21  28  35  42  49  56  63 ]
                    [                                    ]
                    [ 8  16  24  32  40  48  56  64  72 ]
                    [                                    ]
                    [ 9  18  27  36  45  54  63  72  81 ]
```

(5) 500 個の 0 から 10000 までの乱数を要素に持つリストを作りなさい。そして，それを降順に並べ直しなさい。なお，0 から 10000 までの乱数は，Maxima では `random(10000)` で与えられます。

```
(%i123)  r:makelist(random(10000),i,1,500);


(%o123) [719, 2976, 1770, 3297, 8705, 4617, 8543, 7, 6486, 4692,
         4756, 246, 3560, 9431, 5302, 7228, 9731, 6607, 2797, 2785, 7922,

             (途中省略)

         6166, 8048, 6323, 2015, 2063, 243, 847, 2384, 2650, 6904, 2778,
         5177, 2315, 3722, 9325]



(%i124)  sort(r);


(%o124) [7, 46, 83, 97, 126, 170, 235, 243, 246, 265, 271, 274, 303,
         309, 331, 332, 347, 367, 382, 387, 402, 431, 457, 462, 505, 526,

             (途中省略)

         9503, 9518, 9573, 9582, 9582, 9584, 9588, 9629, 9707, 9731, 9736,
         9791, 9817, 9819, 9823, 9826, 9847, 9937, 9972, 9974, 9992]
```

ベクトルの演算

(1) ベクトル $\boldsymbol{v}_1 = (2.62, -0.12, 1.21)$, $\boldsymbol{v}_2 = (2.66, -0.44, 2.12)$ について，$1.27\boldsymbol{v}_1 - 0.92\boldsymbol{v}_2$ を求めなさい。

```
(%i125)  v1:[2.62,-0.12,1.21];

(%o125)                              [2.62, - 0.12, 1.21]


(%i126)  v2:[2.66,-0.44,2.12];

(%o126)                              [2.66, - 0.44, 2.12]


(%i127)  1.27*v1-0.92*v2;

(%o127)                           [0.8802, 0.2524, - 0.4137]
```

(2) ベクトル $\boldsymbol{v}_1 = (9+\sqrt{5}, 9-\sqrt{5}, 3\sqrt{2})$, $\boldsymbol{v}_2 = (5-\sqrt{5}, 5+\sqrt{5}, 2)$ について，ベクトル $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ の長さを求めなさい。また，それぞれの単位ベクトル (長さが 1 のベクトル) を求めなさい。

```
(%i128)  v1:[9+sqrt(5),9-sqrt(5),3*sqrt(2)];

(%o128)                    [sqrt(5) + 9, 9 - sqrt(5), 3 sqrt(2)]


(%i129)  v2:[5-sqrt(5),5+sqrt(5),2];

(%o129)                        [5 - sqrt(5), sqrt(5) + 5, 2]


(%i130)  sqrt(ratsimp(v1.v1));

(%o130)                                  sqrt(190)


(%i131)  sqrt(ratsimp(v2.v2));

(%o131)                                     8


(%i132)  v1/sqrt(ratsimp(v1.v1));

                          sqrt(5) + 9   9 - sqrt(5)   3 sqrt(2)
(%o132)                  [------------, ------------, ----------]
                           sqrt(190)     sqrt(190)    sqrt(190)


(%i133)  v2/sqrt(ratsimp(v2.v2));

                             5 - sqrt(5)   sqrt(5) + 5   1
(%o133)                     [------------, ------------, --]
                                   8             8       4
```

(3)　内積は 2 つの n 次ベクトル $\boldsymbol{v}_1, \boldsymbol{v}_2$ が n 次元空間の中で為す角度 θ を用いて，$\boldsymbol{v}_1 \cdot \boldsymbol{v}_2 = ||\boldsymbol{v}_1||\,||\boldsymbol{v}_2|| \cos(\theta)$ としても定義されます。このことを利用して，2 つの 3 次ベクトル $\boldsymbol{v}_1 = (-1, 0, 2)$, $\boldsymbol{v}_2 = (0, -1, 3)$ が 3 次元空間の中で為す角度を求めなさい。

```
(%i134)  v1:[-1,0,2];

(%o134)                                [- 1, 0, 2]


(%i135)  v2:[0,-1,3];

(%o135)                                [0, - 1, 3]


(%i136)  acos(v1.v2/(sqrt((v1.v1)*(v2.v2))));
                                                6
(%o136)                               acos(----------)
                                           5 sqrt(2)


(%i137)  float(acos(v1.v2/(sqrt((v1.v1)*(v2.v2)))));

(%o137)                               0.55759882669954
```

(4)　三角形 OAB の面積 S は，ベクトル $\overrightarrow{OA}, \overrightarrow{OB}$ を用いて，$S = \dfrac{1}{2}|\overrightarrow{OA} \times \overrightarrow{OB}|$ として，外積の大きさの半分で求まります。このことを利用して，3 点 $O = (0, 0, 0)$, $A = (1.12, 0.37, -0.88)$, $B = (1.71, 0.64, 0.95)$ が成す三角形の面積を求めなさい。

```
(%i138)  OA:[1.12,0.37,-0.88];

(%o138)                            [1.12, 0.37, - 0.88]


(%i139)  OB:[1.71,0.64,0.95];

(%o139)                             [1.71, 0.64, 0.95]


(%i140)  c:transpose(adjoint(matrix(OA,OB,[1,1,1])))[3];

(%o140)                        [0.9147, - 2.5688, 0.0841]


(%i141)  float(sqrt(c.c)/2);

(%o141)                             1.364045668223759
```

行列の演算

(1) 行列

$$A=\begin{pmatrix} t-1 & 1 \\ 0 & -t \end{pmatrix}, B=\begin{pmatrix} t & 0 \\ 1-t & -1 \end{pmatrix}$$

について，$2A+3B$, $AB-BA$, A^5+B^5 を求めなさい。

```
(%i142)  A:matrix([t-1,1],[0,-t]);

                              [ t - 1   1  ]
(%o142)                       [            ]
                              [   0    - t ]


(%i143)  B:matrix([t,0],[1-t,-1]);

                              [   t     0  ]
(%o143)                       [            ]
                              [ 1 - t  - 1 ]


(%i144)  ratsimp(2*A+3*B);

                        [ 5 t - 2      2     ]
(%o144)                 [                    ]
                        [ 3 - 3 t  - 2 t - 3 ]


(%i145)  ratsimp(A.B-B.A);

                       [     1 - t       - t - 1 ]
(%o145)                [                         ]
                       [    2                    ]
                       [ 2 t  - 3 t + 1   t - 1  ]


(%i146)  ratsimp(A^^5+B^^5);

                  [    5      4       3       2           ]
                  [ 2 t  - 5 t  + 10 t  - 10 t  + 5 t - 1 ]
(%o146) Col 1 =   [                                       ]
                  [     5      4      3      2           ]
                  [  - t  + 2 t  - 2 t  + 2 t  - 2 t + 1  ]
```

```
                                  [  4      3      2           ]
                                  [ t  - 2 t  + 4 t  - 3 t + 1 ]
                         Col 2 = [                            ]
                                  [             5              ]
                                  [          - t  - 1          ]
```

(2) 行列

$$\begin{pmatrix} -2.87 & 0.36 & -2.78 & -2.90 & -0.22 \\ -1.95 & -1.69 & 0.56 & -2.87 & -1.67 \\ -0.47 & 0.21 & -2.71 & 0.60 & 1.99 \\ 1.01 & 1.52 & -1.79 & 1.54 & 1.35 \\ -1.06 & -1.92 & -2.86 & -1.15 & 0.24 \end{pmatrix}$$

の行列式と逆行列を求めなさい。

```
(%i147)  mtx:matrix([-2.87,0.36,-2.78,-2.90,-0.22]
         ,[-1.95,-1.69,0.56,-2.87,-1.67],[-0.47,0.21,-2.71,0.60,1.99]
         ,[1.01,1.52,-1.79,1.54,1.35],[-1.06,-1.92,-2.86,-1.15,0.24]);

             [ - 2.87   0.36   - 2.78  - 2.9   - 0.22 ]
             [                                        ]
             [ - 1.95  - 1.69   0.56   - 2.87  - 1.67 ]
             [                                        ]
(%o147)      [ - 0.47   0.21   - 2.71   0.6     1.99  ]
             [                                        ]
             [  1.01    1.52   - 1.79   1.54    1.35  ]
             [                                        ]
             [ - 1.06  - 1.92  - 2.86  - 1.15   0.24  ]


(%i148)  determinant(mtx);

(%o148)                          - 4.742796508100006


(%i149)  invert(mtx);

                [ - 1.960158685729549 ]
                [                     ]
                [  0.17852846069907   ]
                [                     ]
(%o59)  Col 1 = [ - 0.37308034552569  ]
                [                     ]
```

```
[  2.097936886604074  ]
[                     ]
[ - 1.62239971224963  ]
```

```
          [  5.160916357721979  ]          [  0.99513618852071   ]
          [                     ]          [                     ]
          [  0.13161402959919   ]          [ - 0.10477522051627  ]
          [                     ]          [                     ]
Col 2 =   [  0.87094753969421   ]  Col 3 =  [  0.53565721102754   ]
          [                     ]          [                     ]
          [ - 6.250456887486373 ]          [ - 1.661472181347452 ]
          [                     ]          [                     ]
          [  4.275645078882732  ]          [  1.979010630957918  ]

          [  4.778897914192797  ]          [ - 1.018074136588739 ]
          [                     ]          [                     ]
          [  0.39941195595568   ]          [ - 0.29846567053476  ]
          [                     ]          [                     ]
Col 4 =   [  0.32023516029121   ]  Col 5 =  [ - 0.52446083776773  ]
          [                     ]          [                     ]
          [ - 5.221811544666378 ]          [  1.579409746382071  ]
          [                     ]          [                     ]
          [   3.0970501106075   ]          [ - 1.39937274952975  ]
```

(3) 行列

$$\begin{pmatrix} 1.51 & 1.48 & -0.81 \\ -2.45 & -1.17 & -2.21 \\ -2.30 & -2.12 & 1.74 \end{pmatrix}$$

の固有値と固有ベクトルを求めなさい。

```
(%i150)  mtx:matrix([1.51,1.48,-0.81],[-2.45,-1.17,-2.21],[-2.30,-2.12,1.74]);

                       [  1.51    1.48   - 0.81 ]
                       [                        ]
(%o150)                [ - 2.45  - 1.17  - 2.21 ]
                       [                        ]
                       [ - 2.3   - 2.12   1.74  ]


(%i151)  load(eigen);

(%o151)      C:/PROGRA~1/MAXIMA~1.2/share/maxima/5.9.2/share/matrix/eigen.mac


(%i152)  float(expand(eigenvectors(mtx)));
```

```
RAT replaced -0.81 by -81//100 = -0.81
RAT replaced 5.194000000000001 by 2597//500 = 5.194
RAT replaced 2.3 by 23//10 = 2.3
RAT replaced -1.17 by -117//100 = -1.17
RAT replaced -1.48 by -37//25 = -1.48
RAT replaced -5.083 by -5083//1000 = -5.083
RAT replaced -2.45 by -49//20 = -2.45
RAT replaced 1.74 by 87//50 = 1.74
RAT replaced -4.6852 by -11713//2500 = -4.6852
RAT replaced -1.17 by -117//100 = -1.17
RAT replaced 1.74 by 87//50 = 1.74
RAT replaced 1.51 by 151//100 = 1.51
(%o152) [[- 0.1884393480298 %i - 0.66986666681573,
         0.1884393480298 %i - 0.66986666681573, 3.419733333631462],
         [1.0, 1.0, 1.0]], [1.0, - 0.18980066885001 %i - 1.842547458561893,
         - 0.11415511341756 %i - 0.67543650846404],
         [1.0, 0.18980066885001 %i - 1.842547458561893,
         0.11415511341756 %i - 0.67543650846404],
         [1.0, 0.31995646942145, - 1.773083652947805]]
```

(4) 線形漸化式 $x_n = 5x_{n-1} - 6x_{n-2} + 2$, $x_1 = 1$, $x_2 = 2$ を，行列を用いて解きなさい。

```
(%i153)  load(eigen);

(%o153)      C:/PROGRA~1/MAXIMA~1.2/share/maxima/5.9.2/share/matrix/eigen.mac


(%i154)  r:matrix([5,-6,2],[1,0,0],[0,0,1]);
                                [ 5  - 6  2 ]
                                [           ]
(%o154)                         [ 1   0   0 ]
                                [           ]
                                [ 0   0   1 ]


(%i155)  e:eigenvectors(r);
                                             1           1
(%o155)        [[[3, 2, 1], [1, 1, 1]], [1, --, 0], [1, --, 0], [1, 1, 1]]
                                             3           2


(%i156)  p:transpose(apply(matrix,rest(e,1)));
```

```
                                  [ 1   1   1 ]
                                  [           ]
                                  [ 1   1     ]
(%o156)                           [ -   -   1 ]
                                  [ 3   2     ]
                                  [           ]
                                  [ 0   0   1 ]
```

```
(%i157)  p.matrix([3^(n-2),0,0],[0,2^(n-2),0],[0,0,1]).invert(p).[2,1,1];

                  [    n - 2      n - 2     ]
                  [ 3 3      - 2 2      + 1 ]
                  [                         ]
(%o157)           [    n - 2    n - 2       ]
                  [   3      - 2      + 1   ]
                  [                         ]
                  [            1            ]
```

従って，$n \geqslant 2$ において，$x_n = 3^{n-1} - 2^{n-1} + 1$ を満たす。

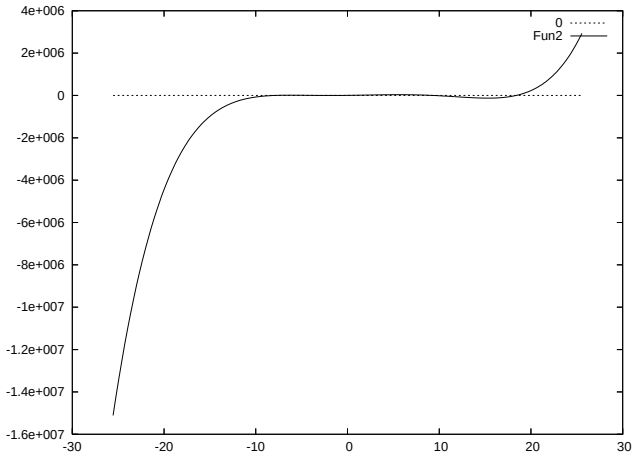
(5)　行列

$$\begin{pmatrix} 2 & -2 & 5 & 8 & -3 \\ 1 & 4 & 3 & 2 & 7 \\ 5 & 7 & -4 & 0 & 4 \\ 6 & 3 & 6 & 9 & 2 \\ 5 & 8 & 9 & 1 & 5 \end{pmatrix}$$

の固有値を数値近似しなさい。

```
(%i158)  mtx:matrix([2,-2,5,8,-3],[1,4,3,2,7],[5,7,-4,0,4]
         ,[6,3,6,9,2],[5,8,9,1,5]);

                        [ 2  - 2   5   8  - 3 ]
                        [                     ]
                        [ 1   4    3   2   7  ]
                        [                     ]
(%o158)                 [ 5   7   - 4  0   4  ]
                        [                     ]
                        [ 6   3    6   9   2  ]
                        [                     ]
                        [ 5   8    9   1   5  ]
```

```
(%i159)  load(nchrpl);

(%o159)     C:/PROGRA~1/MAXIMA~1.2/share/maxima/5.9.2/share/matrix/nchrpl.mac


(%i160)  n:length(mtx);

(%o160)                                  5


(%i161)  c:ncharpoly(mtx,x);
                     5       4        3         2
(%o161)             x  - 16 x  - 120 x  + 1115 x  + 4978 x + 4848

(%i162)  t:sqrt(sum(sum(mtx[i,j]^2,i,1,n),j,1,n));

(%o162)                                sqrt(653)


(%i163)  plot2d([0,c],[x,-t,t]);
```

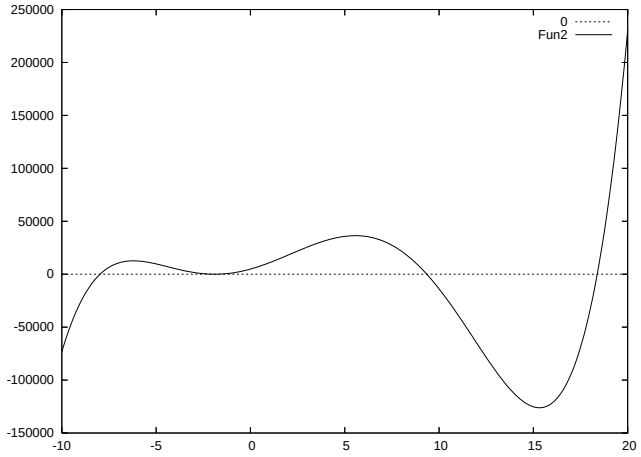


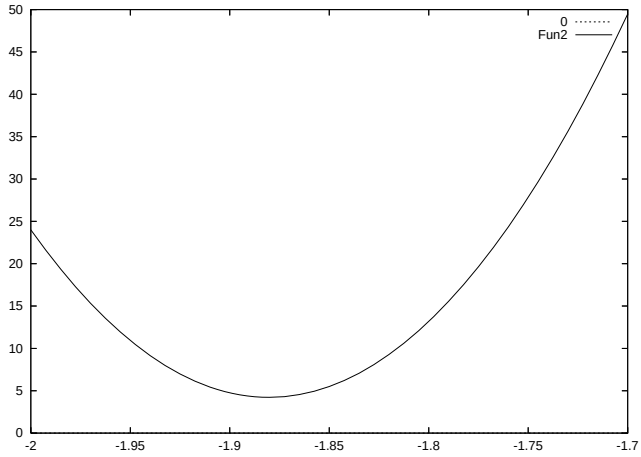

```
(%o163)
```

確かに全ての実数解を含むグラフのようですが，尺度の問題で交点の情報が全く読み取れません。t の範囲は理論的には十分なのですが，コンピュータには不向きです。この点は仕方がありません[5]から，人間が個別に判断して，手作業で範囲設定していく必要があります。まずは，それらしい範囲として $-10 \leqslant x \leqslant 20$ で描いてみます。

```
(%i164)  plot2d([0,c],[x,-10,20]);

(%o164)
```

これで，$x \fallingdotseq -8.01569$, $x \fallingdotseq 9.33907$, $x \fallingdotseq 18.4013$ の 3 つは読み取れました。ですが，まだ $x = -2$ 付近の様子が読み取れません。






```
(%i165)  plot2d([0,c],[x,-2,-1.7]);

(%o165)
```

これで 3 つの実数解と 1 組の共役複素数解があることが分かりました。では，まず実数解をニュートン法で近似しましょう。

```
(%i166)  load(newton);

(%o166)     C:/PROGRA~1/MAXIMA~1.2/share/maxima/5.9.2/share/numeric/newton.mac


(%i167)  a1:newton(c,-8.01569);


Warning:  Float to bigfloat conversion of -8.0156899999999993
```

(5)判断できる上手い方法があれば，是非教えて下さい。

```
(%o167)                          - 7.972000692037633B0

(%i168)  a2:newton(c,9.33907);

Warning:  Float to bigfloat conversion of 9.3390699999999995

(%o168)                           9.3477926515179B0

(%i169)  a3:newton(c,18.4013);

Warning:  Float to bigfloat conversion of 9.3390699999999995

(%o169)                           1.83848041774695B1
```

この 3 つの近似実数解 a_1, a_2, a_3 を使えば，ベアストウ法のような大掛かりな道具は必要ありません。といいますのも，固有多項式 $x^5 - 16x^4 - 120x^3 + 1115x^2 + 4978x + 4848$ を $(x - a_1)(x - a_2)(x - a_2)$ で割ると，少々誤差が出るものの，x の 2 次多項式が得られるからです。

```
(%i170)  float(partfrac(c/((x-a1)*(x-a2)*(x-a3)),x));

RAT replaced -1.83848041774695B1 by -7501/408 = -1.838480392156863B1
RAT replaced -9.3477926515179B0 by -24351/2605 = -9.347792706333973B0
RAT replaced 7.972000692037633B0 by 44986/5643 = 7.972000708842814B0
                      5.771508379631471E+20
(%o170) - ---------------------------------------------
          9.254060881013166E+26 x + 7.377337990311152E+27

                     2.987491147469051E+20
        + ---------------------------------------------
          4.32291185605756E+25 x - 4.040968391817951E+26

                      1.540795460535884E+17                2
        - ---------------------------------------------- + x
          1.466129755069242E+21 x - 2.695450807052546E+22

        + 3.760595919059786 x + 3.53856034167763


(%i171)  float(expand(solve(x^2+3.760595919059786*x+3.53856034167763,x)));
```

```
RAT replaced 3.53856034167763 by 2753//778 = 3.538560411311054
RAT replaced 3.760595919059786 by 14640//3893 = 3.760595941433342
(%o171) [x = - 0.05513576543246 %i - 1.880297970716671,
                     x = 0.05513576543246 %i - 1.880297970716671]
```

これで与えられた行列の固有値はおよそ,

$$\begin{cases} x \fallingdotseq -7.972000692037633, \\ x \fallingdotseq 9.3477926515179, \\ x \fallingdotseq 18.3848041774695, \\ x \fallingdotseq -1.880297970716671 - 0.05513576543246\boldsymbol{i}, \\ x \fallingdotseq -1.880297970716671 + 0.05513576543246\boldsymbol{i} \end{cases}$$

の 5 つであることが分かりました。

D.5　プログラミングの初歩

この解答例は，LISP の視点からは美しくないものもあります。ですが，読者の大部分は LISP よりは C などに親しんでいると思われますので，敢えて手続き型言語風に仕立ててあります。

(1)　気温 t を摂氏で与えると，その気温における音速を返す函数 mysndspd を作りなさい。ただし，気温 t のときの音速は $331.5 + 0.61t$ で得られます。

```
(%i172)  mysound(t):=331.5+0.61*t;

(%o172)                          mysound(t) := 331.5 + 0.61 t
```

(2)　1926 年以後の西暦を与えると，明治通算年，大正通算年，昭和通算年を返す函数 myeras を作りなさい。ただし，西暦 2001 年は，明治 134 年，大正 90 年，昭和 76 年に当たります。

```
(%i173)  myeras(year):=if integerp(year) and year>=1925
         then print("Meiji",year-1867,",Taisho",year-1911,",Showa",year-1925)
         else print("false");

(%o173) myeras(year) := if integerp(year) and year >= 1925

        then print("Meiji", year - 1867, ",Taisho", year - 1911, ",Showa",

        year - 1925) else print("false")
```

(3)　2 変数関数 $z = f(x, y)$ と独立変数の組 $[x, y]$ を与えると停留点を返す函数 mycritpts を作りなさい。ただし，関数 $z = f(x, y)$ について，$\dfrac{\partial z}{\partial x} = 0$, $\dfrac{\partial z}{\partial y} = 0$ を満たす (x_i, y_i) の組が，z の停留点 $(x_i, y_i, f(x_i, y_i))$ を与えます。

```
(%i174)  mycritpts(f,v):=float(solve(makelist(diff(f,v[i])=0,i,1,length(v)),v));

(%o174) mycritpts(f, v) := float(solve(makelist(diff(f, v ) = 0, i, 1,
                                                                 i
                                                                 length(v)), v))
```

(4) 乾球の温度を c_1, 湿球の温度を c_2 を与えると不快の度合いを示す関数 mydiscomfort を作りなさい。不快の度合いは，不快指数 i を $i = 40.6 + 0.72(c_1 + c_2)$ の式で計算し，

- $i < 70$ の場合，快適。
- $70 \leqslant i < 75$ の場合，一部不快。
- $75 \leqslant i < 80$ の場合，半数不快。
- $80 \leqslant i$ の場合，全員不快。

で決めます。

```
(%i175)  mydiscomfort(c1,c2):=block([i:40.6+0.72*(c1+c2)],
         if i>=80 then return("All discomfort")
             else (if i>=75 then return("Half discomfort")
             else (if i>=70 then return("Some discomfort")
             else return("Comfort")
         )));

(%o175) mydiscomfort(c1, c2) := block([i : 40.6 + 0.72 (c1 + c2)],
        if i >= 80 then return("All discomfort")
         else (if i >= 75 then return("Half discomfort")
         else (if i >= 70 then return("Some discomfort")
         else return("Comfort"))))
```

(5) Maxima に n 以下の自然数を選ばせて，人がその数を推測して入力するたびにより大きいか小さいかを返す，数当てゲーム mybigsmall を作りなさい。

```
(%i176)  mybigsmall(n):=block([r,a:0,i],rnd:random(n),
         for i:1 while a#r do(
             a:read("Trial",i,":"),
             if a>r then print("Too big!")
                 else (if a<r then print("Too small!")
                 else print("Bull's eye!"))),
         return("DONE"));

(%o176) mybigsmall(n) := block([r, a : 0, i], r : random(n),
        for i while a # r do (a : read("Trial", i, ":"),
```

```
if a > r then print("Too big!")
 else (if a < r then print("Too small!")
 else print("Bull's eye!"))), return("done"))
```

(6) 2 つのリストを与えると，その積集合を返す函数 myintersection を作りなさい。

```
(%i177)  myintersection(p,q):=block([lst:[],i],
         for i:1 thru length(p) do(
             if member(p[i],q)) then lst:endcons(p[i],lst)),
         return(sort(lst)));

(%o177) myintersection(p, q) := block([lst : [], i],
        for i thru length(p) do (if member(p , q)
                                           i
         then lst : endcons(p , lst)), return(sort(lst)))
                           i
```

(7) 2 つのリストを与えると，その和集合を返す函数 myunion を作りなさい。

```
(%i178)  myunion(p,q):=block([lst:[],i],
         for i:1 thru length(p) do(
             if not(member(p[i],lst)) then lst:endcons(p[i],lst)),
         for i:1 thru length(q) do(
             if not(member(q[i],lst)) then lst:endcons(q[i],lst)),
         return(sort(lst)));

(%o178) myunion(p, q) := block([lst : [], i],
        for i thru length(p) do (if not member(p , lst)
                                                 i
         then lst : endcons(p , lst)), for i thru length(q)
                           i
        do (if not member(q , lst) then lst : endcons(q , lst)),
                          i                              i
        return(sort(lst)))
```

(8) csv 形式のファイル名と左から何列目を選ぶかを与えると，その列をリストとして返す函数 mycolumn を作りなさい。

```
(%i179)  load("C:/PROGRA~1/MAXIMA~1.2/share/maxima/5.9.2/share/contrib/
         numericalio/numericalio.lisp");
```

```
(%o179)      "C:/PROGRA~1/MAXIMA~1.2/share/maxima/5.9.2/share/contrib/numer#
                                icalio/numericalio.lisp"

(%i180)  mycolumn(f,c):=transpose(read_matrix(f))[c];

(%o180)                mycolumn(f, c) := transpose(read_matrix(f))
                                                                    c
```

(9) read_matrix 函数は，読み込める行列のサイズに限界があります。そこでサイズに余裕がある read_list 函数を利用して，csv 形式のファイル名と列数を与えると，行列を返す函数 myread_matrix を作りなさい。

```
(%i181)  myread_matrix(f,c):=block([lst:read_list(f),r,i,j],
         r:length(lst)/c-1,if integerp(c) and c>0 and integerp(r)
             then return(apply(matrix,makelist(makelist(lst[c*i+j],j,1,c),i,0,r)))
         else return("false"));

(%o181) myread_matrix(f, c) := block([lst : read_list(f), r, i, j],
            length(lst)
        r : ----------- - 1, if integerp(c) and c > 0 and integerp(r)
              c
         then return(apply(matrix, makelist(makelist(lst       , j, 1, c),
                                                   c i + j

        i, 0, r))) else return("false"))
```

(10) 4 列の csv 形式のファイルを与えると，各行を係数とする 3 次方程式の解を実数部列，虚数部列に分けた 6 列の csv 形式のファイルを作成する函数 mysolver を作りなさい。ただし，引数は読み込みファイル名と書き出しファイル名を与えるものとします。

```
(%i182)  mysolver(r,w):=block([c:read_matrix(r),i,j,s,x,p,m:[]],
         for i:1 thru length(c) do(s:expand(solve(c[i].[x^3,x^2,x,1]=0,x)),
         p:makelist(if j<=length(s)
             then float([realpart(rhs(s[j])),imagpart(rhs(s[j]))])
             else ["false","false"],j,1,3),
         m:endcons(apply(append,p),m)), write_data(apply(matrix,m),w,'csv),
         return(m));

(%o182) mysolver(r, w) := block([c : read_matrix(r), i, j, s, x, p,
        m : []], for i thru length(c) do (s :
                           3   2
        expand(solve(c   . [x , x , x, 1] = 0, x)),
                     i
```

```
p : makelist(if j <= length(s) then float([realpart(rhs(s )),
                                                          j
imagpart(rhs(s ))]) else ["false", "false"], j, 1, 3),
              j
m : endcons(apply(append, p), m)),
write_data(apply(matrix, m), w, 'csv), return(m))
```

おわりにあれこれ

長々とご閲覧，ありがとうございました。如何でしたでしょう，Maxima を使ってみたいと思っていただけましたでしょうか。なお，大学の定期試験に Maxima は持ち込まないようにしましょう。

この『Maxima 入門ノート』は，これまで数式処理ソフトウェアに触れたこともない，という初心者を読者対象に想定して書き綴ったものです。ですから，この入門ノート程度は修得なさった方には，2006 年 2 月現在，ぽんぽこ氏が執筆中の市販テキスト[6]がお勧めです。まあ，筆者もネタはあるけれど書くのが面倒くさいという理由で，Maxima の応用書・実用書を書くのを怠けております。

この『Maxima 入門ノート』を市販しては如何かというご提案も沢山受けましたが，筆者のポリシーとして,『Maxima 入門ノート』そのものは GNU Free Documentation License のまま，ずっとオンラインでフリードキュメントとして管理いたします。ですから，コピーを情報処理センターなどにおいていただくのも，全く自由です。特に学生の皆さん,「ありがたい」と思っていただけるのでしたら，Maxima も上手に活用してしっかり勉強・研究してください。なお，いずれ応用書・実用書は市販していきたいと思っていますので，そのときはご購入いただけると嬉しく思います[7]。

最後に，今後のバージョンナンバーについてのポリシーを述べておきます。と申しますのも，バージョンナンバーは，これまで結構いい加減にアップしていたので，混乱を招いた経緯があるからです。このバージョンは，1.2.1 ですが，このピリオドで区切られた番号を左から，メジャー暗号，マイナー番号，リビジョン番号を呼びますと，

- 全体の構成を大きく書き替えた場合は，メジャー番号を更新し，マイナー番号とリビジョン番号は 0 にクリアする。
- Maxima のバージョンアップに伴なう内容変更については，メジャー番号はそのままで，マイナー番号を更新し，リビジョン番号は 0 にクリアする。
- 誤植の訂正や図表の差し替えなどが，細かい訂正がある程度進んだら，メジャー番号とマイナー番号はそのままで，リビジョン番号を更新する。

という方針で更新していこうと思います。

[6] ・・・と，ぽんぽこ氏の尻を叩いておく。
[7] まあ，印税生活する気はありませんので，その辺は適当で。

索 引

., 59, 60
?, 5, 98, 101
?truncate, 72

acos, 10
acosh, 11
acot, 10
acoth, 11
acsc, 10
acsch, 11
adjoint, 59, 64
and, 70
append, 56
asec, 10
asech, 11
asin, 10
asinh, 11
atan, 10
atanh, 11
atvalue, 4, 50

bfloat, 8, 11, 103
block, 75
    return, 75
    局所変数, 75, 76
bug_report, 99

cabs, 9
cons, 56
cos, 10
cosh, 11
cot, 10
coth, 11
csc, 10
csch, 11
csv 形式, 78
    — の書き出し, 78
    — の読み込み, 78

del, 45
denom, 32
describe, 5
desolve, 4, 50
determinant, 4, 63
diff, 3, 45
do, 73

%e, 9
eigenvalues, 65
eigenvectors, 65
else, 70
endcons, 56
erf, 48
evenp, 70
exp, 9
expand, 4, 31

factor, 32
float, 8, 11
for, 73
    step, 73
    thru, 73
    while, 74
fpprec, 8, 103

gnuplot, 23
    load, 24
    set
        cntrparam, 26
        contour, 26
        hidden3d, 24
        output, 28
        surface, 27
        terminal, 28
        view, 25
    splot, 24
    unset

contour, 26
hidden3d, 24
surface, 27

%i, 9
ident, 62
if, 70
imagpart, 69
inf, 43
integerp, 70
integrate, 3, 47
invert, 64

limit, 43
minus, 43
plus, 43
linel, 98
listp, 70
load, 37
loadfile, 77
log, 10

makelist, 55
matrix, 4
matrixp, 70
mattrace, 64
minf, 43

mcharpoly, 65
newton, 37, 66
num, 32
nusum, 47

oddp, 70
ode2, 51
or, 70

partfrac, 33
%pi, 9
playback, 77
plot2d, 1, 17
plot_format, 24
媒介変数表示の—, 20
plot_format, 24
plot3d, 2, 5, 21
grid, 22
媒介変数表示の—, 23
primep, 70
product, 46

quit, 97

ratcoef, 31
ratnump, 70
ratsimp, 4, 32
read_list, 78, 80
read_matrix, 78
realpart, 69
rest, 56
reverse, 57

save, 77
sec, 10
sech, 11
simpson, 48
sin, 10
sinh, 11
solve, 2, 34, 101
sort, 57
ordergreatp, 57
sqrt, 9
sum, 46

tan, 10
tanh, 11
taylor, 45
TEX, 28, 100
then, 70
transpose, 59, 61
traprule, 48

write_data, 78

zeromatrix, 62

インストール, 89
陰線処理, 24
— の解除, 24
円周率, 9

関数, 12
—の消去, 13
機械翻訳サービス, 6

起動方法, 1
級数和, 46, 47
強制停止方法, 5
行列, 4, 58
— の跡, 64
— の加減法, 58
逆—, 63, 64
行列式, 4
— の固有多項式, 65
— の固有値, 65
— の固有ベクトル, 65
—式, 63
— のスカラー倍, 58
— の成分, 59
— の積, 60
零—, 62
対角—, 62
単位—, 62
転置—, 61
— のトレース, 64
余因子—, 64
— の累乗, 60, 74
極限, 43
左—, 43
右—, 43
虚数単位, 9
グラフ, 17
平面的な—, 1
—の重ね描き, 18
媒介変数表示の—, 19
陽関数の—, 17
立体的な—, 2
—の視点変更, 22
媒介変数表示の—, 22
陽関数の—, 21
クロソイド, 49
計算順序, 7
誤差関数, 48
コルヌの螺旋, 49

三角関数, 10
逆—, 10
逆双曲—, 11
双曲—, 11
算術平均, 75
指数関数, 9
指数乗, 7, 102
自然対数, 10
—の常用対数化, 10
—の底, 9
四則演算, 7, 102
終了方法, 4
消去
関数の—, 13
変数の—, 12
条件式, 70
数値近似, 8
精度指定, 8
積分, 3, 47
数値—, 48
シンプソン公式, 48
台形公式, 48
—定数, 47
絶対値, 9
漸近線, 43
素因数分解, 32

多項式, 31
—の因数分解, 32
— の簡約化, 4
—の係数, 31
— の展開, 4
—の展開, 31
定義
関数の—, 12
—の消去, 12
変数の—, 11
停止方法, 5
テイラー展開, 45
デバグ, 69, 71
導関数, 45
等高線, 25
— のオプション, 26

ニュートン法, 37
— の弱点, 38

パッケージ, 37
eigen, 65

nchrpl, 64
newton, 66
newton —, 37
numericalio, 78
simpsn —, 48
反復手続き, 73
引数, 12
微小量, 45
微分, 3, 44
微分方程式, 50
初期値, 4, 50
—を解く, 4, 50
連立—, 50
標準偏差, 75
標本—, 75
不偏—, 75
双子素数, 70
ブロック化, 75
分岐構造, 70
平方根, 9
ベクトル, 4, 58
— の大きさ, 59
— の外積, 59
— の加減法, 58
— のスカラー倍, 58
— の成分, 59
— の内積, 59
ヘルプ, 5, 98
函数名一覧の—, 6
変数, 11
局所, 75
方程式, 34
— の解の数値化, 3, 35
— の数値近似解, 3, 36
— を解く, 2, 34
不定 —, 36
不能 —, 36
連立 —, 35
保存, 77
手順の—, 77

無限大, 43

ユーザ函数, 69
mybigsmall, 81, 149
mybmi, 71
mycolumn, 81, 150
mycomplex, 69
mycritpts, 81, 148
mydiagonal, 70
mydiscomfort, 81, 149
myeras, 81, 148
mygeomean, 80
myint, 72
myintersection, 81, 150
myread_matrix, 81, 151
mysolver, 81, 151
mysound, 81, 148
mystatistics, 75
mytwinprime, 70
myunion, 81, 150
有理式, 32
— の漸近線, 33
—の通分, 32
— の部分分数分解, 33
—の分子, 32
—の分母, 32

リスト, 55
— の並べ替え, 57
— のネスト, 55
— の要素
— の追加, 56
要素, 55
連結, 56