Maple V の基本操作と実践

第３版(リリース５対応)

西谷滋人

京都大学工学研究科材料工学科

１９９８年１０月

Department of Materials Science and Engineering•
Kyoto University•Kyoto 606-8501, Japan
bob@karma.mtl.kyoto-u.ac.jp

本書は PowerMac 上で Maple V Release5 と PageMaker6.5J とを使用して編集，整形を行ないました．

Maple V の基本操作と実践 第 3 版(リリース 5 対応)
(Maple VR5: Essentials and Applications on Materials Science)

まえがき

"Any sufficiently advanced technology is indistinguishable from magic." Arthure C. Clarke の言葉です．Mathematicaという数式処理ソフトを初めて使ったときにもこれと同じ感覚を持ちました．数式処理には大型計算機センターのReduceを使うしかなかった当時に，私の持っていたMacSEで複雑な数式を一瞬にして簡単化し，数式らしく表示してくれたのを目のあたりにしたとき，また一歩科学が魔法に近づいた気がしました．

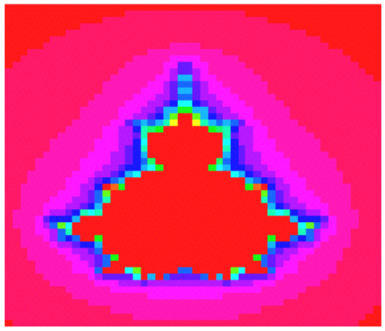
本書で使用法を解説するMapleVは1985年に，Mathematicaに先駆けて市場に現れました．カナダのWaterloo大学とETH Zürichでの研究成果をもとにした計算ライブラリの内部処理は公開されており，その数式処理に対する信頼性の高さで定評があります．日常，私がMapleをどのように使っているかというと，紙と鉛筆，電卓，グラフソフト，programming言語の代わりとして，（１）論文を読むときの数式の導出　（２）ちょっとしたルーチン計算，（３）結果のグラフ化，そして（４）プログラムを作るときのプロトタイプの作成，等です．

Mandelbrot集合を描くという実例を取り上げてみましょう．専門書をひもとくと(「C言語によるアルゴリズム事典」，奥村晴彦著，技術評論社 1991）

> Mandelbrot(マンデルブロート）集合　Mandelbrot set
> (中略)
> 計算機では，ある領域の点(x,y)について，
> z <-- x + i y; count <-- M;
> while (|z| <= 4) and (count>0) do begin
> z <-- z^2 - (x + i Y); count <-- count -1
> end;
> 点(x,y)に count で決まる色(count=0 なら黒)を付ける;
> とする．黒い部分が Mandelbrot 集合で，それ以外の色は，その点と Mandelbrot 集合との ”近さ ” を表す光背である．

となっています．この後，プログラムコードが続き，どこかにあるGraphics表示用の謎のルーチンを呼ぶように指示してあります．これをMapleで実現しようと思うと，

```
> mandel:=proc(x,y)
  local z0,z,count;
  z0:=x+y*I;
  z:=0;
  count:=20;
  while (evalf(abs(z))<4.0 and count>0) do
    z:=z^2-z0;
    count:=count-1;
  od;
  count;
  end:
> with(plots):
> densityplot(mandel,-1..2.5,-1.5..1.5,axes=none,colorstyle=HUE,
  style=patchnogrid,grid=[50,50]);
```



となります．どうです？！　数学的な記述そのままで，しかもわずか数分で目的とするMandelbrot集合を

実際に描くという作業が終了してしまいます.

数式処理や関数のグラフ化等を扱う優秀な市販ソフトウェアとしてはMathematicaとMapleが有名です. どちらかの使い方になじんでいればもう一方の, あるいは今後現れるであろうより優秀なソフトを修得するのにそれほど時間はいらないと思います. 本書ではソフトの詳しい構造や数学の厳密な適用などには余り注意を払っていません. とにかく問題を理解し, 解いていくためにMapleをいかに使用するか, つまり道具として数学(Maple)をいかに使うかに力点を置いて解説しています. ちゃんとした文法やコマンド引数などはhelpを参照し, 試行錯誤をお願いします.

本書はMaple V Relaese 5に基づいて解説しています. リリース４からリリース５への変更はGUI周りの強化が主で, 内部的にはほとんど変わっていません. したがって, 古い教科書(つまり知識)がそのまま使えます. 例外的に影響が大きい変更点は二ヶ所,

直前の出力を参照する記号が ” から % に変わった.

文字列を囲む記号がバッククォート(`)からダブルクォート(“)になった.

です. その他の変更点はヘルプの ” What ’ s new ” から引けます. Maple Vに関する解説書が最近盛んに出版されています. 一部を紹介します.

” MapleVによる数式処理入門 ” 阿部寛著（講談社, 1997)

間違いなく現在日本語で読める絶好の入門書. 著者が長年(たぶん？), 専門で使い込いこまれた経験が凝縮されています. 特に微分方程式の具体的な解き方は充実しています. リリース5でもほとんど書き換える必要なく使えます.

” MapleV リリース5ラーニングガイド ” K.M. ヒール, M.L. ハンセン, K.M. リカード著, 示野信一他訳（シュープリンガー・フェアラーク東京, 1998)

新しいリリース5の "Maple V Learning Guide" の訳.

“ Maple V Programming Guide, ”(M.B.Monagan et al. Springer 1998）

プログラミングやMapleの内部構造などをより詳細に知りたいときには購入する必要があります. 内容は一般的なhelpでは得られないようなMapleの概念を統一的に説明しています. ただ, 記述が抽象的（数学的？）なため初心者には何を言っているのか全く理解できません. 日本語訳は近刊予定です. Maple Vを系統的に理解し, 効率良く使うには是非とも必要です.

” はじめてのMapleV リリース４ ” K.M. ヒール, M.L. ハンセン, K.M. リカード著, 笠島友美訳（シュープリンガー・フェアラーク東京, 1997)

リリース４の "Maple V Learning Guide" の訳. リリース３やリリース５の日本語版がいかにもマニュアルの翻訳という雰囲気で読みにくいのに対して, リリース4版は数学を理解している人がコマンドを確認しながら翻訳されたらしく, とても理解しやすくなっています. リリース5でも使えます.

” Maple Vと利用の実際 - 数式処理とCG- ” 小国　力著（サイエンス社 1997)

多くの解説書(MATLAB,Mathematice)を出している著者によるMaple V解説書. 広い分野を網羅しており, 経験のある研究者が具体的な問題を解くときの足掛かりとなる.

優れた題材とテキストを提供下さった材料工学科の沼倉宏先生と河合潤先生に感謝いたします. 特に第６章の題材は河合潤先生の手によるものです. また, Mapleの導入・管理では応用システム科学教室の河野浩之先生のお世話になりました. 心より感謝いたします. このテキストは河野先生の強力な後押しで完成することができました. なお, 誤りや不正確な記述が多々あると思いますが, 適宜訂正していきますので, e-mailで御連絡下さい.

目次

第１章　最初の一歩
　１）Maple の起動法 ---------- 1
　２）簡単な演算 ---------- 1
　３）間違い修正 ---------- 1
　４）実行の中断 ---------- 2
　５）ヘルプファイル ---------- 2
　６）保存と終了 ---------- 3
第２章　簡単な数式処理とプロット
　１）変数，式の定義とそのキャンセル ---------- 4
　２）関数 ---------- 5
　３）プロット ---------- 5
　４）方程式の解 ---------- 7
　５）式の変形 ---------- 7
第３章　微積分とその応用
　１）総和の計算 ---------- 8
　２）極限 ---------- 8
　３）微分 ---------- 9
　４）微分方程式 ---------- 9
　５）積分 ---------- 9
　６）級数 ---------- 10
　７）複素関数 ---------- 10
第４章　データ構造と線形代数
　１）集合，リスト，表，配列 ---------- 12
　２）線形代数 ---------- 15
第５章　MapleV でのプログラミング
　１）プログラミングの流れ ---------- 17
　２）Maple の制御構造 ---------- 17
　３）プログラミングの実践 ---------- 18
　４）その他のテクニック ---------- 21
第６章　データ処理
　１）データの入出力 ---------- 23
　２）フーリエ変換 ---------- 24
　３）畳み込み ---------- 26
　４）非線形最小二乗法 ---------- 27
実践演習
　実践演習[１] 化学反応式の係数決定（連立方程式の整数解）---------- 32
　実践演習[２] トンネル効果（式の変形）---------- 33
　実践演習[３] 熱膨張係数の導出（複雑な関数の近似と積分）---------- 38
　実践演習[４] 陽電子消滅寿命（連立微分方程式）---------- 40
　実践演習[５] Hamiltonian の解（線形代数）---------- 41
　実践演習[６] オイラー角（線形代数）---------- 42
　実践演習[７] 組成自由エネルギー曲線（連立方程式の数値解）--------- 43
　実践演習[８] 減衰振動のフーリエ変換（高速フーリエ変換) ---------- 46

付録Ａ　コマンドのまとめ
　　1）関数パッケージ ---------- 47
　　2）式の変形 ---------- 47
　　3）線形代数 ---------- 48
付録Ｂ　入出力に関する補足
　　１）データの入出力（フィルターとしての使用）---------- 49
　　２）C, LATEX への出力 ---------- 50
　　３）LaTeX,HTML 書類への保存---------- 50
索 引---------- 51

第１章　最初の一歩

ここではMapleの基礎的な操作法と注意事項を述べます.

１）Mapleの起動法

PCではほかのソフトウェアと同様の立ち上げ方で起動します. unixではxmapleとするとmotif版のmapleが立ち上がります．mapleだけですと普通のvt100上でのtext版が立ち上がります. これをunixのredirection機能を使ってfilterとして機能させることも可能です(付録Ｂデータの入出力参照)．起動できない場合はPATHとexecutabilityを確認して下さい.

２）簡単な演算

それでは簡単な計算を実行させてみましょう.

```
> 1+1;(enter)
```

$$2$$

これ以降は最後の(enter)を省きます．enterとshift+enterは違った意味を持ちます．複数行にまたがる入力ではenterの代わりにshift+enterを入力します．最後にenterをいれれば，その領域すべてを一度に入力したことになります.

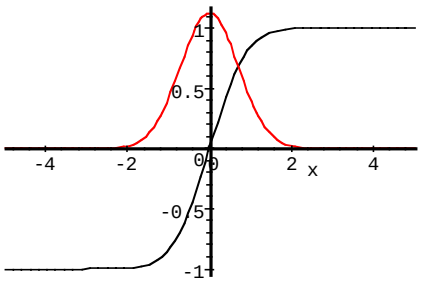
```
> set1:={1,2,3};（ここではshift+enterで改行だけを入れています）
  set2:={3,4};   （ここではenterで入力しています）
```

$$set1 := \{1, 2, 3\}$$

$$set2 := \{4, 3\}$$

本書ではこれ以降，複数行にまたがる入力の際には2行目以降の入力のプロンプト(>)をつけていません．最後の；（セミコロン）を忘れがちです．出力を抑止したいときには最後の；を：（コロン）にすれば，なにも出力しません．ただし，内部での代入は実行されています．次に関数のプロットをしてみましょう.

```
> plot({erf(x),diff(erf(x),x)},x=-5..5);
```



となります.

３）間違い修正

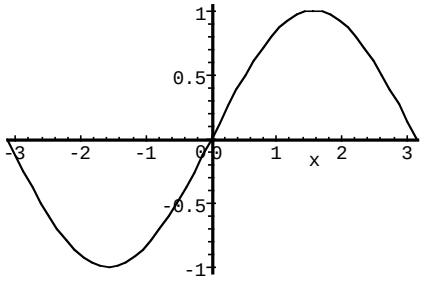
打ち間違いなどの訂正はアローキー, あるいはマウスのクリックによってできます. 後は訂正してenterを入れれば入力されます．ある領域を選択して，削除・カットおよびペーストなどの作業もマウスを使ってできます. 例えばサイン関数をーπから＋πまでプロットしようとした時

```
> plot({sin(x)},x=-pi..pi);
```

Error, (in plot) parameter range must evaluate to a numeric

という警告が出ました. これはMapleでは大文字と小文字を区別しているためにおこったことです．そこで，pi->Pi と修正すると無事表示されます.

```
> plot({sin(x)},x=-Pi..Pi);
```



4）実行の中断

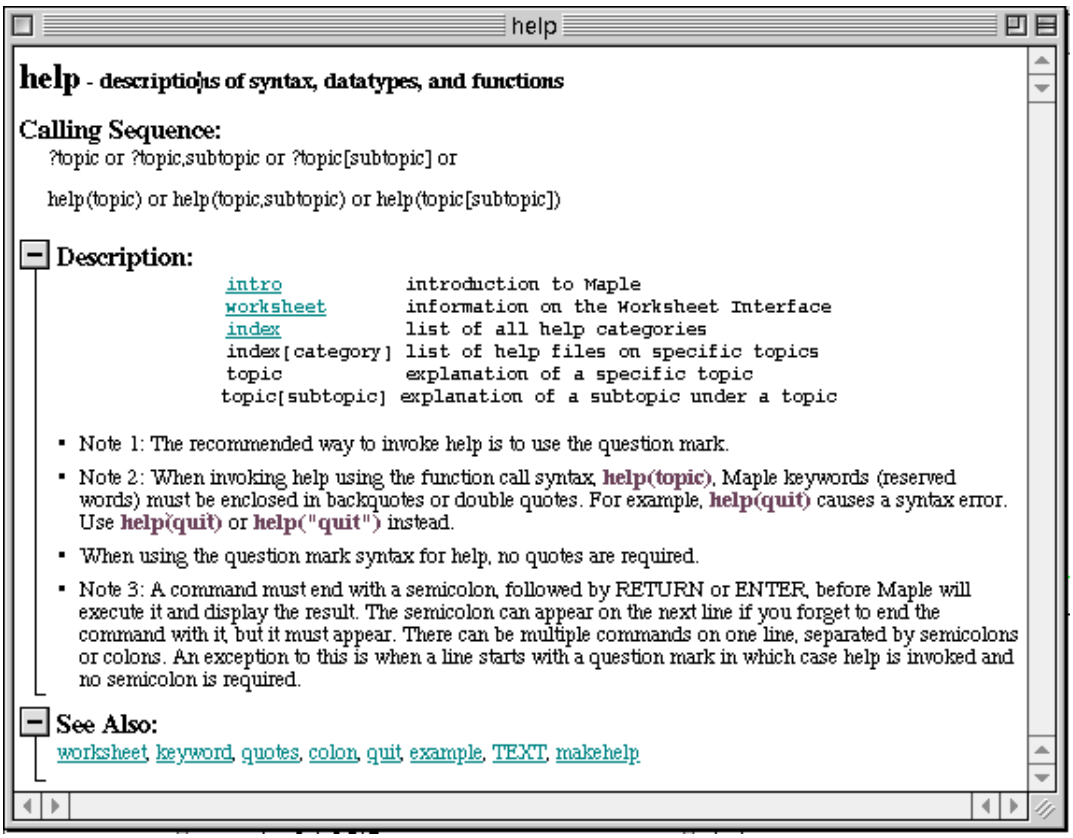
enter を押した後，入力ミスに気づいて計算をやめさせたいときには中断ができます．PC では tool bar の stop マークで，そして unix では interrupt button で中断します.

5）ヘルプファイル

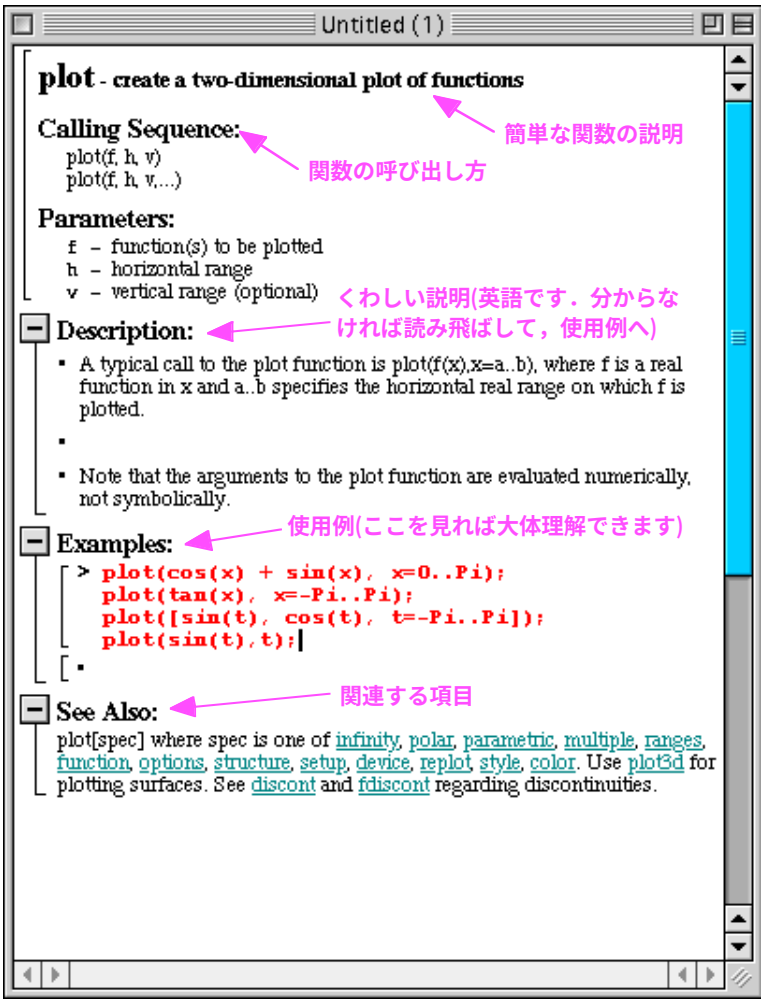
```
> ?(key word)
```

でキーワードに関するヘルプが表示されます．その他，？？や？ index あるいは？（キーワードの最初の一部）を使って，類推によってキーワードの情報を取り出すことができます．その他の help に関する操作はメニューバーの ”Help ”にいくつか用意されています．以下は help の画面出力です.

```
>?help;
```



```
> ?plot;
```



6）保存と終了

保存(save)はメニューバーからセレクトします．終了はメニューバーから exit(Quit)です．

第２章　簡単な数式処理とプロット

簡単な数値の代入と関数，関数の定義と式の変形，グラフのプロットを扱います．式の変形は難しく，なかなか思うように単純化してくれません．人間だとすぐに分かることも，Mapleに教えてやらなければなりません．しかし，Maple はややこしい式の変形でも係数や次数を見落とすことはありません．実践演習［2］を参考にうまいやり方を身に付けて，Maple を積極的に活用して下さい．

１）変数，式の定義とそのキャンセル

Maple の基本的な代入は

```
> mass:=10;
```

$$mass := 10$$

によって行われます．あらかじめ Maple で定義されていてよく使う定数は Pi, I, infinity などです．(?ininames 参照)

式の定義も同様に行えます．

```
> force:=- mass*accel;
```

$$force := -10\ accel$$

直前の結果を参照するには % を使います．

```
> exp1:=%;
```

$$exp1 := -10\ accel$$

これら一度数値を入れた変数を元へ戻すには

```
> restart;
```

によって行います．これで起動初期のなにも入力されていない状態に戻ります．ひとつの定義だけを初期状態に戻したいときには ’（シングルクォート）をもちいて

```
> mass:='mass';
```

$$mass := mass$$

によって行います．これによって，

```
> force:= - mass*accel;
```

$$force := -mass\ accel$$

となります．

一時的な代入は subs で行います．

```
> subs(mass=10,accel=14,force);
```

$$-140$$

こうすると，

```
> force;
```

$$-mass\ accel$$

とそれぞれの変数が数値を取るのではなく，変数のままで扱われます．逆に一時的に値ではなく変数そのものを使用するにもシングルクォートを使います．

```
> x := 2; y := 3;
  f:='x+y';
  g:=x+y;
```

$$x := 2$$

$$y := 3$$

$$f := x + y$$
$$g := 5$$

（注：続けて入力される方はここら辺で restart をかけてください．以下では数値を代入した変数を，free の変数として使っています．そのまま入力を続けると叱られます）

2）関数

よく使う関数はそのままの形で使えます．三角関数(trigonometric functions)はラジアンで入れてください．log（もちろん ln も）は自然対数です．底をあらわにするときは

```
> log[2](5);
```

$$\frac{\ln(5)}{\ln(2)}$$

としてください．数値として取り出したいときには

```
> evalf(%);        --->evaluating float の略です．
```

$$2.321928094$$

とします．

Maple が提供する膨大な数の関数から，目的とするものを探しだすには help を使って下さい．以下に関数等の index を表示する keywords をまとめておきます．

?inifcns - 起動時から認識されている関数

?index[package] - 関連する関数を集めたパッケージです．微分方程式(DETools),線形代数(linalg), プロット関係の関数(plots)等があります．

?index[function] - Maple の標準関数．

これらの関数は起動時から読み込まれている関数と，ユーザーが呼び出さなければならない関数とがあります．呼び出しが必要な時には

```
>readlib(cmd)
```

あるいは

```
>with(package)
```

でおこなってください．

単純なユーザー関数の定義は次の 2 種類をよく使います．

i)　矢印による定義. (すぐ下に例があります)

ii)　unapply による定義.

unapply は

```
>eq1:=exp(-x)*cos(10*x);
> f1:=unapply(eq1,x);
```

のように，一度求めた式を関数として定義するときに使います．ややこしい操作が必要な関数を定義するには第 5 章で示す proc を使います．

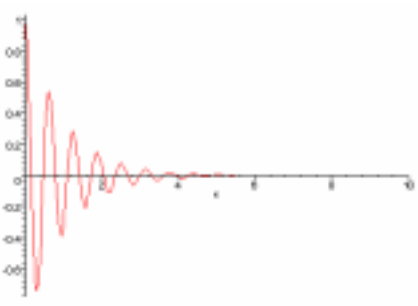
3）プロット

複雑な関数がどのような形をしているかを直感的に理解しておくのは常に重要なことです．先程のユーザー定義関数を使って，関数が実際にどのような形をしているかplotさせてみましょう．

```
> eq1:=x->exp(-x)*cos(10*x);
```

$$eq1 := x \to \mathbf{e}^{(-x)} \cos(10\,x)$$

```
> plot(eq1(x),x=0..10);
```



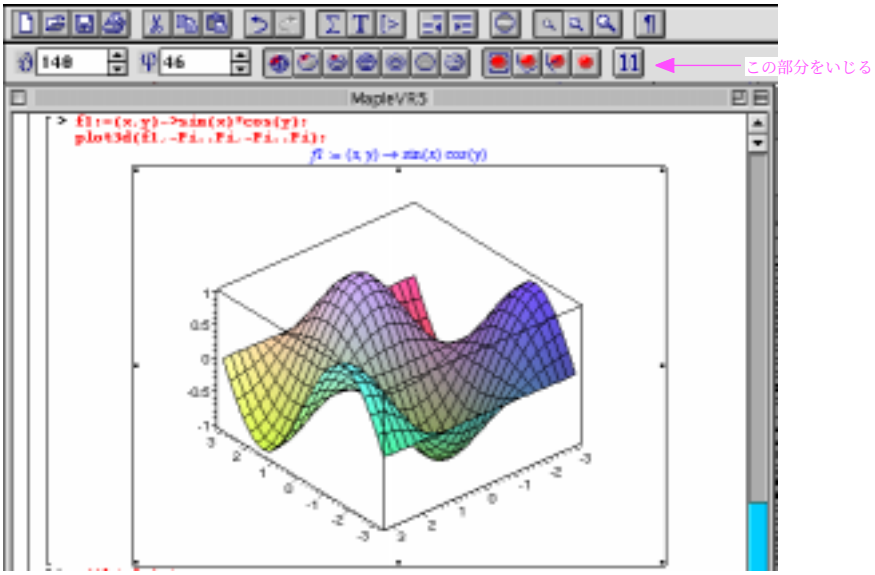
となります．パラメーターによるプロットも可能です．例えば

```
> plot([sin(t), cos(t), t=-Pi..Pi]);
```

によって円が描けます．真円にしたいときには出力図形を選んだ後，メニューバー下段の(1:1)ボタンを押してください（下図参照）．２変数の場合には plot3d を使います．変数名が明らかな時には省略可能です．

```
> f1:=(x,y)->sin(x)*cos(y);
  plot3d(f1,-Pi..Pi,-Pi..Pi);
```

$$f1 := (x, y) \rightarrow \sin(x)\cos(y)$$



オプションや特殊な plotting 法があります．plot に対する一部の簡単な操作（視点の変更，表面の加工，軸の挿入)はメニューバーからできます．さらに

```
>with(plots)
```

で呼び出される plots packageには多くの便利な表示関数が用意されています．その一部の機能については後ほど実例をお見せします．詳しい内容はそれぞれの help を参照ください．

４）方程式の解

solve で方程式の解が求まります.

```
> eqset:={x+y=1,y=1+x^2};
```

$$eqset := \{y = 1 + x^2, x + y = 1\}$$

```
> solve(eqset,{x,y});
```

$$\{x = 0, y = 1\}, \{y = 2, x = -1\}$$

これだけでは

```
> x;y;
```

$$x$$

$$y$$

のように変数 x,y へ代入されていません．これを代入するには

```
> solset:=solve(eqset,{x,y});
```

$$solset := \{x = 0, y = 1\}, \{y = 2, x = -1\}$$

```
> solset[1];
```

$$\{x = 0, y = 1\}$$

```
> assign(solset[1]);
> x;y;
```

$$0$$

$$1$$

のようにassignを使います．解を整数の範囲で求めるisolveというのもあります（実践演習[1]参照）．代数的に解けない方程式に対しても数値的に解く関数(fsolve)がありますが，これについては実践演習[7]で扱っています.

５）式の変形

式の展開や簡単化のために使われるコマンドを以下にまとめて記します.詳しい意味や使用例はヘルプを参照してください．具体的な使用法は実践演習[2]に示しました．”MapleV ラーニングガイド ”の 2.6 数式の変形　に分かりやすい実例が載っています.

expand - Expand an Expression（展開）

simplify - Apply Simplification Rules to an Expression（簡単化，これですべてうまく行くといいのですが...）

collect - Collect Coefficients of Like Powers（同じ項により式をまとめる）

combine - combine terms into a single term（規則にしたがって式をまとめる）

coeff - extract a coefficient of a polynomial（多項式の係数の取り出し）

sort - sort a list of values or a polynomial（式や値のソート）

factor - Factor a Multivariate Polynomial（一つ以上の変数を持つ多項式の因数分解）

normal - normalize a rational expression（分子と分母からなる式の共通因子の削除．分数の結合も行う）

convert - convert an expression to a different form（関数を違う関数へ変換）

gcd - greatest common divisor of polynomials（多項式の最大公約数）

lcm - least common multiple of polynomials（多項式の最小公倍数）

numer - numerator of an expression （分母）

denom - denominator of an expression（分子）

radsimp - simplification of an expression containing radicals（分母の有理化)

assume - The Assume Facility（変数の範囲や関係を規定）

第３章　微積分とその応用

前章の簡単な数式処理の続きで微分・積分とその周辺の題材です．級数は物理や化学の論文でよく使われています．基本的にはうまく解析的に解が出てこないところで，近似として使われます．したがって，どの程度の近似かを常に意識する必要があります．

１）総和の計算

総和は

```
> sum(i,i=1..10);
```

$$55$$

のようにして求まります．以下は高校でお目にかかった公式です．

```
> sum(i^2,i=1..n);
```

$$\frac{1}{3}(n+1)^3-\frac{1}{2}(n+1)^2+\frac{1}{6}n+\frac{1}{6}$$

```
> assume(a<1);
  sum(a^i,i=1..infinity);
  a:='a':
```

$$-\frac{a\sim}{a\sim-1}$$

a は収束するように仮定(assume)して求めています．あとに残らないよう直後に a を default に戻しています．

２）極限

極限の値は limit によってもとまります．

```
> limit(sin(x)/x, x=0);
```

$$1$$

```
> limit(1/x,x=0,complex);
```

$$\infty$$

高校で習った知識がそのまま使えます．

```
> sum(1^i/i!,i=0..infinity);
```

$$\mathbf{e}$$

例えば，以下のような式に単純な代入を行うと

```
> r:=(x^2-1)/((x+1)*(x^2+2));
```

$$r:=\frac{x^2-1}{(x+1)\,(x^2+2)}$$

```
> subs(x=-1,r);
Error, division by zero
```

とエラーを返してきます．これの極限をとると

```
> limit(r,x=-1);
```

$$\frac{-2}{3}$$

つぎに右側極限と左側極限とが違う値に収束している場合は

```
> limit(tan(x),x=Pi/2);
```

$$undefined$$

です．しかし，右と左を指定すると

```
> limit(tan(x),x=Pi/2,right);
  limit(tan(x),x=Pi/2,left);
```

$$-\infty$$

$$\infty$$

とちゃんと求めてくれます．

3）微分

微分は diff によって行います．

```
> diff(x^2,x);
```

$$2\,x$$

```
> diff(y^2*x^2,x,x);
```

$$2\,y^2$$

関数の一般形のままでも形式的な微分を表示してくれます．

```
> c:=(x,t)->X(x)*T(t);
```

$$c := (x,t) \rightarrow \mathrm{X}(x)\,\mathrm{T}(t)$$

```
> diff(c(x,t),t);
```

$$\mathrm{X}(x)\left(\frac{\partial}{\partial t}\mathrm{T}(t)\right)$$

```
> diff(c(x,t),x,x);
```

$$\left(\frac{\partial^2}{\partial x^2}\mathrm{X}(x)\right)\mathrm{T}(t)$$

4）微分方程式

微分方程式は dsolve によって解きます．例えば，

```
> eq:=diff(y(x),x)+y(x)=0;
```

$$eq := \left(\frac{\partial}{\partial x}\mathrm{y}(x)\right) + \mathrm{y}(x) = 0$$

```
> dsolve(eq,y(x));
```

$$\mathrm{y}(x) = \_C1\,\mathbf{e}^{(-x)}$$

と求められます．さらに初期条件を付け加えるときには

```
> dsolve({eq,y(0)=a},y(x));
```

$$\mathrm{y}(x) = a\,\mathbf{e}^{(-x)}$$

として代入します．この他に連立微分方程式，べき級数解(option=series)，数値解(option=numeric)等も求めることができます．連立微分方程式については実践演習[4]に示しました．

5）積分

積分は

```
> int(ln(x),x);
```

$$x\ln(x) - x$$

```
> int(sin(x),x=-Pi..0);
```

-2

などで求めます．積分公式がないと求められないような関数も

```
> eq:=x^2/sqrt(1-x^2);
  int(eq,x);
```

$$eq := \frac{x^2}{\sqrt{1-x^2}}$$

$$-\frac{1}{2}x\sqrt{1-x^2}+\frac{1}{2}\arcsin(x)$$

```
> eq2:=exp(-x^2);
  int(eq2,x=-z..z);
```

$$eq2 := \mathbf{e}^{(-x^2)}$$

$$\operatorname{erf}(z)\sqrt{\pi}$$

という具合です．

6）級数

Mapleで式のTayler級数展開を見てみましょう．位数は最後につけます．例えばある関数を原点の周りで４次まで展開してみましょう．

```
> series(f(x),x=0,4);
```

$$f(0)+D(f)(0)\,x+\frac{1}{2}(D^{(2)})(f)(0)\,x^2+\frac{1}{6}(D^{(3)})(f)(0)\,x^3+O(x^4)$$

この関数を取り出すためにはconvertを使います．

```
> convert(%,polynom);
```

$$f(0)+D(f)(0)\,x+\frac{1}{2}(D^{(2)})(f)(0)\,x^2+\frac{1}{6}(D^{(3)})(f)(0)\,x^3$$

これらの詳しい使い方は実践演習[3]に簡単な例を示しました．Mapleはさらに漸近級数を求めるためにasymptという関数も用意しています．

7）複素関数

多くの関数は複素数を引数としてとることができます．

```
> series(exp(x),x=0.5*I,3);
```

$$(.8775825619+.4794255386\,I)+(.8775825619+.4794255386\,I)\,(x-.5\,I)+$$
$$(.4387912810+.2397127693\,I)\,(x-.5\,I)^2+O((x-.5\,I)^3)$$

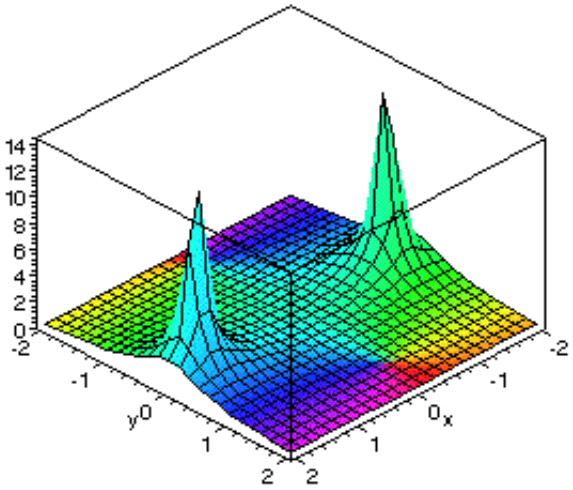
```
> int(exp(x/2),x=1+1*I..0+1*I);
  evalc(%);
```

$$-2\,\mathbf{e}^{\left(\frac{1}{2}\right)}(-1)^{\left(\frac{1}{2}\frac{1}{\pi}\right)}+2\,(-1)^{\left(\frac{1}{2}\frac{1}{\pi}\right)}$$

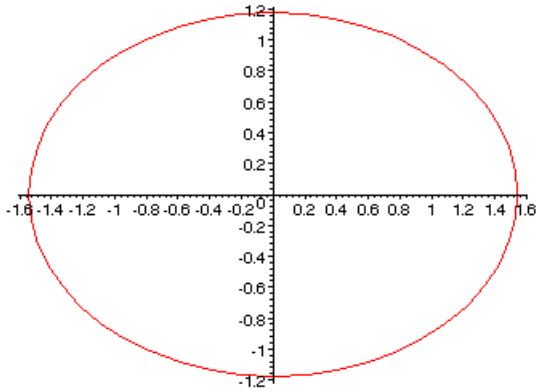
$$-2\,\mathbf{e}^{\left(\frac{1}{2}\right)}\cos\left(\frac{1}{2}\right)+2\cos\left(\frac{1}{2}\right)+I\left(-2\,\mathbf{e}^{\left(\frac{1}{2}\right)}\sin\left(\frac{1}{2}\right)+2\sin\left(\frac{1}{2}\right)\right)$$

さらに複素平面での関数のプロットもお任せでやってくれます．

```
> with(plots):
  complexplot3d( sec(z) , z = -2 - 2*I .. 2 + 2*I );
```




```
> complexplot(sin(x+I),x=-Pi..Pi);
```



などで求めます.

第４章　データ構造と線形代数

Mapleは種々のデータ構造を持っています．これらデータ構造の中で有用ながら混乱しがちな集合，リスト，表，配列について説明します．さらに工学で頻繁に使う行列とベクトルを扱う線形代数のパッケージについて述べます．

１）集合，リスト，表，配列

集合	set	{a,b,c}
リスト	list	[a,b,c]
表	table	
配列	array	

集合　set　{a,b,c}

集合は普通に使われる集合と同じ意味を持ちます. 集合に対する組み込みのオペレーターには以下のものがあります.

FUNCTION: union - set union operator（和）

FUNCTION: intersect - set intersection operator（積）

FUNCTION: minus - set difference operator（差）

```
> set1:={1,2,3};
  set2:={3,4};
```

$$set1 := \{1, 2, 3\}$$

$$set2 := \{3, 4\}$$

```
> set3:=set1 union set2;
```

$$set3 := \{1, 2, 3, 4\}$$

```
> set4:=set1 intersect set2;
```

$$set4 := \{3\}$$

```
> set5:=set1 minus set2;
```

$$set5 := \{1, 2\}$$

集合では値が重複する時には一つだけを残して他は削除されます. 値には基本的に順番がありません.

リスト　list　[a,b,c]

リストには順番があります．また，重複する場合にもそのまま残されます．

ここではリストの操作について例を挙げます．

```
> list1:=[1,2,3];
  list2:=[3,4];
```

$$list1 := [1, 2, 3]$$

$$list2 := [3, 4]$$

この結合をつくるときにはop(list)を使います．この関数はlistの要素からなる部分列を生成します．これを使ってリストの内容を取りだします．

```
> list3:=[op(list1),op(list2)];
```

$$list3 := [1, 2, 3, 3, 4]$$

集合と違って, 要素は全て保存されます. リストに含まれている要素の個数を知りたいときには nops を使います.

```
> n:=nops(list3);
```

$$n := 5$$

要素の内容は list[*index*] で取り出せます.

```
> list3[5];
```

$$4$$

*index*は１から始まります. 全ての要素を表示したいときにはprint(*list*)を使います. またseqを使っても可能です. これを利用して, 逆順に並び替えてみましょう.

```
> list4:=[seq(list3[n-i+1],i=1..n)];
```

$$list4 := [4, 3, 3, 2, 1]$$

並び替えの結果を直接 list3 にするとデータは破壊されます. 従って, 元の名前を使いたいときにはその後で代入をします.

```
> list3:=list4;
```

$$list3 := [4, 3, 3, 2, 1]$$

データの入れ換えは単純に,

```
>list3[2]:=x;
```

$$list3_2 := x$$

です. 全部を表示してみると

```
> list3;
```

$$[4, x, 3, 2, 1]$$

表 table

Mapleでは表形式のデータ構造を取ることができます. その場合*index(subscript)*は名前なども使えます. 表を作るときには table 関数を用いて以下のようにします.

```
> atomicweight:=table([Fe=56,C=12]);
```

$$atomicweight := \mathrm{table}([Fe = 56, C = 12])$$

```
> atomicweight[Fe];
```

$$56$$

表の変更は

```
> atomicweight[Fe]:=55.847;
```

$$atomicweight_{Fe} := 55.847$$

で行います. また追加は

```
> atomicweight[Al]:=27;
```

$$atomicweight_{Al} := 27$$

です. 内容の表示は print で行います.

```
> print(atomicweight);
```

$$\mathrm{table}([Al = 27, Fe = 55.847, C = 12])$$

削除は

```
> atomicweight[C]:='atomicweight[C]';
```

$$atomicweight_C := atomicweight_C$$

```
> print(atomicweight);
```

```
table([
    Al = 27
    Fe = 55.847
    ])
```

です．（”ちょん”はここでは普通のシングルクォート"'"を使います．バッククォート"`"ではありません）

配列　array

配列は表の subscript（添字)を整数に限定したものです．これは行列やベクトルとして使えます．表の場合の table と同じように配列では array を用います．ただし，subscript の範囲を明示する必要があります．ベクトルはarray(1..n)で配列はarray(1..m,1..n)として指定します．特別の関係を持った行列は以下のようにして定義することができます．

```
> array(identity,1..3,1..3);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> array(sparse,1..3,1..3);
```

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

```
> array(1..2,1..2,[[1,2],[3,4]]);
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> A:=array(antisymmetric,1..3,1..3);
  A[1,2]:=1;A[1,3]:=2;A[2,3]:=3;
  print(A);
```

$$A := \mathrm{array}(antisymmetric, 1 .. 3, 1 .. 3, [\,])$$

$$A_{1,2} := 1$$

$$A_{1,3} := 2$$

$$A_{2,3} := 3$$

$$\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 3 \\ -2 & -3 & 0 \end{bmatrix}$$

構造の確認，構造変換と関数の適用

type

前に作ったデータがどういう構造を持っているのか分からなくなってしまうときがあります．例えば，リストなのか，配列なのかという場合です．このようなときには

```
> type(A,array);
```

$$true$$

などで確認することができます．返答は true か false です．

convert

集合，リスト，表，配列の相互のデータ構造間の構造変換も行ってくれます．以下では配列をリストのリスト(listlist)に変換しています．

```
> convert(A,listlist);
```

$$[[0,1,2],[-1,0,3],[-2,-3,0]]$$

map

全ての要素について関数を適用するような場合には

```
> map(sin,A);
```

$$\begin{bmatrix} 0 & \sin(1) & \sin(2) \\ -\sin(1) & 0 & \sin(3) \\ -\sin(2) & -\sin(3) & 0 \end{bmatrix}$$

のように map によっておこないます．

２）線形代数

Mapleの標準関数を使って線形代数(linear algebra)を適用することは可能ですが面倒です．普通はMaple ライブラリーに用意されている package を使います．コマンド with(linalg)によってライブラリーをコールします．これによって行列やベクトルがより直感的に定義でき，演算や操作が容易に行えます．

```
> with(linalg):
> ma:=matrix(A);
```

$$ma := \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 3 \\ -2 & -3 & 0 \end{bmatrix}$$

```
> vec1:=vector([x,y,z]);
```

$$vec1 := [x, y, z]$$

等です．ここで vector は縦（列）ベクトル(column vector)を生成することに注意ください．行列の転置を行う演算 transpose(vec)を行うと横（行）ベクトル(row vector)となります．（英語で座席は row で，新聞の囲み記事は column です）．

標準で用意されている配列に対する演算は evalm で，足し算，内積，行列のべき乗，スカラー乗算等の基本的な演算のみが行えます．覚えやすいので便利です．例えば，

```
> evalm(ma^2);
```

$$\begin{bmatrix} -5 & -6 & 3 \\ -6 & -10 & -2 \\ 3 & -2 & -13 \end{bmatrix}$$

```
> evalm(ma &* vec);
```

$$[y+2z,-x+3z,-2x-3y]$$

これだけでは余り多くのことができません．さらに多くの高機能な演算が linalg package に用意されています．使い方は以下の通りです．matrix計算の時には次元が整合していなければなりません．

```
> multiply(vec1,vec1);
  multiply(transpose(vec1),vec1);
  multiply(vec1,transpose(vec1));
```

$$x^2+y^2+z^2$$

$$x^2+y^2+z^2$$

$$\begin{bmatrix} x^2 & xy & xz \\ xy & y^2 & yz \\ xz & yz & z^2 \end{bmatrix}$$

全ての演算がwith(linalg)を実行したときに表示されますので，関連している項目のヘルプを参照してください．以下に主に使う演算を示します．

linalg[matadd] - matrix or vector addition（和）(matadd(A, B, c1, c2):c1*A+c2*B)
linalg[dotprod] - vector dot (scalar) product（内積）
linalg[crossprod] - vector cross product（外積）
linalg[inverse] - compute the inverse of a matrix（逆行列）
linalg[det] - determinant of a matrix（行列式）
linalg[trace] - the trace of a matrix（対角和）
linalg[adjoint] - compute the adjoint of a matrix（随伴）
linalg[transpose] - compute the transpose of a matrix（転置行列）
linalg[eigenvals] - compute the eigenvalues of a matrix（固有値）
linalg[eigenvects] - find the eigenvectors of a matrix（固有ベクトル）
linalg[angle] - compute the angle between vectors（角度）
linalg[rowdim] - determine the row dimension of a matrix（行の数）
linalg[coldim] - determine the column dimension of a matrix（列の数）
linalg[vectdim] - determine the dimension of a vector（ベクトルの要素数）

この他にもcurl,diverge,grad,jacobianなどの微分演算子や，行列やベクトルの行や列を操作するオペレーションが用意されています．使用例は第６章４）と実践演習[5][6]に示しました．

第５章　MapleV でのプログラミング

１）プログラミングの流れ

ある決まった操作をするときや関数が複雑になってきた場合にプログラミングが必要になってきます．Maple では BASIC や C などの一般的なプログラム言語より圧倒的に楽にプログラミングができます．このプログラミングの便利さは，ちょっとしたルーチン計算のために使えるだけでなく，本格的なプログラムを書く前にそのひな型を検討する際にも多いに役立ちます．

では，Maple ではプログラムはどのようにして書くのでしょうか？その前にプログラミングの一般的な注意を述べておきます．

１．本当にプログラミングが必要か？

いくらプログラミングが楽だといってもやはりプログラミングには時間がかかります．従って，先ず本当にその処理や関数をわざわざ書く必要があるのかを考えてください．『あるものは使え！』が原則です．Maple で提供されている多くの関数の中に使える関数があるかを先ず探してみてください．

”ライトついてますかー問題発見の人間学”

ドナルド・C・ゴース，ジェラルド・M・ワインバーグ著（共立出版１９８７）．

２．プログラムの作成

プログラムが必要であるという結論に達したときにはそれを実際に作っていくしかありません．この作業の手順は普通のプログラムと全く同じです．

入力と出力の決定

プログラミング（本当の）

デバッグ

となるでしょう．本格的に問題を解くときにはアルゴリズムやデータ構造を考える必要がありますが，我々が解く必要がある問題はほとんどの場合単純です．従って，素直に手でやる計算を自動化するだけで出来上がります．下手にテクニックを考えるよりも何をしているのかが見るだけで理解できるプログラムを書くように心掛けてください．

３．汎用性？

さらに汎用性や，多くの人があるいは何度も使う可能性がある場合には ERROR,type check, ヘルプファイル，ユーザライブラリの作成のような作業を行う必要があります．

２）Maple の制御構造

プログラムでの構造は以下の３種類だけで，あとはこれらの組み合わせです．

if　　場合分け

for　　決まった分だけぐるぐる

while　あるところまでぐるぐる

この構文を以下に示します．

```
if condition1 then
  statement sequence 1
elif condition2 then
```

```
   statement sequence 2
else
   default statement sequence
fi;

for i from start by change to finish
do
   statement sequence
od;

while condition
do
   statement sequence
od;
```

next,break

forやwhile-loopの途中で，処理を省きたいときやloopから脱出したいときにはnext,breakをそれぞれ使います．構文は

```
if condition next;
```
odまで跳んで次のloopへ

```
> for i from 1 to 10 do
    if (modp(i,2)=0) then next fi;
    printf("%d, "`,i);
  od;
```

1, 3, 5, 7, 9,

```
if condition break;
```
一番近いodの外へ出てloopを脱出

```
> for i from 1 to 10 do
    if (i>5) then break fi;
    printf(`%d, `,i);
  od;
```

1, 2, 3, 4, 5,

ひな型

procedure（手続関数）のひな型です．

```
name := proc(input sequences)
local variable sequence;
global variable sequence;
options option sequence;
description descripiton sequence;
statement 1;
:
:
statement n;
end;
```

procedureからの戻り値は最後のstatementからの出力です．

3）プログラミングの実践

それでは学生の成績データの平均と分散を求めるprocedureを題材に実際のプログラミングの流れを見てみましょう．先ずデータを用意します．

```
> list1:=[48,56,68,72];
```

list1 := [48, 56, 68, 72]

次にひな型の作成とデータがうまく読み込まれているかの確認をしておきます．改行はshift+enterでおこないます．

```
> stat:=proc(data1)
  print(data1);
  end;
```

stat := proc(data1) print(data1) end

```
> stat(list1);
```

[48, 56, 68, 72]

次に実際の処理がうまく動くか確かめておきます. (脚注参照)

```
> n:=nops(list1);
  ave:=add(list1[i],i=1..n)/n;
  disp:=0;
  for i from 1 to n do
    disp:=disp+(list1[i]-ave)^2;
  od;
  disp:=disp/n;
```

$$n := 4$$
$$ave := 61$$
$$disp := 0$$
$$disp := 169$$
$$disp := 194$$
$$disp := 243$$
$$disp := 364$$
$$disp := 91$$

出力が多すぎる時はod;をod:とセミコロンをコロンに変えることで抑制することができます. これらを先ほど作ったひな型に加え local 変数を宣言しておきます. local variables は proc 内でのみ使われる変数です. proc内で外部の変数を参照することも可能です. 同じ名前が使われているときには内部変数が優先されます.

```
> stat:=proc(data1::list)
  local n,ave,disp,i;
  if nargs=0 then ERROR("no argument") fi;
  n:=nops(list1);
  ave:=add(list1[i],i=1..n)/n;
  disp:=0;
  for i from 1 to n do
    disp:=disp+(list1[i]-ave)^2;
  od:
  disp:=disp/n;
  printf("Average      =%10.5f\n",ave);
  printf("Dispersion   =%10.5f\n",disp);
  printf("Standard mean=%10.5f\n",evalf(sqrt(disp)));
  end;
  stat(list1);
```

```
stat := proc(data1::list)
local n, ave, disp, i;
    if nargs = 0 then ERROR("no argument") fi;
    n := nops(list1);
    ave := add(list1[i], i = 1 .. n) / n;
    disp := 0;
    for i to n do disp := disp + (list1[i] - ave)^2 od;
    disp := disp / n;
    printf("Average      =%10.5f\n", ave);
    printf("Dispersion   =%10.5f\n", disp);
    printf("Standard mean=%10.5f\n", evalf(sqrt(disp)))
end
```

```
> stat(list1);
Average      =  61.00000
Dispersion   =  91.00000
Standard mean=   9.53939
```

---

for を使った総和は教育的なものです．実際の数値を足しあわせるときには例で示してあるような，add 関数を使ったほうが，はるかに効率が良いようです．

```
Dispersion   =  91.00000
Standard mean=   9.53939
```

ここで，出力の見栄えをよくするためにprintf文を使っています．書式はC言語でおなじみのものですが，慣れない人はhelpを参照してください．文字列はダブルクォートで囲まれています．またエラー処理を追加しています．さらに最初の引数をdata1::listとすることによって，data1のtypeがlistであるかどうかをMapleが判断してくれます．

上のプログラム中にnargsという変数が出てきています．proc内には特別の意味を持ったnargsとargsというのがあります．これまたC言語ではおなじみです．nargsはprocがいくつのinputを受け取ったかを表しています．それぞれのinputはargs[i]によって取りだすことが可能です．もうひとつproc内で重要な特殊名procnameがあり，これはプロシージャの名前が割り当てられています．

もうひとつの重要なコマンドはRETURNで，それ以降の計算が必要ないときに戻り値を持ってprocedureを抜ける関数です．これを使った再帰的な関数を定義してみましょう．近似で有用なChebyshev多項式です．これを漸化式で表すと

$$T_n(x) = 2\,x\,T_{n-1}(x) - T_{n-2}(x) \qquad \text{for n>= 2}$$

で，$T_0(x)$=1と$T_1(x)$=xです．これをMapleで表すと，

```
> T:=proc(n::nonnegint,x::name)
  #Chebyshev polynomials
  option remember;
  if n=0 then RETURN(1);
  elif n=1 then RETURN(x);
  fi;
  2*x*T(n-1,x)-T(n-2,x);
  end;
```

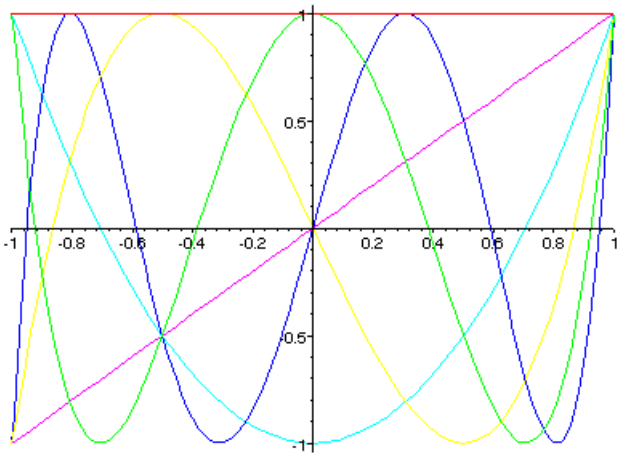
$$T := \mathbf{proc}(n::nonnegint, x::name)$$
$$\quad \mathbf{option}\ remember;$$
$$\quad\quad \mathbf{if}\ n = 0\ \mathbf{then}\ \mathrm{RETURN}(1)\ \mathbf{elif}\ n = 1\ \mathbf{then}\ \mathrm{RETURN}(x)\ \mathbf{fi};$$
$$\quad\quad 2{*}x{*}\mathrm{T}(n-1,x) - \mathrm{T}(n-2,x)$$
$$\quad \mathbf{end}$$

となります．option rememberはremember tableに結果を保存させて計算速度を上げるためにつけてあります．#はコメントに使っています．この結果をplotさせてみます．まず，5次のChebyshev多項式までを関数として定義します．'.'(ピリオド)は連結作用素で二つの要素を連結してくれます．

```
> for i from 0 to 5 do
    f.i:=unapply(expand(T(i,x)),x);
  od;
```

$$f0 := 1$$
$$f1 := x \to x$$
$$f2 := x \to 2\,x^2 - 1$$
$$f3 := x \to 4\,x^3 - 3\,x$$
$$f4 := x \to 8\,x^4 - 8\,x^2 + 1$$
$$f5 := x \to 16\,x^5 - 20\,x^3 + 5\,x$$

```
> plot({f0,f1,f2,f3,f4,f5},-1..1);
```



この他にいくつもの有益なテクニックと優れた実例が“ MapleV Programming Guide ”に掲載されています．一読されることをお勧めします．

４）その他のテクニック

debug に関する注意

プログラムにはバグが付き物です．バグ取りにはprintlevel, trace, mint, profile等を使います．この中で一番簡単なのは計算途中の出力の量を変えてくれる，printlevel の設定です．これは

```
> printlevel :=11;
```

として，設定します．元に戻すには printlevel:=1 としてください．

traceは特定のプロシージャの結果だけを調べます．mintはmapleとは別のプログラムとして用意され syntax のチェックなどをしてくれ，本格的なプログラムを作るときなどに便利です．（付録 B で使用例を紹介しています）．さらに実行速度が問題になるときには profile を使います．

その他，help file,user library の作り方も含めて help あるいは “ MapleV Programming Guide ” を参照してください．

定義関数の plot に関する注意

例えば

filter(x) 1-x/10 for x<10

0 for x>=10

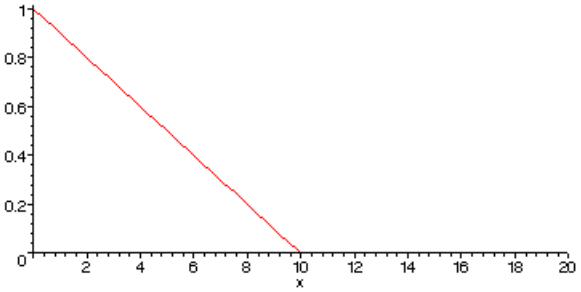
という関数を考えてみましょう．これは

```
> filter:=x->if x<10 then 1-x/10 else 0 fi;
```

filter := **proc**(*x*) **option** *operator*, *arrow*; **if** *x* < 10 **then** 1 – 1 / 10*x* **else** 0 **fi end**

となります．うまく定義できているか確認しておきましょう．

```
> plot('filter(x)','x'=0..20);
```



ここで，自分で定義した関数のplotを行うときにはシングルクォートで囲む必要があることに注意ください．plotでは初めに引数がチェックされます．したがって，ユーザー定義関数では

引数が名前であるために if 文などでうまく処理されず error が返ってきます．これを回避するにはシングルクォートを使って引数の評価を行わずに plot コマンドへ渡すことで解決できます．

第６章　データ処理

Maple の応用として，有用なデータ処理に関連して，１）データの読み込み，２）フーリエ変換，３）畳み込み，そして４）非線形の最小二乗法について解説し，具体的な操作を見ていきます．

（参考　河合潤：『化学計測学』 合志陽一編　昭晃堂　1997,

より高度な Filtering や Fitting の記述が，W.Gander and U.von Matt, "Smoothing Filters", Solving Problems in Scientific Computing Using Maple and MATLAB, Walter Gander and Jiri Hrebicek, Springer 1991, Chapter 9. にあります．ただし MATLAB のスクリプトです)

１）データの入出力

手っ取り早くデータを読み込むには readdata で行います(脚注)．

```
> T:=readdata(`bob's pb1400:DATA101`,1):
```

コロンで出力を抑制しています．ここで，最初の引数がファイルの名前で，次がデータの列の数です．ファイル名は特殊な記号（例えばスラッシュ／）が入ったときにはバッククォート（｀）で囲っておく必要があります．パス名をちゃんと書かなかった場合は PC では Maple の入っているフォルダーだけを，unix では Maple を起動したディレクトリーだけを探します．ちゃんと読めているか見ておきましょう．

```
> T[1];
  n:=nops(T);
```

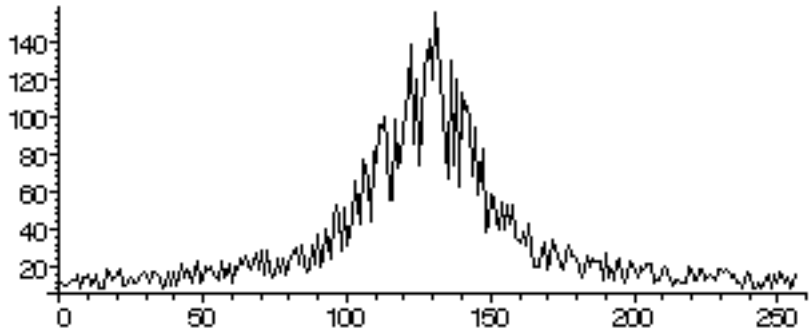
$$11.70159307$$

$$n := 256$$

となっています．浮動小数点のデータも自動的に読まれています．ただし，コメントなどが入っているデータは stats[importdata]で行ってください（詳しくはヘルプを参照)．

次にこのデータが正しく読み込まれているかをチェックしましょう．一個一個確かめるのも手ですが全体を表示させてその形で判断してみましょう．最も簡単には

```
> with(plots):
> listplot(T);
```



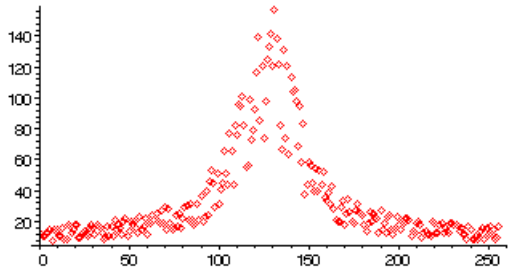
注) データは例えば，

```
> f1:=t->subs({a=10,b=40000,c=380,d=128},a+b/(c+(t-d)^2) );
  T:=[seq(f1(i)*(0.6+0.8*evalf(rand()/10^12)),i=1..256)]:
  writedata(`bob's pb1400:DATA101`,convert(T,list));
```

によって生成，ファイルに保存することができます．

でできます．うまく読めているようです．ただこの関数ではあまり自由に加工できません．以下では点の座標の並び(listlist)を使って出力する方法を示しておきます．

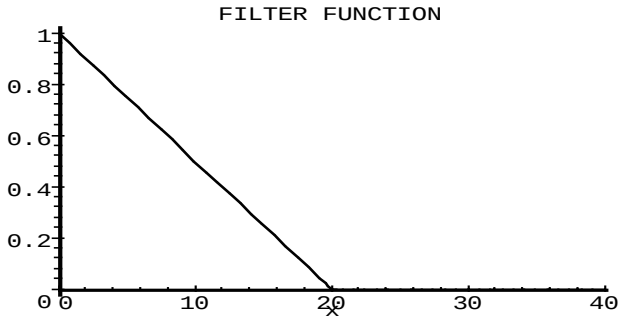
```
> datapoint:=[seq([i,T[i]],i=1..256)]:
  plot(datapoint,style=POINT);
```



2）フーリエ変換

フーリエ変換を用いたデータのフィルタリングについての例です．MapleではFFT(高速フーリエ変換）のルーチンが用意されています．詳しくは help を参照下さい．

演習： 与えられた256個のデータ（上図：ファイル名DATA101）をフーリエ変換し，高周波成分を下図の台形フィルタを用いて取り除き，フーリエ逆変換せよ．上図のノイズ成分はどうなったか．振幅強度$\sqrt{a_n^2+b_n^2}$ をプロットせよ(128チャンネル)．はじめのデータの面積強度とフーリエ変換で求めたデータの面積強度とはどういう関係があるのか．



解説： 時系列データ$x(t)$ の解析にはそのフーリエ変換

$$X(f)=\int_{-\infty}^{\infty} x(t)\exp\left(2\pi i f t\right)dt$$

が最も多用されます．これは，高速フーリエ変換アルゴリズムの発明によって計算機での処理が容易となったことが大きく効いています．時間経過と共に変化する時系列測定データ$x(t)$ に含まれるさまざまな周波数成分のすべてについて，何 Hz の周波数成分がどの程度の割合でその時系列データの中に含まれているのかを明らかにできるからです．時系列データの中のノイズはδ関数に似ており，そのフーリエ変換は全周波数成分を一様に含んでいます．一方，測定したいデータは一般に，時間に対してゆっくり変化する量です．測定によって得られたデータの時間軸は連続ではなくとびとびの値をとるので，離散フーリエ変換を取ることになります．この例では時間データは256チャンネルのデータです．高周波成分（= ノイズ成分）を取り除くために，データに対して窓関数（低域通過フィルタ）をかけてフーリエ逆変換

$$x(t)=\int_{-\infty}^{\infty} X(f)\exp\left(-2\pi i f t\right)df$$

することによってフィルターを掛けた後のスムーズなカーブが得られます. 低域通過フィルタには，方形窓，修正方形窓，フォンハン窓（２乗余弦窓），ハミング窓（平たん部のある２乗余弦窓）などさまざまなものがあります.

実践：　具体的に計算を見ていきましょう.

```
> T:=readdata(`bob's pb1400:DATA101`,1):
  Idata:=array([seq(0,i=1..256)]):
  Rdata:=convert(T,array):
```

全パワーを求めておきましょう.

```
> temp:=add(i^2,i=T);
```

$$temp := 534310.3367$$

実際に FFT を実行します.

```
> readlib(FFT);
```

$$\mathbf{proc}(m, x, y) \ \ldots \ \mathbf{end}$$

```
> FFT(8,Rdata,Idata);
```

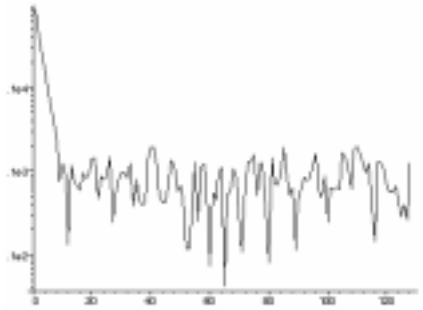
$$256$$

これだけで終わりです. ここでFFTの第一引数の８はデータの個数が2^8であることを示しています. フーリエ変換された後のデータ構造を見ておきましょう.

```
> printf("%3d %15.5f %15.5f\n",1,Rdata[1],Idata[1]);
  for i from 2 to 128 do
    printf("%3d %15.5f %15.5f %15.5f %15.5f\n",
       i,Rdata[i],Idata[i],Rdata[258-i],Idata[258-i]);
  od;
  1      8499.54360        0.00000
  2     -4162.42568      -81.92030     -4162.42568       81.92031
  3      2602.79575      120.20850      2602.79575     -120.20850
  4     -1772.62797      -85.42422     -1772.62797       85.42423
  5      1061.13706       59.90167      1061.13706      -59.90167
  6      -586.57228      -14.66303      -586.57228       14.66303
  7       235.82976     -145.00230       235.82976      145.00230
                              .....
124       187.72794       72.93267       187.72794      -72.93267
125      -160.89158     -128.35496      -160.89158      128.35496
126        91.48342       49.80754        91.48342      -49.80754
127        36.31291      -82.64862        36.31291       82.64862
128        53.48855        6.15179        53.48855       -6.15179
```

波数空間での強度をログで見てみましょう.

```
> Adata:=[seq([i,sqrt(Idata[i]^2+Rdata[i]^2)],i=1..128)]:
  with(plots):
  logplot(Adata);
```



離散的な形の Parseval の定理が成り立っていることも確認できます.

```
> add(Rdata[i]^2+Idata[i]^2,i=1..256)/256;
```

$$534310.3324$$

先程求めた実空間での全パワーの結果と一致しています.

次にフィルター関数を作りましょう.

```
> filter:=x-> piecewise(x>=0 and x<=20,(1-x/20) );
```

$$filter := x \rightarrow \text{piecewise}\left(0 \le x \text{ and } x \le 20, 1 - \frac{1}{20}x\right)$$

```
> plot(filter(x),x=0..40,title="FILTER FUNCTION");
```

この結果が本節の初めに示したフィルター関数の図です. フィルターを通したデータを作ります.

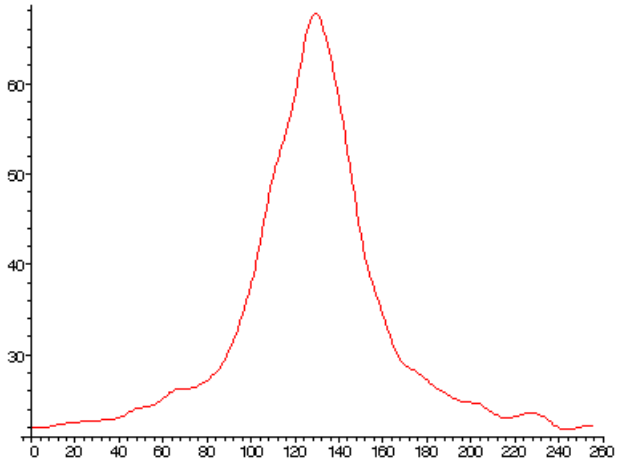
```
> FRdata:=array([seq(Rdata[i]*filter(i),i=1..256)]):
> FIdata:=array([seq(Idata[i]*filter(i),i=1..256)]):
```

逆フーリエ変換を実行すると,

```
> iFFT(8,FRdata,FIdata);
```

256

```
> Adata:=[seq([i,FRdata[i]],i=1..256)]:
> plot(Adata);
```



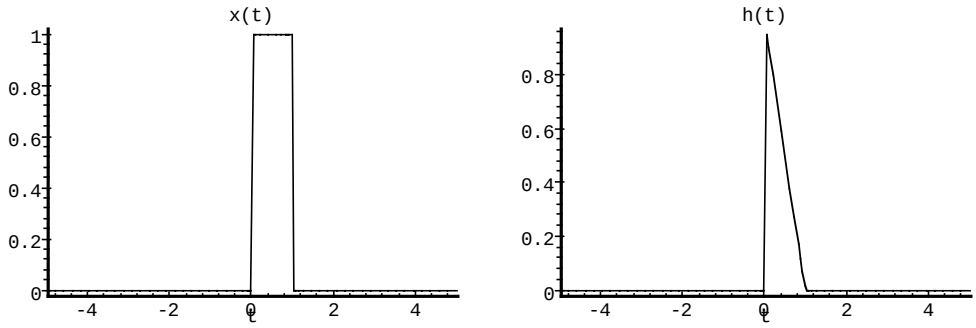
ノイズが取り除かれているのが分かるでしょう.

3) 畳み込み

波長t の真のスペクトル$x(t)$ が下図左パネルのようなかたちをしている. 分光器の装置関数が下図右パネルの$h(t)$ で表されるとき, 観測されるスペクトル

$$f(t) = \int_{-\infty}^{\infty} x(t-\tau)h(\tau)\,d\tau$$

はどのような形状となるか.



解説:　時刻t の観測値$x(t)$ を観測しようとするとき，そのt の瞬間のみではなく，t の過去と未来が重み$h(t)$ で見えることがあります．このとき実測されるスペクトルは$x(t)$ と$h(t)$ の畳み込み(convolution),

$$f(t)=\int_{-\infty}^{\infty} x(t-\tau)h(\tau)\,d\tau$$

となります．スリットを持つ分光器によって，ある波長t のスペクトル強度$x(t)$ を観測するとき，分光器はその前後の波長$t-\tau$ の光も重み$h(t-\tau)$ で見ていることを意味しています．畳み込みを間単に$f=x*h$ と表すと，そのフーリエ変換は

$$F(x*h)=F(x)\,F(h)$$

となります．すなわちコンボリューションのフーリエ変換は，フーリエ変換の積になるわけです．目的のスペクトル$x(t)$ を求めるためには，わり算$F(x*h)/F(h)$ の結果をフーリエ逆変換すればよいことがわかります．コンボリューションの形の積分方程式を解くより，わり算の方が計算上簡単な操作ですむ場合が多く，これをデコンボリューションといいます．

実践：

```
> fun:=x->piecewise(x>=0 and x<=1,1);
  h:=x->piecewise(x>=0 and x<=1, 1-x);
```

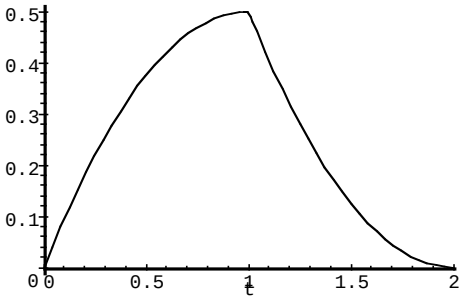
$$fun := x \to \text{piecewise}(0 \le x \ \textbf{and}\ x \le 1, 1)$$

$$h := x \to \text{piecewise}(0 \le x \ \textbf{and}\ x \le 1, 1-x)$$

```
> h:=x->(1-x)*Heaviside(x)*Heaviside(1-x);
```

$$h := x \to (1-x)\,\text{Heaviside}(x)\,\text{Heaviside}(1-x)$$

```
> plot(int(fun(x)*h(t-x),x=-infinity..infinity),t=0..2);
```



４）非線形最小二乗法

非線形の最小二乗法を用いてデータフィットをしてくれる関数は残念ながらMapleでは用意されていないようです(Don't ask me why.)．そこで，実際に非線形最小二乗法のプログラムを作成し，実際のデータへのフィッティングを試みてみましょう．

問題：　与えられたデータ(ファイル名'DATA101'）を，

$$f(t)=a+\frac{b}{c+(t-d)^2} \qquad (6.1)$$

なるローレンツ型関数にカーブフィッティングするプログラムを作成し，そのパラメータを決定

せよ．ここで a はバックグラウンドの強度，b はローレンツ関数の強度，c は線幅，d はピーク位置を表す．

解説： (6.1)式の特徴は，パラメータが線形関係にないということですが，以下では線形近似に基づいた反復解法を用います．非線形最小二乗法の注意事項は補足に記しました．

今，パラメータの初期値を$(a_0+\Delta a_1, b_0+\Delta b_1, c_0+\Delta c_1, d_0+\Delta d_1)$ としましょう．このとき関数fを真値(a_0, b_0, c_0, d_0) のまわりでテイラー展開し高次項を無視すると，

$$\begin{aligned}\Delta f &= f(a_0+\Delta a_1, b_0+\Delta b_1, c_0+\Delta c_1, d_0+\Delta d_1) - f(a_0, b_0, c_0, d_0)\\ &= \frac{\partial f(a_0, b_0, c_0, d_0)}{\partial a}\Delta a_1 + \frac{\partial f(a_0, b_0, c_0, d_0)}{\partial b}\Delta b_1\\ &+ \frac{\partial f(a_0, b_0, c_0, d_0)}{\partial c}\Delta c_1 + \frac{\partial f(a_0, b_0, c_0, d_0)}{\partial d}\Delta d_1\end{aligned} \qquad (6.2)$$

となります．(6.2)式の偏微分は計算可能です．'DATA101' のデータはt=1から256までの時刻に対応したデータ点$f_1 \sim f_{256}$ とします．各測定値とモデル関数から予想される値との差$\Delta f_1 \sim \Delta f_{256}$ は，

$$\begin{pmatrix}\Delta f_1\\ \Delta f_2\\ \vdots\\ \Delta f_{256}\end{pmatrix} = \boldsymbol{J}\begin{pmatrix}\Delta a_1\\ \Delta b_1\\ \Delta c_1\\ \Delta d_1\end{pmatrix} \qquad (6.3)$$

となります．ここで4行256列の行列

$$J = \begin{pmatrix}\left(\frac{\partial f}{\partial a}\right)_1 & \left(\frac{\partial f}{\partial b}\right)_1 & \left(\frac{\partial f}{\partial c}\right)_1 & \left(\frac{\partial f}{\partial d}\right)_1\\ \vdots & \vdots & \vdots & \vdots\\ \left(\frac{\partial f}{\partial a}\right)_{256} & \left(\frac{\partial f}{\partial b}\right)_{256} & \left(\frac{\partial f}{\partial c}\right)_{256} & \left(\frac{\partial f}{\partial d}\right)_{256}\end{pmatrix} \qquad (6.4)$$

はヤコビ行列と呼ばれる行列です．逆行列J^{-1}は転置行列J^tを用いて

$$\boldsymbol{J}^{-1} = \left(\boldsymbol{J}^t\boldsymbol{J}\right)^{-1}\boldsymbol{J}^t \qquad (6.5)$$

と表されます．従って真値からのずれは

$$\begin{pmatrix}\Delta a_2\\ \Delta b_2\\ \Delta c_2\\ \Delta d_2\end{pmatrix} = \left(\boldsymbol{J}^t\boldsymbol{J}\right)^{-1}\boldsymbol{J}^t\begin{pmatrix}\Delta f_1\\ \Delta f_2\\ \vdots\\ \Delta f_{256}\end{pmatrix} \qquad (6.6)$$

として計算できます．理想的には$(\Delta a_2, \Delta b_2, \Delta c_2, \Delta d_2)$ は$(\Delta a, \Delta b, \Delta c, \Delta d)$ に一致するはずですが，測定誤差と高次項のために一致しません．初期値に比べ，より真値に近づくだけです．そこで，新たに得られたパラメータの組を新たな初期値に用いて，より良いパラメータに近付けていくという操作を繰り返します．新たに得られたパラメータと前のパラメータの差がなくなったところで計算を打ち切り，フィッティングの終了となります．

実践： 実際にパラメータフィットを行っていきましょう．サブパッケージとして線形代数計算のためにlinalgを呼びだしておきます．

```
> with(linalg):
```

```
Warning, new definition for norm
Warning, new definition for trace
```

データを読み込みます.

```
> T:=readdata(`bob's pb1400:DATA101`,1):
  datapoint:=[seq([i,T[i]],i=1..256)]:
```

(6.1)式のローレンツ型の関数を仮定しておきましょう.

```
> f:=a+b/(c+(x-d)^2);
  f1:=unapply(f,x);
```

$$f := a + \frac{b}{c + (x - d)^2}$$

$$f1 := x \to a + \frac{b}{c + (x - d)^2}$$

(6.2)式の関数の微分を求め，新たな関数として定義します.

```
> dfda:=unapply(diff(f,a),x);
  dfdb:=unapply(diff(f,b),x);
  dfdc:=unapply(diff(f,c),x);
  dfdd:=unapply(diff(f,d),x);
```

$$dfda := 1$$

$$dfdb := \frac{1}{x \to c + (x - d)^2}$$

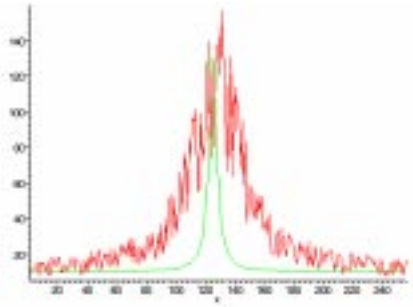
$$dfdc := x \to -\frac{b}{\left(c + (x - d)^2\right)^2}$$

$$dfdd := x \to -\frac{b\,(-2\,x + 2\,d)}{\left(c + (x - d)^2\right)^2}$$

初期値を仮定します.

```
> guess1:={a=10,b=1200,c=10,d=125};
  plot({datapoint,subs(guess1,f1(x))},x=1..256);
```

$$guess1 := \{a = 10, b = 1200, d = 125, c = 10\}$$



(6.3)式の左辺のΔf_i を求めます．データの出力を抑制するために od の後はセミコロンではなくコロンにしてあります．デバッグや慣れないときには出力を多めに，データ数を少なめにして，関数の内側から順番に内容を確認しながら打ち込んで下さい.

```
> imax:=nops(T);
  df:=[seq(evalf(subs(guess1,T[i]-f1(i))),i=1..imax)]:
```

$$imax := 256$$

次に(6.4)式に従ってヤコビ行列を求めます.

```
> Jac:=array(sparse,1..imax,1..4);
  for i from 1 to imax do
    Jac[i,1]:=evalf(subs(guess1,dfda(i)));
    Jac[i,2]:=evalf(subs(guess1,dfdb(i)));
    Jac[i,3]:=evalf(subs(guess1,dfdc(i)));
    Jac[i,4]:=evalf(subs(guess1,dfdd(i)));
  od:
```

$$Jac := \mathrm{array}(sparse, 1 .. 256, 1 .. 4, [\ ])$$

(6.5)式の公式によるヤコビ行列の逆行列の計算です.

```
> IJac:=inverse(multiply(transpose(Jac),Jac)):
  tJac:=multiply(IJac,transpose(Jac)):
```

最後に(6.6)式により，新たなパラメータの組を求めます.

```
> guess2:=multiply(tJac,df);
```

$$guess2 := [12.69373250, 4132.316870, 42.34612280, .6167914619]$$

```
> guess3:={a=subs(guess1,a)+guess2[1],
    b=subs(guess1,b)+guess2[2],
    c=subs(guess1,c)+guess2[3],
    d=subs(guess1,d)+guess2[4]};
```

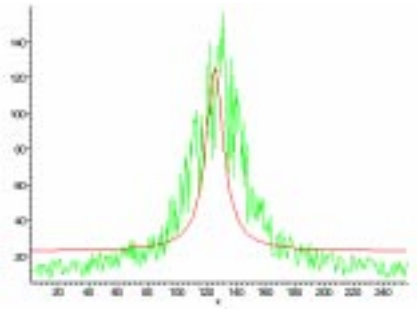
$$guess3 := \{c = 52.34612280, b = 5332.316870, d = 125.6167915, a = 22.69373250\}$$

```
> guess1:=guess3;
```

$$guess1 := \{c = 52.34612280, b = 5332.316870, d = 125.6167915, a = 22.69373250\}$$

求めたパラメータを用いたモデル関数と，データをプロットしてみます. 前回より近づいているのがわかるでしょう.

```
> plot({datapoint,subs(guess1,f1(x))},x=1..256);
```

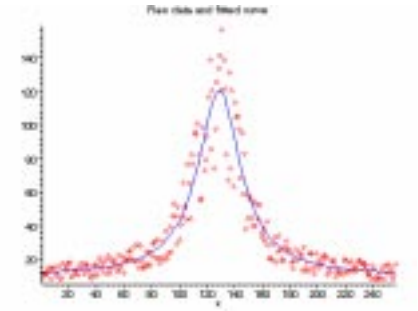


Δf_i 以下の操作を4回繰り返した後の結果です. データにほぼフィットした結果が得られています. 見やすくするために出力を少し工夫してあります.

```
> print(guess1);
```

$$\{c = 353.0702877, d = 128.7247646, a = 10.00417520, b = 39094.33977\}$$

```
> with(plots):
  F2:=plot({datapoint},x=1..256,color=red,style=point):
  F1:=plot(subs(guess1,f1(x)),x=1..256,style=line,color=blue,thickness=1):
  display({F1,F2},title="Raw data and fitted curve");
```



補足：最小二乗法に関するメモ

前述のフィッティング法の説明では，テイラー展開を用いた説明であり，まるで最小二乗法を用いてないような印象を与えたかもしれません．しかしこれは，最小二乗法の χ^2 関数に Newton-Raphson 法を適用し，二次微分を無視した方法と考えることも可能です．この様子を以下に記しておきます．

当てはめたいモデルは $\boldsymbol{x}$ がデータの横軸，$\boldsymbol{a}$ がパラメータの組とすると，

$$\boldsymbol{y} = \mathrm{f}(\boldsymbol{x};\boldsymbol{a})$$

です．最小二乗法の χ^2 関数は

$$\chi^2(\boldsymbol{a}) = S(\boldsymbol{a}) = \sum_{i=1}^{N} \left[y_i - f(x_i;\boldsymbol{a}) \right]^2$$

です. 後の式を単純にするために $y_i - f(x_i;\boldsymbol{a}) \Rightarrow f(x_i;\boldsymbol{a})$ と置き換え, あらかじめ実験値を引いておきます．S が $\boldsymbol{a}$ で極小値を持つ(安定である)ための条件は

$$\frac{\partial S(\boldsymbol{a})}{\partial a_k} = -2\sum_{i=1}^{N} f(x_i;\boldsymbol{a})\frac{\partial f(y_i;\boldsymbol{a})}{\partial a_k} = 0 \quad .$$

ヤコビ行列を $\boldsymbol{J}_i(\boldsymbol{a}) = \left[\partial f_i / \partial a_j \right]$ と表すと

$$\nabla S(\boldsymbol{a}) = -2\, \boldsymbol{J}^t(\boldsymbol{a}) f(\boldsymbol{y};\boldsymbol{a}) = 0$$

と書き換えることができます．これに Newton-Raphson 法を適用すると

$$\nabla^2 S(\boldsymbol{a})\, \Delta\boldsymbol{a}^{(k)} = -\nabla S\left(\boldsymbol{a}^{(k)}\right)$$

となります. $S(\boldsymbol{a})$ のあらわな 2 次微分の表現は

$$\frac{\partial^2 S(\boldsymbol{a})}{\partial a_k \partial a_{k'}} = -2\sum_{i=1}^{N} \frac{\partial \mathrm{f}(y_i;\boldsymbol{a})}{\partial a_k}\frac{\partial \mathrm{f}(y_i;\boldsymbol{a})}{\partial a_{k'}} + \sum_{i=1}^{N} \frac{\partial^2 \mathrm{f}(y_i;\boldsymbol{a})}{\partial a_k \partial a_{k'}} \mathrm{f}(y_i;\boldsymbol{a})$$

です．ここで，k 回目の推定パラメータ $\boldsymbol{a}^{(k)}$ に対する修正値を $\Delta\boldsymbol{a}^{(k)}$ とすると，

$$\left\{ J^t\left(\boldsymbol{a}^{(k)}\right) J\left(\boldsymbol{a}^{(k)}\right) + \sum_{i=1}^{N} \mathrm{f}\left(x_i;\boldsymbol{a}^{(k)}\right) \nabla^2 \mathrm{f}\left(x_i;\boldsymbol{a}^{(k)}\right) \right\} \Delta\boldsymbol{a}^{(k)} = J^t\left(\boldsymbol{a}^{(k)}\right) \mathrm{f}\left(x_i;\boldsymbol{a}^{(k)}\right)$$

が得られます．鉤括弧内の第 2 項を無視すると，先程の（6.3）式の線形的な関係が現れます．この無視する項は, 線形の場合には確かに現れないし, 1次の項に比べて小さいと考えられます. さらに $f(x_i;\boldsymbol{a})$ を掛けて和を取っているため, それらがお互いにキャンセルしてくれる可能性があります.

このGauss-Newton法と呼ばれる非線形最小二乗法は線形問題から拡張した方法として論理的に簡明であり，広く使われています．しかし，収束性は高くなく，むしろ発散しやすいので注意が必要です．先程の 2 次の項を無視するのでなく，うまく見積もる方法を用いたのが Levenberg-Marquardt 法です．明快な解説が Numerical Recipes in C(C 言語による数値計算のレシピ）William H. Press 他著，技術評論社 1993 にあります．

実践演習

うまく結果が出ないときに試してください．

以下の入力を打ち込むときの一般的なコツを最初にまとめておきます．

1．restart をかける：続けて入力すると前の入力が生きています．違う問題へ移るときやもう一度入力をし直すときには restart を入力してください．

2．出力してみる：紙面の関係で出力を抑止していますが，できるだけ多く出力するようにしてください．最後の：を；に変えるとか，printf を使ってください．

3．関数に値を代入してみる：数値が返ってくるべき時に変数があればどこかで入力をミスっています．plot で Plotting error, empty plot が出た場合にチェック下さい．

4．内側から順に入力する：長い入力は内側の関数から順に何をしているか確認しながら打ち込んで下さい．

実践演習[1] 化学反応式の係数決定（連立方程式の整数解）

次の反応式

$$a\,MnO_4^{2-} + b\,H^+ \text{ -> } c\,MnO_4^- + d\,MnO_2 + e\,H_2O$$

の係数を元素と電荷の保存から求めよ．

（参考『はじめての Maple』，B.W. チャー他著，サイバネットシステム株訳，シュープリンガー・フェアラーク東京，1993, p.205）

実践：　先ず元素，電荷の保存の条件式を代入します．

```
> Mn:=a=c+d;
  OO:=4*a=4*c+2*d+e;
  H:=b=2*e;
  Q:=-2*a+b=-c;
```

$$Mn := a = c + d$$

$$OO := 4\,a = 4\,c + 2\,d + e$$

$$H := b = 2\,e$$

$$Q := -2\,a + b = -c$$

これを整数の範囲で連立方程式を解いてくれる isolve を使って解きます．

```
> ans:=isolve({Mn,OO,H,Q});
```

$$ans := \{b = 4\,\_N1, a = 3\,\_N1, d = \_N1, c = 2\,\_N1, e = 2\,\_N1\}$$

方程式と未知数の数が一致していません．従って任意定数 *_N1* が出てきます．N1 に 1 を代入してそれぞれの未知変数を決めます．

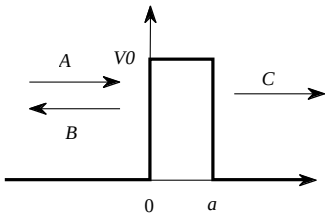
```
> subs(_N1=1,ans);
```

$$\{c = 2, e = 2, b = 4, a = 3, d = 1\}$$

実践演習[２] トンネル効果（式の変形）

（参考：シッフ著　量子力学（井上　健訳），吉岡書店 1970.

小出昭一郎著　基礎物理学選書 5Aー量子力学，裳華房 1969.



基礎：　量子効果の基礎的な例である1次元のトンネル効果についての式を導いてみましょう．上図のようなポテンシャルを考えます．詳しい解説は成書を参考にしていただくとして，以下で扱う数式の簡単な説明だけをしておきます．まず，ポテンシャルエネルギー$V(x)=0$ の領域での波動方程式は

$$-\frac{\hbar^2}{2m}\frac{d^2\varphi(x)}{dx^2}=\varepsilon\,\varphi(x)$$

ですから，波動関数は

$x\le 0$ では　　$\varphi(x)=Aexp\left(ikx\right)+Bexp\left(-ikx\right)$

$x\ge a$ では　　$\varphi(x)=Cexp\left(ikx\right)$　．

ここで$k=\sqrt{2m\varepsilon/\hbar^2}$ は波数ベクトルの大きさです．

ポテンシャルの壁の内側では$\varepsilon\lesseqgtr V_0$ によって事情が変わってきます．$\varepsilon\ge V_0$ では$\kappa=\sqrt{2m\left(\varepsilon-V_0\right)/\hbar^2}$ と定義すると，波動関数は

$0\le x\le a$ では　　$\varphi(x)=Fexp\left(i\kappa x\right)+Gexp\left(-i\kappa x\right)$

となります．<u>波動関数は粒子の座標に関する滑らかな連続関数でなければならない</u>という条件を $x=0$ と$x=a$ に適用すると，条件はそれぞれ

$x=0$ で$\varphi(x)$ が連続：$A+B=F+G$

$x=0$ で$\varphi'(x)$ が連続：$k\left(A-B\right)=\kappa\left(F-G\right)$

$x=a$ で$\varphi(x)$ が連続：$F\exp\left(i\kappa a\right)+G\exp\left(-i\kappa a\right)=C\exp\left(ika\right)$

$x=a$ で$\varphi'(x)$ が連続：$\kappa F\exp\left(i\kappa a\right)-\kappa G\exp\left(-i\kappa a\right)=kC\exp\left(ika\right)$

で与えられます．係数が5個で，方程式が4個ですから，それぞれの係数の比だけが求まります．これらからF,Gを消去して，B/A 入射波と反射波の複素振幅の比，およびC/A 入射波と透過波の複素振幅の比が求まります．これらの二乗が反射係数と透過係数にほかなりません．結果は

$$\left|\frac{B}{A}\right|^2=\left[1+\frac{4k^2\kappa^2}{\left(k^2-\kappa^2\right)^2\sin^2\kappa a}\right]^{-1}=\left[1+\frac{4\varepsilon\left(\varepsilon-V_0\right)}{V_0^2\sin^2\kappa a}\right]^{-1}$$

$$\left|\frac{C}{A}\right|^2 = \left[1 + \frac{\left(k^2-\kappa^2\right)^2 \sin^2 \kappa a}{4k^2\kappa^2}\right]^{-1} = \left[1 + \frac{V_0^2 \sin^2 \kappa a}{4\varepsilon\left(\varepsilon - V_0\right)}\right]^{-1}$$

となります.

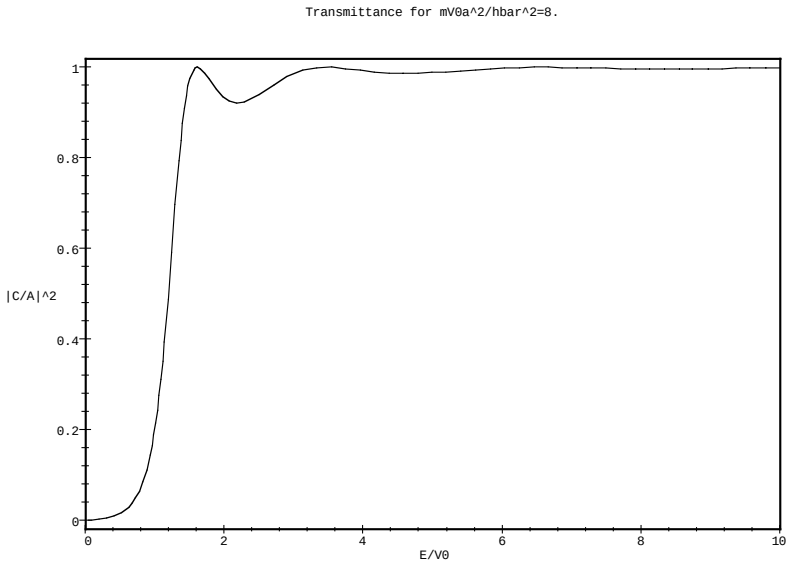
$0 < \varepsilon < V_0$ では$\alpha = \sqrt{2m\left(V_0-\varepsilon\right)/\hbar^2}$ として，波動関数は

$0 \le x \le a$ では $\varphi(x) = F\exp(\alpha x) + G\exp(-\alpha x)$

です．同様な計算によって

$$\left|\frac{C}{A}\right|^2 = \left[1 + \frac{V_0^2 \sinh^2 \alpha a}{4\varepsilon\left(\varepsilon - V_0\right)}\right]^{-1}$$

となります. $mV_0a^2/\hbar^2 = 8$ の場合の透過率を求めると下図のようになります.



実践： 境界条件から複素振幅の比の導出を実際にMapleで行ってみましょう．先ず境界での連続条件を打ち込むと，

```
> restart;
  eq1:=A+B=F+G;
  eq2:=k*(A-B)=kappa*(F-G);
  eq3:=F*exp(I*kappa*a)+G*exp(-I*kappa*a)=C*exp(I*k*a);
  eq4:=kappa*F*exp(I*kappa*a)-kappa*G*exp(-I*kappa*a)=k*C*exp(I*k*a);
```

$$eq1 := A + B = F + G$$

$$eq2 := k\,(A - B) = \kappa\,(F - G)$$

$$eq3 := F\,\mathbf{e}^{(I\,\kappa\,a)} + G\,\mathbf{e}^{(-I\,\kappa\,a)} = C\,\mathbf{e}^{(I\,k\,a)}$$

$$eq4 := \kappa\,F\,\mathbf{e}^{(I\,\kappa\,a)} - \kappa\,G\,\mathbf{e}^{(-I\,\kappa\,a)} = k\,C\,\mathbf{e}^{(I\,k\,a)}$$

次にB,Cを求めてみましょう.

```
> solve({eq1,eq2,eq3,eq4},{B/A,C/A});
```

としても何も返ってきません．Mapleはこれだけではお手上げな様です．仕方がないので，順番に解いていきましょう．F,Gを消去するという方針にしたがって，

```
> sol1:=solve({eq3,eq4},{F,G});
```

$$sol1 := \left\{ F = \frac{1}{2}\frac{C(\kappa+k)\,\mathbf{e}^{(-I\kappa a + I k a)}}{\kappa},\ G = \frac{1}{2}\frac{C(\kappa-k)\,\mathbf{e}^{(I k a + I\kappa a)}}{\kappa} \right\}$$

```
> assign(sol1);
```

アサインでF,Gへ答えを代入することを忘れずに！！

```
> F;
```

$$\frac{1}{2}\frac{C(\kappa+k)\,\mathbf{e}^{(-I\kappa a + I k a)}}{\kappa}$$

```
> sol2:=solve({eq1,eq2},{A,B});
```

$$sol2 := \left\{ B = -\frac{1}{4}\frac{C\left(\kappa^2 \left(\mathbf{e}^{(-I\kappa a)}\right)^2 - \kappa^2 - k^2 \left(\mathbf{e}^{(-I\kappa a)}\right)^2 + k^2\right)\mathbf{e}^{(I k a + I\kappa a)}}{\kappa k},\right.$$
$$\left. A = \frac{1}{4}\frac{C\left(\kappa^2 \left(\mathbf{e}^{(-I\kappa a)}\right)^2 - \kappa^2 + k^2 \left(\mathbf{e}^{(-I\kappa a)}\right)^2 - k^2 + 2\kappa \left(\mathbf{e}^{(-I\kappa a)}\right)^2 k + 2\kappa k\right)\mathbf{e}^{(I k a + I\kappa a)}}{\kappa k} \right\}$$

```
> assign(sol2);
> A/C;
```

$$\frac{1}{4}\frac{\left(\kappa^2 \left(\mathbf{e}^{(-I\kappa a)}\right)^2 - \kappa^2 + k^2 \left(\mathbf{e}^{(-I\kappa a)}\right)^2 - k^2 + 2\kappa \left(\mathbf{e}^{(-I\kappa a)}\right)^2 k + 2\kappa k\right)\mathbf{e}^{(I k a + I\kappa a)}}{\kappa k}$$

```
> A/B;
```

$$-\frac{\kappa^2 \left(\mathbf{e}^{(-I\kappa a)}\right)^2 - \kappa^2 + k^2 \left(\mathbf{e}^{(-I\kappa a)}\right)^2 - k^2 + 2\kappa \left(\mathbf{e}^{(-I\kappa a)}\right)^2 k + 2\kappa k}{\kappa^2 \left(\mathbf{e}^{(-I\kappa a)}\right)^2 - \kappa^2 - k^2 \left(\mathbf{e}^{(-I\kappa a)}\right)^2 + k^2}$$

でほぼ求まっています．ここからの式の単純化は苦労します．以下では簡単に求めているように見えますが，ここまで来るのに大分試行錯誤しています．手でやるほうが早いですが，expとtrigonal関数の間の対応や分母の有理化などを扱うときや，式が膨大な場合はMapleを補助的に使うと計算間違いが少なくすみます．もっとうまいやり方がある場合は教えて下さい．先ずはB/Aで計算を進めていきます．

```
> BB:=simplify(convert(A/B,trig));
```

$$BB := -\frac{-\kappa^2 + \kappa^2\cos(\kappa a)^2 - 2I\kappa k\cos(\kappa a)\sin(\kappa a) + k^2\cos(\kappa a)^2 - k^2}{\kappa^2\cos(\kappa a)^2 - \kappa^2 - k^2\cos(\kappa a)^2 + k^2}$$

振幅の二乗を求めるために，分子の複素共役を取ってみましょう．

```
> conjugate(BB);
```

$$-\left(\frac{-\kappa^2 + \kappa^2\cos(\kappa a)^2 - 2I\kappa k\cos(\kappa a)\sin(\kappa a) + k^2\cos(\kappa a)^2 - k^2}{\kappa^2\cos(\kappa a)^2 - \kappa^2 - k^2\cos(\kappa a)^2 + k^2}\right)$$

うまくいきません．これはMapleが数の範囲を複素数まで考えているためです．そこで，k,kappa,aなどの変数を実数であるとMapleにあらかじめ教えておきます．コマンドは

```
> assume(k,real):
  assume(kappa,real):
  assume(a,real):
```

です．以下の出力では assume がかかっている変数には～(チルデ)がついています．simplify の二つ目の変数は side relations(副関係式)をあらわに指定しています．

```
> simplify(conjugate(BB)*BB,{cos(kappa*a)^2=1-sin(kappa*a)^2});
```

$$\frac{4\,\kappa\sim^2 k\sim^2 \sin(\kappa\sim a\sim)^2 + (-2\,\kappa\sim^2 k\sim^2 + \kappa\sim^4 + k\sim^4)\sin(\kappa\sim a\sim)^4}{(-2\,\kappa\sim^2 k\sim^2 + \kappa\sim^4 + k\sim^4)\sin(\kappa\sim a\sim)^4}$$

```
> collect(%,sin(kappa*a));
```

$$1 + 4\frac{\kappa\sim^2 k\sim^2}{(-2\,\kappa\sim^2 k\sim^2 + \kappa\sim^4 + k\sim^4)\sin(\kappa\sim a\sim)^2}$$

次に A/C を求めていきます．

```
> CC:=convert(A/C,trig);
```

$$CC := \frac{1}{4}((\kappa\sim^2(\cos(\kappa\sim a\sim) - I\sin(\kappa\sim a\sim))^2 - \kappa\sim^2 + k\sim^2(\cos(\kappa\sim a\sim) - I\sin(\kappa\sim a\sim))^2 - k\sim^2$$
$$+ 2\,\kappa\sim(\cos(\kappa\sim a\sim) - I\sin(\kappa\sim a\sim))^2 k\sim + 2\,\kappa\sim k\sim)(\cos(k\sim a\sim + \kappa\sim a\sim) + I\sin(k\sim a\sim + \kappa\sim a\sim)))/(\kappa\sim k\sim)$$

```
> simplify(conjugate(CC)*CC);
```

$$-\frac{1}{4}\frac{-2\,\kappa\sim^2 k\sim^2 + \kappa\sim^4\cos(\kappa\sim a\sim)^2 - k\sim^4 + k\sim^4\cos(\kappa\sim a\sim)^2 - 2\,\kappa\sim^2 k\sim^2\cos(\kappa\sim a\sim)^2 - \kappa\sim^4}{\kappa\sim^2 k\sim^2}$$

```
> simplify(%,{cos(kappa*a)^2=1-sin(kappa*a)^2});
```

$$\frac{1}{4}\frac{(k\sim^4 - 2\,k\sim^2\kappa\sim^2 + \kappa\sim^4)\sin(\kappa\sim a\sim)^2 + 4\,k\sim^2\kappa\sim^2}{k\sim^2\kappa\sim^2}$$

```
> collect(%,sin(kappa*a));
```

$$\frac{1}{4}\frac{(k\sim^4 - 2\,k\sim^2\kappa\sim^2 + \kappa\sim^4)\sin(\kappa\sim a\sim)^2}{k\sim^2\kappa\sim^2} + 1$$

で大体もとまりました．

次に先ほど示した，透過率の plot を試みてみましょう．先ず求める関数を打ち込んでみます．

```
> CC3:=epsilon->(1+V0^2*sinh(alpha*a)^2/4/epsilon/(V0-epsilon))^(-1);
```

$$CC3 := \varepsilon \to \frac{1}{1 + \frac{1}{4}\frac{V0^2\sinh(\alpha\,a)^2}{\varepsilon\,(V0-\varepsilon)}}$$

これだけでは変数の値が定まっていませんのプロットできません．条件$mV_0a^2/\hbar^2 = 8$ からmを消去することを試みます．

```
> eq5:=m*V0*a^2/hbar^2=8;
```

$$eq5 := \frac{m\,V0\,a\sim^2}{hbar^2} = 8$$

```
> alpha:=sqrt(2*m/hbar^2*(V0-epsilon));
```

$$\alpha := \sqrt{2}\sqrt{\frac{m\,(V0-\varepsilon)}{hbar^2}}$$

```
> solm:=solve({eq5},m);
```

$$solm := \left\{ m = 8\,\frac{hbar^2}{V0\,a\sim^2} \right\}$$

これを単純化するうまい方法が見つけられませんでした．仕方なく，二乗してV0を置き換えて，さらにルートを取りました．

```
> alpha:=sqrt(expand(subs(V0=1/iV0,(subs(solm,(alpha*a)^2)))));
```

$$\alpha := 4\sqrt{1 - iV0\,\varepsilon}$$

先程の関数の一部を取りだして書き換えます．ここで，alpha*a=alpha と代えています．

```
> fun:=subs(V0=1/iV0,V0^2*sinh(alpha)^2/4/epsilon/(V0-epsilon));
```

$$fun := \frac{1}{4}\,\frac{\sinh(4\sqrt{1 - iV0\,\varepsilon})^2}{iV0^2\,\varepsilon\left(\frac{1}{iV0} - \varepsilon\right)}$$

このままではうまく単純化できません．分母と分子を別々に扱います．

```
> den_f:=denom(fun)/iV0;
  num_f:=numer(fun)/iV0;
```

$$den\_f := 4\,iV0\,\varepsilon\,(-1 + iV0\,\varepsilon)$$

$$num\_f := -\sinh(4\sqrt{1 - iV0\,\varepsilon})^2$$

iVo*epsilon を置き換えます．

```
> den_f1:=subs(iV0*epsilon=x,iV0^2*epsilon^2=x^2,expand(den_f));
```

$$den\_f1 := -4\,x + 4\,x^2$$

```
> num_f1:=subs(iV0*epsilon=x,num_f);
```

$$num\_f1 := -\sinh(4\sqrt{1 - x})^2$$

unapply 関数でユーザー関数とします．

```
> CC3:=unapply((1+num_f1/den_f1)^(-1),x);
```

$$CC3 := \frac{1}{x \to 1 - \dfrac{\sinh(4\sqrt{1 - x})^2}{-4\,x + 4\,x^2}}$$

```
> plot(CC3(x),x=0..10,labels=["E/V0","|C/A|^2"]);
```

で先程の透過率の図が plot されます．

実践演習[３] 熱膨張係数の導出（複雑な関数の近似と積分）

（参考　キッテル著　固体物理学入門　宇野良清他訳，丸善 1978）

基礎：　熱膨張(thermal expansion)は原子間ポテンシャルの二次以上の項によって現れます. 平衡点からの原子の変位 x のポテンシャルエネルギーを

$$U(x) = c\,x^2 - g\,x^3$$

と取ることができます．二次までの項では古典的な調和振動子を表し，熱膨張は現れません．x^3 の項は原子間相互作用の非対称性を表し，この項が熱膨張係数と直接かかわってきます.

有限温度での平均の変位は，ボルツマン分布関数を計算することで求まります．平均の位置xは，熱力学的な確率で重みづけられ，

$$\langle x \rangle = \frac{\int_{-\infty}^{\infty} x \exp\left(-\beta U(x)\right) dx}{\int_{-\infty}^{\infty} \exp\left(-\beta U(x)\right) dx}$$

で計算できます．ここで$\beta \equiv 1/\left(k_B T\right)$ です．この積分を実行すると$\langle x \rangle = \frac{3g}{4c^2} k_B T$ となります.

実践：　最後の導出が問題です．ここでは先ず近似によって式を簡単化します．これはお決まりの Taylor 展開です．味噌はポテンシャルエネルギーの項で二次の項はそのまま残し，三次以上を展開することです．つまり三次以上の変化が$1/k_B T$ よりも小さいと考えると展開が可能です．実際に Maple で展開してみると，

```
> series(exp(-beta*(-g*x^3)),x,5);
```

$$1 + \beta\, g\, x^3 + O(x^5)$$

です．これから関数を取りだします，

```
> temp:=convert(%,polynom);
```

$$temp := 1 + \beta\, g\, x^3$$

```
> f1:=unapply(exp(-beta*c*x^2)*temp,x);
```

$$f1 := x \to \mathbf{e}^{(-\beta\, c\, x^2)} \left(1 + \beta\, g\, x^3\right)$$

- α から α まで積分して無限大まで持っていきます.（内側のコマンドから順に確認しながら打ち込んで下さい．式をどのように変形していくかが分かります）

```
> limit(int(f1(x),x=-alpha..alpha),alpha=infinity);
```

$$\lim_{\alpha \to \infty} \frac{\mathbf{e}^{(-\alpha^2\, \beta\, c)} \sqrt{\pi}\, \mathrm{erf}(\alpha \sqrt{\beta\, c})\, \mathbf{e}^{(\alpha^2\, \beta\, c)}}{\sqrt{\beta\, c}}$$

うまく計算してくれません．これはβc が非負である必要があるためです．実践演習[２]で用いた assume で，

```
> assume(beta>0);
  assume(c>0);
```

としておきます．実際の積分を実行してみましょう．先ず分母は，

```
> limit(int(f1(x),x=-alpha..alpha),alpha=infinity);
```

$$\frac{\sqrt{\pi}}{\sqrt{\beta\sim\ c\sim}}$$

次に分子は

```
> limit(normal(int(x*f1(x),x=-alpha..alpha)),alpha=infinity);
```

$$\frac{3}{4}\frac{\sqrt{\pi}\ g}{\beta\sim\ \sqrt{\beta\sim\ c\sim}\ c\sim^2}$$

です．これを取りだして

```
> den:=%%;
```

$$den := \frac{\sqrt{\pi}}{\sqrt{\beta\sim\ c\sim}}$$

```
> num:=%%;
```

$$num := \frac{3}{4}\frac{\sqrt{\pi}\ g}{\beta\sim\ \sqrt{\beta\sim\ c\sim}\ c\sim^2}$$

```
> simplify(num/den);
```

$$\frac{3}{4}\frac{g}{\beta\sim\ c\sim^2}$$

実践演習[４] 陽電子消滅寿命（連立微分方程式）

次の微分方程式の一般解を求めよ．

$$\left(\frac{\partial}{\partial t}\mathrm{n}(t)\right)+\lambda\,\mathrm{n}(t)=0 \qquad (1)$$

さらに

$$\left(\frac{\partial}{\partial t}\mathrm{n1}(t)\right)+(\lambda 1+K12)\,\mathrm{n1}(t)=0 \qquad (2a)$$

$$\left(\frac{\partial}{\partial t}\mathrm{n2}(t)\right)+\lambda 2\,\mathrm{n2}(t)=K12\,\mathrm{n1}(t) \qquad (2b)$$

の連立微分方程式を$\mathrm{n1}(0)=n0,\mathrm{n2}(0)=0$を初期条件として求めよ．

基礎： これは固体中に入った陽電子がとる状態の寿命を記述する微分方程式です．陽電子は，あるひとつのサイト上で（１）式のような微分方程式にしたがった寿命を示します．しかし，複数のサイトがあると，遷移確率をK_{ij} $(K_{ij}\neq K_{ji})$とした場合，正確には

$$\frac{dn_i(t)}{dt}+\left(\lambda_i+\sum_{j\neq i}^{N}K_{ij}\right)n_i(t)=\sum_{j\neq i}^{N}K_{ji}\,n_j(t)$$

なる連立微分方程式で表す必要があります．しかし，これを解けば分かるように，２つのサイトだけで陽電子が消滅すると仮定しても，表式は相当複雑になります．そこで，trapping modelが提唱され，それに基づいて実験結果の解析がおこなわれています．このモデルでは，ある時刻$t=0$で全ての陽電子がひとつの状態に居り，そこで消滅するかまたは他のサイトへ遷移していくと考えます．遷移した状態からさらに他の状態への遷移は小さいと考えると相当単純になります．ここで消滅サイトが２つだとすると(2a),(2b)式が得られます．

（参考　"Positron in Solids", ed by P.Hautojarvi, Springer-Verlag, Berlin 1979）

実践：

```
> deq:={diff(n(t),t)+lambda*n(t)=0};
```

$$deq:=\left\{\left(\frac{\partial}{\partial t}\mathrm{n}(t)\right)+\lambda\,\mathrm{n}(t)=0\right\}$$

```
> dsolve(deq,n(t));
```

$$\mathrm{n}(t)=\_C1\,\mathbf{e}^{(-\lambda\,t)}$$

```
> deq:={diff(n1(t),t)+(lambda1+K12)*n1(t)=0,
  diff(n2(t),t) +(lambda2)*n2(t)=K12*n1(t),
  n1(0)=n0,n2(0)=0};
```

$$deq:=\left\{\left(\frac{\partial}{\partial t}\mathrm{n2}(t)\right)+\lambda 2\,\mathrm{n2}(t)=K12\,\mathrm{n1}(t),\left(\frac{\partial}{\partial t}\mathrm{n1}(t)\right)+(\lambda 1+K12)\,\mathrm{n1}(t)=0,\ \mathrm{n1}(0)=n0,\mathrm{n2}(0)=0\right\}$$

```
> dsolve(deq,{n1(t),n2(t)});
```

$$\left\{\mathrm{n2}(t)=-\frac{n0\,K12\,\mathbf{e}^{(-\lambda 2\,t)}}{\lambda 2-\lambda 1-K12}+\frac{n0\,K12\,\mathbf{e}^{(-t\,\lambda 1-t\,K12)}}{\lambda 2-\lambda 1-K12},\ \mathrm{n1}(t)=n0\,\mathbf{e}^{(-t\,\lambda 1-t\,K12)}\right\}$$

実践演習[５] Hamiltonian の解（線形代数）

異核 s 価電子ボンド 2 量体の原子軌道の線形結合から求めた永年方程式は

$$\begin{pmatrix} -\frac{1}{2}\Delta E-\left(E-\bar{E}\right) & h-\left(E-\bar{E}\right)S \\ h-\left(E-\bar{E}\right)S & +\frac{1}{2}\Delta E-\left(E-\bar{E}\right) \end{pmatrix}\begin{pmatrix} c_A \\ c_B \end{pmatrix}=0$$

である．ここで, $\Delta E=E_A-E_B$ は原子エネルギー準位のづれ, $\bar{E}=½\left(E_A+E_B\right)$ ，h と S とはそれぞれ軌道間のボンド積分と重なり積分を表す．この式を解き, S の2次以上の項を無視することによって結合状態と反結合状態の固有値

$$E^{\pm}=\bar{E}+|h|\,S\mp\frac{1}{2}\sqrt{4h^2+\left(\Delta E\right)^2}$$

を導け．

（参考　分子・固体の結合と構造，ディヴィット・ペティフォー著，技報堂 1997）

実践：

```
> with(linalg):
> diago1:=-1/2*DE-(E1-Eav);
  diago2:=1/2*DE-(E1-Eav);
  ndiago:=h-(E1-Eav)*S;
```

$$diago1 := -\frac{1}{2}DE - E1 + Eav$$

$$diago2 := \frac{1}{2}DE - E1 + Eav$$

$$ndiago := h - (E1 - Eav)\,S$$

```
> ham1:=matrix(2,2,[diago1,ndiago,ndiago,diago2]);
```

$$ham1 := \begin{bmatrix} -\frac{1}{2}DE - E1 + Eav & h - (E1 - Eav)\,S \\ h - (E1 - Eav)\,S & \frac{1}{2}DE - E1 + Eav \end{bmatrix}$$

```
> sol1:=det(ham1);
```

$$sol1 := -\frac{1}{4}DE^2 + E1^2 - 2\,E1\,Eav + Eav^2 - h^2 + 2\,h\,S\,E1 - 2\,h\,S\,Eav$$
$$- S^2\,E1^2 + 2\,S^2\,E1\,Eav - S^2\,Eav^2$$

```
> solve(sol1=0,E1);
```

$$\frac{1}{2}\frac{8\,h\,S - 8\,Eav + 8\,S^2\,Eav + 4\sqrt{DE^2 + 4\,h^2 - S^2\,DE^2}}{-4 + 4\,S^2},$$
$$\frac{1}{2}\frac{8\,h\,S - 8\,Eav + 8\,S^2\,Eav - 4\sqrt{DE^2 + 4\,h^2 - S^2\,DE^2}}{-4 + 4\,S^2}$$

```
> Eplus:=%[2];
```

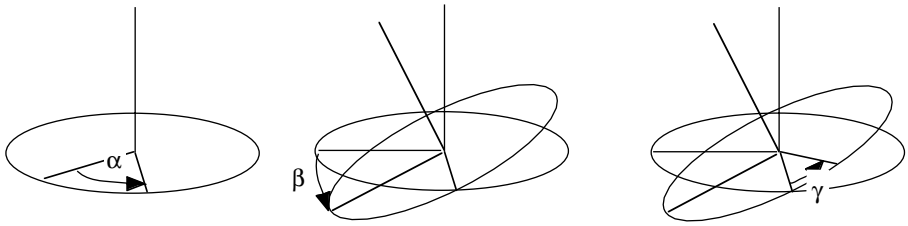
$$Eplus := \frac{1}{2}\frac{8\,h\,S - 8\,Eav + 8\,S^2\,Eav - 4\sqrt{DE^2 + 4\,h^2 - S^2\,DE^2}}{-4 + 4\,S^2}$$

```
> series(Eplus,S,2);
```

$$\left(Eav + \frac{1}{2}\sqrt{DE^2 + 4\,h^2}\right) - h\,S + O(S^2)$$

実践演習[６] オイラー角（線形代数）

中心点を固定した，図のような軸の周りの連続する回転を考え，２つの座標系の方向余弦をオイラー角で表せ．



（参考：ベクトル・テンソルと行列，ジョージ・アルフケン著，講談社 1977）

```
> with(linalg):

Warning: new definition for   norm
Warning: new definition for   trace
> Z1:=matrix(3,3,[cp,-sp,0,sp,cp,0,0,0,1]);
```

$$Z1 := \begin{bmatrix} cp & -sp & 0 \\ sp & cp & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> Z2:=matrix(3,3,[cz,-sz,0,sz,cz,0,0,0,1]);
```

$$Z2 := \begin{bmatrix} cz & -sz & 0 \\ sz & cz & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

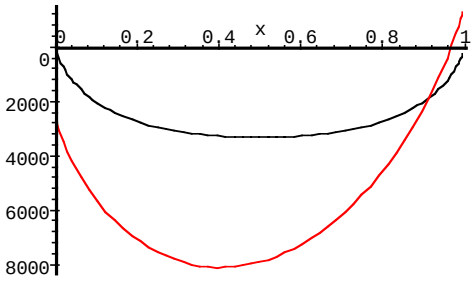
```
> Y:=matrix(3,3,[ct,0,st,0,1,0,-st,0,ct]);
```

$$Y := \begin{bmatrix} ct & 0 & st \\ 0 & 1 & 0 \\ -st & 0 & ct \end{bmatrix}$$

```
> Trans1:=multiply(Z1,Y,Z2);
```

$$Trans1 := \begin{bmatrix} cp\,ct\,cz - sp\,sz & -cp\,ct\,sz - sp\,cz & cp\,st \\ sp\,ct\,cz + cp\,sz & -sp\,ct\,sz + cp\,cz & sp\,st \\ -st\,cz & st\,sz & ct \end{bmatrix}$$

実践演習[７] 組成自由エネルギー曲線（連立方程式の数値解）



上図は2成分系1300Kにおける，正則溶体近似に基づいた液相と固相の自由エネルギー組成曲線を表している．この曲線を表す関数を求めよ．また，ある温度での２相平衡の組成を求めよ．

（参考：材料組織学，杉本孝一，長村光造，山根壽己，牧正志，菊池潮美，落合庄治郎，村上陽太郎著，朝倉書店 1991）

基礎：　材料科学でよく出てくる状態図の求め方の基礎です．馴染みのない方がほとんどだと思いますので，詳しく解説しながら数値的に連立方程式を解く方法を見ていきます．

先ず状態図の求め方を逆順で書くと，

１）状態図を求めることは二相平衡の条件を求めることである．

２）固体と液体の化学ポテンシャルが等しい組性を求める．

３）化学ポテンシャルは自由エネルギー関数の微分で求まる．

４）正則溶体近似に基づいた自由エネルギー関数を求める．

５）自由エネルギー関数に必要な値を仮定する．

となります．２）と３）とはまとめて共通接線を求めることと等価です．まず４）の正則溶体近似に基づいた自由エネルギー関数は

```
> Gs:=(xb,temp)->(1-xb)*muas+xb*mubs+dGms(xb,temp);
  Gl:=(xb,temp)->(1-xb)*mual+xb*mubl+dGml(xb,temp);
```

$$Gs := (xb, temp) \to (1 - xb)\, muas + xb\, mubs + \mathrm{dGms}(xb, temp)$$

$$Gl := (xb, temp) \to (1 - xb)\, mual + xb\, mubl + \mathrm{dGml}(xb, temp)$$

です．この中でdGms,dGmlで表した混合の自由エネルギーは正則溶体では理想溶体の混合のエントロピーと相互作用パラメーター(omega)を用いて，

```
> dGms:=(xb,temp)->Omegas*(1-xb)*xb+RR*temp*(xb*ln(xb)+(1-xb)*ln(1-xb));
  dGml:=(xb,temp)->Omegal*(1-xb)*xb+RR*temp*(xb*ln(xb)+(1-xb)*ln(1-xb));
```

$$dGms := (xb, temp) \to Omegas\,(1 - xb)\, xb + RR\, temp\,(xb \ln(xb) + (1 - xb) \ln(1 - xb))$$

$$dGml := (xb, temp) \to Omegal\,(1 - xb)\, xb + RR\, temp\,(xb \ln(xb) + (1 - xb) \ln(1 - xb))$$

となります．これらの式中で使われている定数と変数を以下のようにとります．

```
> RR:=8.314:
  Omegas:=16.7*1000: muas:=0: mubs:=0:
  Omegal:=0:
  dHa:=8.37*1000: dHb:=12.55*1000:
  dSa:=dHa/1000: dSb:=dHb/1500:
```

mual と mubl は融点において液相と固相の化学ポテンシャルの差が消えることからその温度依存を仮定します．

```
> mual:=temp->dHa-temp*dSa;
  mubl:=temp->dHb-temp*dSb;
```

$$mual := temp \to dHa - temp\, dSa$$

$$mubl := temp \rightarrow dHb - temp\ dSb$$

これに伴って先程定義した液相の自由エネルギー関数のmuも温度に依存した関数として定義し直しておく必要が出てきます.

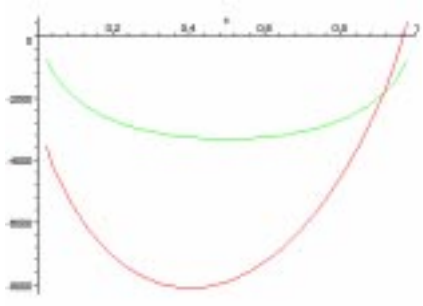
```
> Gl:=(xb,temp)->(1-xb)*mual(temp)+xb*mubl(temp)+dGml(xb,temp);
```

$$Gl := (xb, temp) \rightarrow (1 - xb)\ \mathrm{mual}(temp) + xb\ \mathrm{mubl}(temp) + \mathrm{dGml}(xb, temp)$$

これらの液相･固相の自由エネルギー関数がある一定温度でちゃんと再現されているか確かめておきましょう.

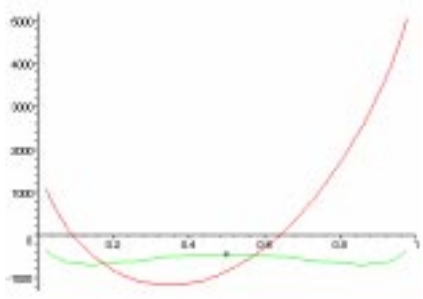
```
> templ:=1300;
  plot({Gl(x,templ),Gs(x,templ)},x=0..1);
```

$$templ := 1300$$



```
> templ:=800;
  plot({Gl(x,templ),Gs(x,templ)},x=0..1);
```

$$templ := 800$$



これで作業項目の４），５）が終わりました．あとは共通接線を求めればいいだけです.

関数の微分がdiffによって求められることを思い出して,

```
> diff(Gl(y,temp),y);
```

$$4180.00 + .003333333\ temp + 8.314\ temp\ (\ln(y) - \ln(1 - y))$$

まず傾きの差を

```
>  F1:=(x,y,temp)->subs(x1=x,y1=y,diff(Gs(x1,temp),x1)-
diff(Gl(y1,temp),y1));
```

$$F1 := (x, y, temp) \rightarrow \mathrm{subs}\left(x1 = x, y1 = y, \left(\frac{\partial}{\partial x1}\mathrm{Gs}(x1, temp)\right) - \left(\frac{\partial}{\partial y1}\mathrm{Gl}(y1, temp)\right)\right)$$

次に切片の差は

```
> F2:=(x,y,temp)->subs(x1=x,y1=y,Gs(x1,temp)-diff(Gs(x1,temp),x1)*x1-
(Gl(y1,temp)-diff(Gl(y1,temp),y1)*y1));
```

$$F2 := (x, y, temp) \rightarrow \mathrm{subs}\left(x1 = x, y1 = y, \mathrm{Gs}(x1, temp) - \left(\frac{\partial}{\partial x1} \mathrm{Gs}(x1, temp) \right) x1 - \mathrm{Gl}(y1, temp) + \left(\frac{\partial}{\partial y1} \mathrm{Gl}(y1, temp) \right) y1 \right)$$

これを fsolve で連立方程式として，x，y について数値的に求めてみましょう．

```
> fsolve({F1(x,y,1300)=0,F2(x,y,1300)=0},{x,y},{x=0.5..0.99,y=0.5..0.99});
```

$$\{y = .8308884633, x = .9685370576\}$$

となり，解が得られます．これを関数と定義します．

```
> FindXY:=(temp)->fsolve({F1(x,y,temp)=0,F2(x,y,temp)=0}, {x,y},
{x=0.5..0.99, y=0.5..0.99});
```

$$FindXY := temp \rightarrow \mathrm{fsolve}(\{\mathrm{F1}(x, y, temp) = 0, \mathrm{F2}(x, y, temp) = 0\}, \{x, y\}, \{y = .5 \mathinner{..} .99, x = .5 \mathinner{..} .99\})$$

```
> FindXY(1300);
```

$$\{x = .9685370576, y = .8308884633\}$$

これを温度を順繰りに変えて求めます．

```
> for temp from 1400 by -100 to 900
  do
    print(temp);
    FindXY(temp);
  od;
```

$$1400$$
$$\{y = .9164387422, x = .9843946402\}$$
$$1300$$
$$\{x = .9685370576, y = .8308884633\}$$
$$1200$$
$$\{x = .9524858191, y = .7434269425\}$$
$$1100$$
$$\{x = .9363612791, y = .6542285850\}$$
$$1000$$
$$\{x = .9204118951, y = .5636183410\}$$
$$900$$

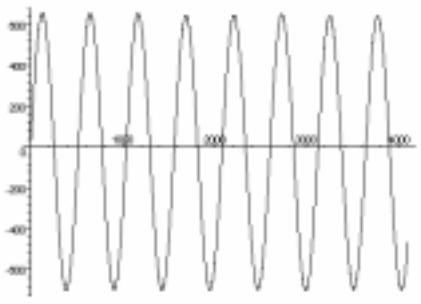
$$\mathrm{fsolve}(\{16700.0\,(1-x)\,x + 7482.600\,x\ln(x) + 7482.600\,(1-x)\ln(1-x) - (-33400.0\,x + 16700.000 + 7482.600\ln(x) - 7482.600\ln(1-x))\,x - 837.000000 - 4183.000000\,y - 7482.600\,y\ln(y) - 7482.600\,(1-y)\ln(1-y) + (4183.00000 + 7482.600\ln(y) - 7482.600\ln(1-y))\,y = 0, -33400.0\,x + 12517.00000 + 7482.600\ln(x) - 7482.600\ln(1-x) - 7482.600\ln(y) + 7482.600\ln(1-y) = 0\}, \{x, y\}, \{y = .5 \mathinner{..} .99, x = .5 \mathinner{..} .99\})$$

ちゃんと求まっているでしょうか？　このような作業も立派なプログラミングの基礎です．あとはこれを関数として全て定義してしまえば終わりです．しかし，最後のところでエラーが出ています．これがなぜ出たか考えてみてください．広い温度範囲で求めたり，純 A 組性に近いところでの固相ー液相平衡などに対応する必要があります．違ったアルゴリズムを使うか，条件判断を加えてより汎用性の高いプログラムを組むのが本来のプログラミングの難しいところです（本書の範囲を越えますのでここでは扱いません）．

実践演習[８] 減衰振動のフーリエ変換（高速フーリエ変換)

ファイル名 DATA102 は，鋼の棒をわずかに弾性変形して自由振動させたときの波形である．この振動にはどのような周波数成分が含まれているか，複素フーリエ変換により調べよ．固体を振動させると，振幅は大なり小なり減衰する．これは，固体中のさまざまな欠陥が弾性エネルギーを吸収して熱エネルギーに変えてしまうことに起因する．

```
> T:=readdata(`bob's pb1400:DATA102`,1):
> with(plots):
  listplot(T);
```



```
> with(linalg):
  n:=vectdim(T);
  Rdata:=convert(T,array):
  Idata:=array(sparse,1..n):
```

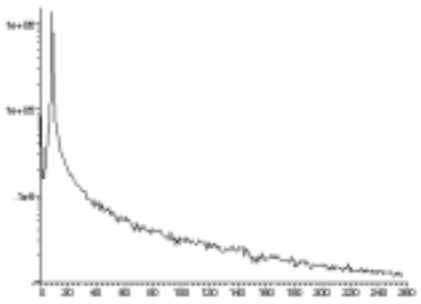
$$n := 4096$$

```
> readlib(FFT);
  FFT(12,Rdata,Idata);
```

$$\mathbf{proc}(m, x, y) \ \ldots \ \mathbf{end}$$

$$4096$$

```
> with(plots):
  datapoint:=[seq([i,sqrt(Rdata[i]^2+Idata[i]^2)],i=1..256)]:
  logplot(datapoint);
```



付録Ａ　コマンドのまとめ

本文で引用したコマンドをメモ代わりに載せておきます. 徐々に充実させます. 有益なヘルプは次の通りです.

?inifcns - 起動時から認識されている関数
?index[package] - 関連する関数を集めたパッケージです．微分方程式(DETools), 線形代数(linalg)，プロット関係の関数(plots)等があります.
?index[function] - Maple の標準関数.

1）関数パッケージ

?inifcns - 起動時から認識されている関数
?integer - 整数関数
?polynom - 多項式の取り扱いに関する関数
?ratpoly - 有理関数
?vector - ベクトル関数
?matrix - 行列に関する関数
?numtheory - 整数論
?combinat - 組み合わせ数学
?stats - 統計学
?orthopoly - 直交多項式

2）式の変形

expand - Expand an Expression（展開）
simplify - Apply Simplification Rules to an Expression（簡単化）
collect - Collect Coefficients of Like Powers（同じ項により式をまとめる）
combine - combine terms into a single term（規則にしたがって式をまとめる）
coeff - extract a coefficient of a polynomial（多項式の係数の取り出し）
sort - sort a list of values or a polynomial（式や値のソート）
factor - Factor a Multivariate Polynomial（一つ以上の変数を持つ多項式の因数分解）
normal - normalize a rational expression（分子と分母からなる式の共通因子の削除．分数の結合も行う）
convert - convert an expression to a different form（関数を違う関数へ変換）
gcd - greatest common divisor of polynomials（多項式の最大公約数）
lcm - least common multiple of polynomials（多項式の最小公倍数）
numer - numerator of an expression （分母）
denom - denominator of an expression（分子）
assume - The Assume Facility（変数の範囲や関係を規定）

3）線形代数

linalg[matadd] - matrix or vector addition（和）(matadd(A, B, c1, c2):c1*A+c2*B)
linalg[multiply] - matrix-matrix or matrix-vector multiplication（積）
linalg[dotprod] - vector dot (scalar) product（内積）
linalg[crossprod] - vector cross product（外積）
linalg[inverse] - compute the inverse of a matrix（逆行列）
linalg[det] - determinant of a matrix（行列式）
linalg[trace] - the trace of a matrix（対角和）
linalg[adjoint] - compute the adjoint of a matrix（随伴）
linalg[transpose] - compute the transpose of a matrix（転置行列）
linalg[eigenvals] - compute the eigenvalues of a matrix（固有値）
linalg[eigenvects] - find the eigenvectors of a matrix（固有ベクトル）
linalg[angle] - compute the angle between vectors（角度）
linalg[rowdim] - determine the row dimension of a matrix（行の数）
linalg[coldim] - determine the column dimension of a matrix（列の数）
linalg[vectdim] - determine the dimension of a vector（ベクトルの要素数）
この他にもcurl,diverge,grad,jacobianなどの微分演算子や, 行列やベクトルの行や列を操作するオペレーションが用意されています.

付録Ｂ　入出力に関する補足

Unixでの使用の際にはフィルターとしてMapleが使えると便利です．また，Mapleにはプログラム言語への翻訳やLATEXへの出力機能が備わっています．これらを簡単に紹介しておきます．

１）データの入出力（フィルターとしての使用）

unixの標準的な使い方として，

```
maple -f action.txt < input.data >output.data
```

というのが用意されていればいいのですが，どうやら現versionではなさそうです．そこでそれに近い動作をさせるスクリプトを示し，あわせてmintの使い方も少し説明しておきます．これは`DATA101`中のデータを読み込んで，`out.data`に書き込むスクリプトです．editorで以下のような入力スクリプトを用意します．

```
[bob@white tmp]$ cat test.txt
T:=readdata('/home/bob/DATA101',float,1):
writedata('/home/bob/out.data',T);
quit:
```

非常に単純なスクリプトで書けます．これが動くかを確認するために，mintでスクリプトのチェックをしてみました．上記のスクリプトにはバグが潜んでいます．

```
[bob@white tmp]$ mint < test.txt
    |\^/|      Maple V Release 5 Diagnostic Program
._|\|   |/|_.  Copyright (c) 1981-1997 by Waterloo Maple Inc. All rights
 \  MINT   /   reserved. Maple and Maple V are registered trademarks of
 <____ ____>   Maple and Maple V are registered trademarks of
      |        Waterloo Maple Inc.
on line     1: T:=readdata('/home/bob/DATA101',float,1):
                           ^ syntax error
on line     2: writedata('/home/bob/out.data',T);
                         ^ syntax error
```

mintは二つのsyntax errorを出しています．syntax errorはバックコートで囲むべき文字列をシングルコートで囲んだ事によるerrorです．これを訂正して，

```
[bob@white tmp]$ cat test.txt
T:=readdata(`/home/bob/DATA101`,float,1):
writedata(`/home/bob/out.data`,T);
quit:
[bob@white tmp]$ mint < test.txt
    |\^/|      Maple V Release 5 Diagnostic Program
._|\|   |/|_.  Copyright (c) 1981-1997 by Waterloo Maple Inc. All rights
 \  MINT   /   reserved. Maple and Maple V are registered trademarks of
 <____ ____>   Maple and Maple V are registered trademarks of
      |        Waterloo Maple Inc.
```

と全くエラーを返しません．走らせてみると，

```
[bob@white tmp]$ maple < test.txt
    |\^/|     Maple V Release 5 (Kyoto University)
._|\|   |/|_. Copyright (c) 1981-1997 by Waterloo Maple Inc. All rights
 \  MAPLE  /  reserved. Maple and Maple V are registered trademarks of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> T:=readdata(`/home/bob/DATA101`,float,1):
> writedata(`/home/bob/out.data`,T);
```

```
> quit:
bytes used=434620, alloc=458668, time=0.07
[bob@white tmp]$ wc ../out.data
    256     256    3585 ../out.data
```

めでたく出力されています.

2）C, LATEX への出力

Maple で作った式や関数などは，Fortran や C,LATEX 等などで使える format に変換することが可能です．以下にその例をお見せしておきます.

```
> series(exp(-beta*(-g*x^3)),x,5);
  temp:=convert(%,polynom);
  f1:=unapply(exp(-beta*c*x^2)*temp,x);
  eqn1:=simplify(limit(int(f1(x),x=-alpha..alpha),alpha=infinity));
```

$$1 + \beta\, g\, x^3 + O(x^5)$$

$$temp := 1 + \beta\, g\, x^3$$

$$f1 := x \rightarrow \mathbf{e}^{(-\beta\, c\, x^2)} (1 + \beta\, g\, x^3)$$

$$eqn1 := \lim_{\alpha \rightarrow \infty} \frac{\sqrt{\pi}\ \mathrm{erf}(\alpha \sqrt{\beta\, c})}{\sqrt{\beta\, c}}$$

```
> readlib(C);
```

proc() ... **end**

```
> C(f1(x),optimized);
      t2 = x*x;
      t4 = exp(-beta*c*t2);
      t9 = (1.0+t2*x*g*beta)*t4;
> latex(f1(eqn1));
{e^{-\beta\,c\left (\lim _{\alpha\rightarrow \infty }{\frac {\sqrt {
\pi }{\it erf}(\alpha\,\sqrt {\beta\,c})}{\sqrt {\beta\,c}}}\right )^{
2}}}\left (1+\beta\,g\left (\lim _{\alpha\rightarrow \infty }{\frac {
\sqrt {\pi }{\it erf}(\alpha\,\sqrt {\beta\,c})}{\sqrt {\beta\,c}}}
\right )^{3}\right )
```

3）LaTeX,HTML 書類への保存

Maple で作った書類はメニューバーの File->ExportAs から

```
Plain Text...
Maple Text...
LaTeX...
HTML..
```

等に変換してくれます．本書の全てのスクリプトを HTML で

```
http://hightc.mtl.kyoto-u.ac.jp/Maple/Maple.index
```

から引けるようになっています.

索引

記号

.(ピリオド) 20
:(コロン) 1
;(セミコロン) 1
`(バッククォート) 20, 23
'(シングルクォート) 4, 14, 21
"(ダブルクォート) 4
[] 12
{} 12
20

A

add 19
adjoint 16, 48
angle 16, 48
args 20
array 14
assign 7, 35
assume 7, 35, 38, 47

B

break 18

C

C 50
Chebyshev 多項式 20
coeff 7, 47
coldim 16, 48
collect 7, 47
combine 7, 47
complex 8
conjugate 35
convert 7, 15, 47
convolution 27
crossprod 16, 48

D

debug 21
denom 7, 47
det 16, 41, 48
diff 9
dotprod 16, 48
dsolve 9, 40

E

eigenvals 16, 48
eigenvects 16, 48
evalf 5
expand 47

F

factor 7, 47
FFT 24
filter 1
for 17
Fortran 50
fsolve 45

G

gcd 7, 47

H

Hamiltonian 41

I

I 4
if 17
infinity 4
int 9
intersect 12
inverse 16, 48
isolve 32

L

LATEX 50
lcm 7, 47
limit 8, 38
linalg 15
list 12
listplot 23

M

Mandelbrot 集合 前書き i
map 15
matadd 16, 48
matrix 41, 42
mint 49
minus 12
multiply 42, 48

N

nargs 20
next 18
nops 13
normal 7, 47
numer 7, 47

O

op 12

P

package 15
Parseval の定理 25
Pi 4
plots 6
print 13
printf 20
printlevel 21
procedure 18
procname 20

R

readdata 23
readlib 5
recursive. 再帰的 *を参照*
remember 20
rowdim 16, 48

S

seq 13
series 38, 41
set 12
simplify 7, 35, 39, 47
solve 7, 41
sort 7, 47
subs 4
sum 8

T

table 13
Tayler 級数展開 10
trace 16, 48
transpose 15, 16, 48
type 15

U

unapply 5
union 12

V

vectdim 16, 48

W

while 17
with 5, 15

イ

因数分解 7, 47

エ

永年方程式 41
エラー処理 20

オ

オイラー角 42

カ

解 7
改行 1
外積 16, 48
角度 16, 48
関数 5
関数の plot 1, 21
関数の定義 5
簡単化 7, 47

キ

逆行列 16, 30, 48
級数 8, 10
行列 47
行列式 16, 48
極限 8
近似 38

ク

組み合わせ 47
組み込み関数 5

ケ

係数 7, 47
結合 12
減衰振動 46

コ

高速フーリエ変換 46
コメント 20
固有値 16, 48
固有ベクトル 16, 48
コロン 1

サ

差 12
再帰的 20
最小公倍数 7, 47
最大公約数 7, 47
三角関数 5

シ

式の変形 7, 33, 47, 48
実行の中断 2
　中断 2
自由エネルギー曲線 43
集合 12
初期化 4
初期条件 9

ス

随伴 16, 48
数式処理 4
複素数 4

セ

制御構造 17
整数関数 47
整数論 47
積 12, 48
積分 9, 38
セミコロン 1
線形代数 15, 41, 42

ソ

ソート 7, 47
総和 8

タ

対角和 16, 48
代入 4
多項式 47
畳み込み 26, 27

チ

調和振動子 38
直交多項式 47

テ

データ構造 12
データ処理 23
データフィット 27
定義関数の plot 21
テイラー級数展開 . Tayler級数展開 *を参照*
手続関数 18
デバッグ 21, 49
展開 7, 47
転置 16, 48

ト

統計学 47
同類項 7, 47
トレース 16, 48
トンネル効果 33

ナ

内積 16, 48

ニ

入出力 23, 49
入力 1

ネ

熱膨張 38

ハ

パーシバルの定理 25
配列 14

ヒ

微積分 8
非線形最小二乗法 27
微分 9
微分方程式 9
表 13

フ

フーリエ変換 24
フィルタ 24
複素関数 10
プログラミング 17
プロット 4, 5
分子 7, 47
分数の結合 7, 47
分母 7, 47

ヘ

ベクトル関数 47

ヘルプ 2

ホ

方程式の解 7
ボルツマン分布 38

マ

間違い修正 1

ム

無限大 4

ヤ

ヤコビ行列 28, 29

ユ

有理関数 47

ヨ

要素 13
陽電子消滅寿命 40
呼び出し 5

リ

リスト 12

ル

ループ 18

レ

連結作用素 20
連立微分方程式 40
連立方程式の数値解 43
連立方程式の整数解 32

ロ

ローレンツ型関数 27

ワ

和 12
和(行列) 16, 48