

# Maple V の基本操作と実践

第 2 版

西谷滋人

京都大学工学研究科材料工学科

1998年3月

Department of Materials Science and Engineering•

Kyoto University•Kyoto 606-8317, Japan

bob@karma.mtl.kyoto-u.ac.jp

Copyright ©1996-1998 by Shigeto R. Nishitani

本書はPowerMac 7600/120上でMaple V Release3 and 4とPageMaker6.5Jとを使用して編集，整形を行いました．

MapleとMaple VはWaterloo Maple Inc.の登録商標です．

PageMaker, Acrobatはアドビシステムズ社の登録商標です．

Apple, Macintosh, Power MacintoshはApple Computer, Inc.の登録商標です．

UNIXはAT&Tベル研究所の登録商標です．

X Window Systemはマサチューセッツ工科大学の登録商標です．

OSF/MotifとMotifはOpen Software Foundationの登録商標です．

---

Maple Vの基本操作と実践 第2版 (Maple V: Essentials and Applications on Materials Science)

Copyright © 1996-1998 by Shigeto R. Nishitani

Maple Vの基本操作と実践は以下の条件で配付することを許可します．

1． 著者に無断でのいかなる改変も許しません．ただし，ローカルなマシンへのインストール情報やライブラリーの追加に関する記述はこのかぎりではありません．

2． 本書あるいは本書の一部を商用の文書の一部として使うことは許しません．

ただし，本書に誤りや不正確な記述が書いてあった場合にも，著者はいかなる責任，債務は負わないものとします．

コピーしてどんどん広めて下さい．

## 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当たりにしたとき、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。普段、私が Maple をどのように使っているかという、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)

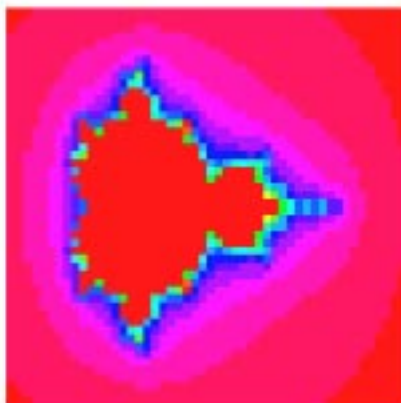
Mandelbrot (マンデルブロート) 集合 Mandelbrot set  
(中略)

```
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 <-- z2 - (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 (abs(z)<4.0 and count>0) do  
>   z:=z2-z0;  
>   count:=count-1;  
> od;  
> count;  
> end;  
> with(plots):  
> densityplot(mandel,-1..2.5,-1.5..1.5,axes=none,colorstyle=HUE,style=patchngrid,grid=[50,50]);
```



となります。どうです?! 数学的な記述そのまま、しかもわずかに数分で目的とする 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。Maple V に関する日本語の解説書が盛んに出版されています。一部を紹介します。

”よくわかる Maple”(Maple V Language Reference Manual の訳)

(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)

リリース3に基づいた上級者向き解説書。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。リリース4の対応する書籍は”Programming Guide”です。日本語訳は出ていませんが、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)

新しいリリース4の”Maple V Learning Guide”の訳。前の日本語版がいかにマニュアルの翻訳という雰囲気を読みにくかったのに対して、今回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。

”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, Mathematica) を出している著者による Maple V 解説書。多くの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。入門書としても MapleV の概観や基本的な機能を全て取り上げており、よくまとまっている。日本語の参考書としては現在もっとも充実している。

本書は Maple V Release 4 に基づいて解説しています。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、総合情報メディアセンターの辻斉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	最初の一步 .....	7
1 )	Maple の起動法 .....	7
2 )	簡単な演算 .....	7
3 )	間違い修正 .....	8
4 )	実行の中断 .....	8
5 )	ヘルプファイル .....	8
6 )	保存と終了 .....	9
第 2 章	簡単な数式処理とプロット .....	10
1 )	変数, 式の定義とそのキャンセル .....	10
2 )	関数 .....	11
3 )	プロット .....	12
4 )	方程式の解 .....	13
5 )	式の変形 .....	13
第 3 章	微積分とその応用 .....	14
1 )	総和の計算 .....	14
2 )	極限 .....	14
3 )	微分 .....	15
4 )	微分方程式 .....	15
5 )	積分 .....	16
6 )	級数 .....	16
7 )	複素関数 .....	16
第 4 章	データ構造と線形代数 .....	18
1 )	集合, リスト, 表, 配列 .....	18
2 )	線形代数 .....	21
第 5 章	MapleV でのプログラミング .....	23
1 )	プログラミングの流れ .....	23
2 )	Maple の制御構造 .....	23
3 )	プログラミングの実践 .....	24
4 )	その他のテクニック .....	27
第 6 章	データ処理 .....	29
1 )	データの入出力 .....	29
2 )	フーリエ変換 .....	30
3 )	畳み込み .....	33
4 )	非線形最小二乗法 .....	34
第 7 章	入出力に関する補足 .....	40
1 )	データの入出力 (フィルターとしての使用) .....	40
2 )	C, LATEX への出力 .....	41

実践演習 .....	43
実践演習[1] 化学反応式の係数決定 (連立方程式の整数解) .....	43
実践演習[2] トンネル効果 (式の変形) .....	44
実践演習[3] 熱膨張係数の導出 (複雑な関数の近似と積分) .....	51
実践演習[4] 陽電子消滅寿命 (連立微分方程式) .....	53
実践演習[5] Hamiltonian の解 (線形代数) .....	54
実践演習[6] オイラー角 (線形代数) .....	55
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) .....	56
実践演習[8] 減衰振動のフーリエ変換 (高速フーリエ変換) .....	59
付録A コマンドのまとめ .....	60
1) 関数パッケージ .....	60
2) 式の変形 .....	60
3) 線形代数 .....	60
索引 .....	62

---

---

# 第 1 章 最初の一步

---

---

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

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(第 7 章 1) データの入出力参照)。大型計算機センターのアプリケーションサーバ上には既にインストールされています。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 := { 2, 3, 1 }
```

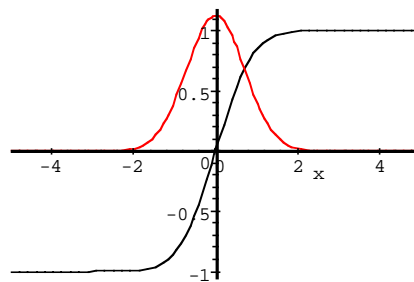
```
set2 := { 4, 3 }
```

最後の ; (セミコロン) を忘れがちです。セミコロンを忘れて enter を入れてしまったときには続けてセミコロン, enter を打ち込めば望む結果が得られます。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);
```

---

MapleV の京都大学におけるサイトライセンスは、1995 年度に工学部数理工学教室で取得されており、現在、複数の部局・教室などの協力のもとでライセンスが維持されています。学内ライセンスを持つ場合、数多くのマシン (Mac, DOS, Unix 等) に、ユーザ数の制限を受けずにインストールすることが許されています (<http://www.kuamp.kyoto-u.ac.jp/MapleV/> を参照下さい)。学内ライセンスに対する問い合わせは、maple-request@kuamp.kyoto-u.ac.jp まで。応用システム科学教室の河野先生 (当時、数理工学教室) が中心になって、対応しています。



となります。

### 3) 間違い修正

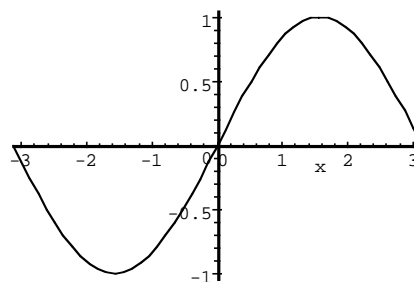
打ち間違いなどの訂正はアローキー ,あるいはマウスのクリックによってできます .後は訂正してenterを入れれば入力されます .ある領域を選択して ,削除・カットおよびペーストなどの作業もマウスを使ってできます .例えばサイン関数を- から+ までプロットしようとした時

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

*Error, (in plot) invalid range*

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 .Macでは通常の"apple+period"で ,Windows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の出力結果です .

```
> ?help;
```

FUNCTION: help - descriptions of syntax, datatypes, and functions      関数の名前です .

CALLING SEQUENCE:      入力方法や呼びだし方の簡単な記述です .

?topic or ?topic,subtopic or ?topic[subtopic] or  
help(topic) or help(topic,subtopic) or help(topic[subtopic])

SYNOPSIS:      概要です .

?intro      introduction to Maple  
?library      Maple library functions and procedures



?index list of all help categories  
?index,<category> list of help files on specific topics  
?<topic> explanation of a specific topic  
?<topic>,<subtopic> explanation of a subtopic under a topic  
?distribution for information on how to obtain Maple  
?copyright for information about copyrights

注意事項が続きます .

- Note 1: The recommended way to invoke help is to use the question mark.

- Note 2: When invoking help using the function call syntax, help(topic), Maple keywords (reserved words) must be enclosed in backquotes. For example, help(quit) causes a syntax error. Use help(`quit`) instead. Note that the string delimiter is the backquote (`), not the apostrophe ('), nor the double quote ("). When using the question mark syntax for help, no quotes are required.

- Note 3: A command must end with a semicolon, followed by RETURN or ENTER, before Maple will execute it and display the result. The semicolon can appear on the next line if you forget to end the command with it, but it must appear. There can be multiple commands on one line, separated by semicolons or colons. An exception to this is when a line starts with a question mark in which case help is invoked and no semicolon is required.

- To contact Waterloo Maple Software, see ?distribution. To contact the authors of Maple, see ?scg.

普通の関数ですと , この後に具体的な使用例が続きます .

SEE ALSO: keywords, quotes, colons, quit, example, scg, distribution, TEXT,

makehelp 関連するキーワードです . 単語をクリックすればそれに関する help が現れます .

## 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 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  $\text{Pi}$ ,  $\text{I}$ ,  $\text{infinity}$  などです．

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 " (ダブルクォート) を使います．

```
> expl:=";
```

```
expl := -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
```

## 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

?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

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。

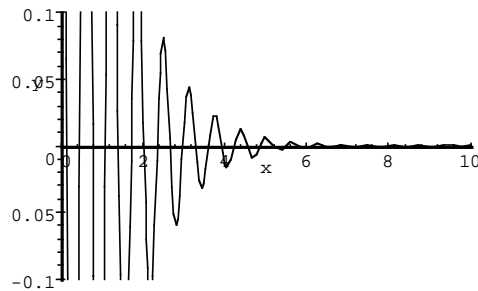
## 3) プロット

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);
```

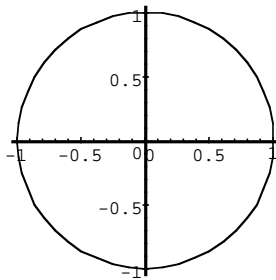
$$eq1 := x \rightarrow e^{-x} \cos(10x)$$

```
> 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]);
```

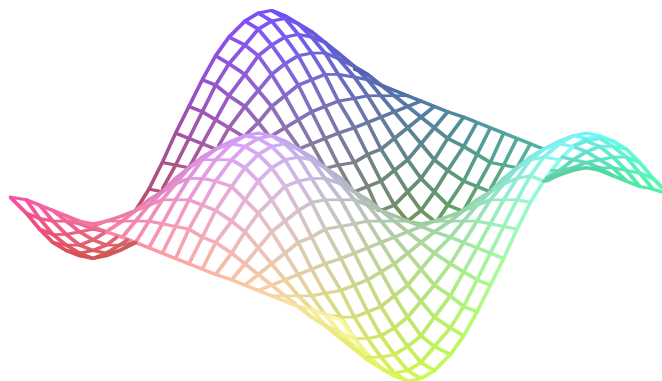


によって円が描けます。2変数の場合にはplot3dを使います。変数名が明らかな時には省略可能です。

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

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

```
> plot3d(f1,-Pi..Pi,-Pi..Pi);
```



この他にもオプションの指定や特殊なplotting法があります。特に

```
> 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});
```

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

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});
```

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

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

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

```
> 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 というものもあります .

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 .

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 ( 分子 )

denom - denominator of an expression ( 分母 )

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。(注：前章から続けて入力される方はこちら辺でrestartをかけてください。以下では前章で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)

### 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);`

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

> `sum(a^i,i=1..infinity);`

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

aはもちろん収束するとして求めてくれます。

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

1

∞

高校で習った知識がそのまま使えます。

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

2.718281828

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異なる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);`

$-\infty$

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

$\infty$

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);`

$2x$

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

$2y^2$

関数の一般形のままで形式的な微分を表示してくれます。

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

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

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

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

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

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

### 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;`

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

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

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

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));`

$$y(x) = e^{(-x)} a$$

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);`

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

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

という具合です。

## 6) 級数

Mapleで式のTaylor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);`

$$(.8775825619 + .4794255386 I) + (.8775825619 + .4794255386 I)(x - .5 I) +$$

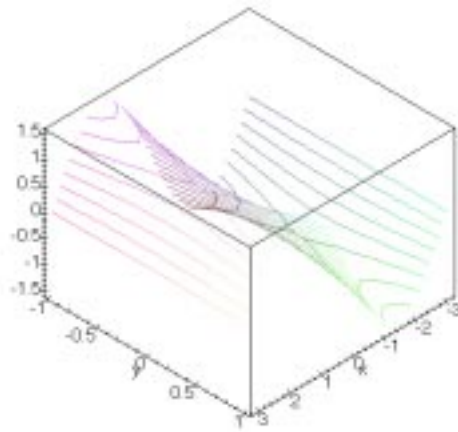
$$(.4387912810 + .2397127693 I)(x - .5 I)^2 + O((x - .5 I)^3)$$

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

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

`> plot3d(Re(sin(x+y*I)),x=-Pi..Pi,y=-1..1,axes=BOXED,style=contour);`





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 := {2, 3, 1}
```

```
set2 := {4, 3}
```

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

```
set3 := {4, 2, 3, 1}
```

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

```
set4 := {3}
```

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

```
set5 := {2, 1}
```

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、

>  $list3[5] := x;$

$list3_5 := x$

です。全部を表示してみると

>  $list3;$

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

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。

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

$atomicweight := table([  
Fe = 56  
C = 12  
])$

>  $atomicweight[Fe];$

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。

>  $print(atomicweight);$

```
table([
  Fe = 55.847
  Al = 27
  C = 12
])
```

削除は

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

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

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

です。( " ちゃん " 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(antisymmetric,1..3,1..3);
```

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

```
> A:=array(symmetric,1..3,1..3);
```

```
      A := array(symmetric, 1 .. 3, 1 .. 3, [ ])
```

```
> print(A);
```

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{1,2} & A_{2,2} & A_{2,3} \\ A_{1,3} & A_{2,3} & A_{3,3} \end{bmatrix}$$

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わからなくなってしまうことがあります。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(name,list);**

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。

> **convert(list1,set);**

**{ 27, 34, 85, 65 }**

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,list1);**

**[ sin(65), sin(34), sin(27), sin(85) ]**

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(2,2,[1,2,3,4]);**

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

> **vec:=vector([x,y]);**

$$vec := [x \quad y]$$

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、

> **ma2:=evalm(ma^2);**

$$ma2 := \begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$$

> **evalm(ma+ma2);**

$$\begin{bmatrix} 8 & 12 \\ 18 & 26 \end{bmatrix}$$

> evalm(ma &\* vec);

$$[x + 2y \quad 3x + 4y]$$

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。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[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[eigenvecs] - 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。詳しい使用例は第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。

”ライトついで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た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 ( 手順関数 ) のひな型です .

```

name := proc(input sequences)
local variable sequence;
global variable sequence;
options option sequence;
description descripition 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:=0.0;
> 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 := 243
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。これら为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 arguments`) fi;
> n:=nops(data1);
> ave:=add(list1[i],i=1..n)/n;
> disp:=0.0;
> for i from 1 to n do
> disp := disp+(data1[i]-ave)^2
> od;
> disp:=disp/n;
> printf(`Average = %10.5f\n`,ave);
> printf(`Dispersion = %10.5f\n`,disp);
```

---

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。

```

> disp:=sqrt(evalf(disp));
> printf(" Standard mean = %10.5f\n",disp);
> end;
> stat(list1);
Average = 61.00000
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。文字列はバック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

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x) \quad \text{for } n \geq 2$$

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;

```

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;

```

$f0 := 1$

```

> 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。(大げさすぎて...第7章で解説しています)。さらに実行速度が問題になるときは `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 "はじめての MapleV", "よくわかる MapleV" 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

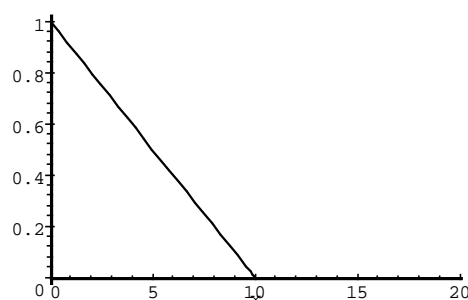
**> filter:=x->if x<10 then 1-x/10 else 0 fi;**

*filter :=*

*proc(x) options 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では最初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。

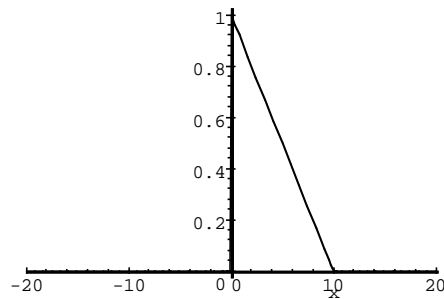
#### 階段関数に関する注意

プロットだけでなくfourier変換などでは上のような階段関数の積分が必要になってきます。上述の関数はsingularityがあるためにうまく積分ができません。そこで不連続あるいは滑らかでない関数を定義するときには組み込み関数であるHeavisideあるいはpiecewiseを使ってください。上の関数は次のように定義できます。

```
> h:=x->(1-x/10)*Heaviside(x)*Heaviside(10-x);
```

$$h := x \rightarrow \left( 1 - \frac{1}{10}x \right) \text{Heaviside}(x) \text{Heaviside}(10 - x)$$

```
> plot(h(x),x=-20..20);
```



(注：どちらを使っても次章で扱うconvolutionを計算するとplot表示がおかしくなります。どうやらこれはバグのようです。)

---



---

## 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립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。

> **T:=readdata("DATA101",1):**

コロンで出力を抑制しています(脚注)。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];**

**.1108250000000000 10<sup>-12</sup>**

> **n:=nops(T);**

**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);**

---

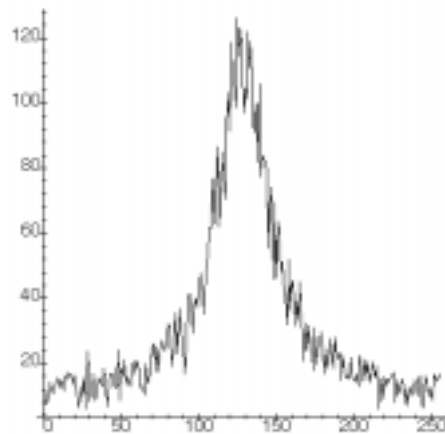
注) データは例えば、

**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):**

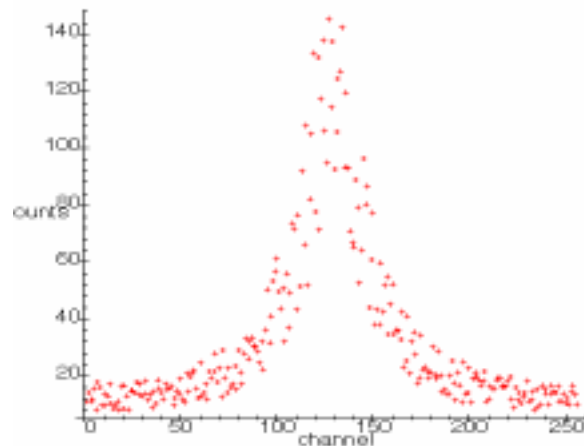
> **writedata("DATA104",convert(T,list));**

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 .

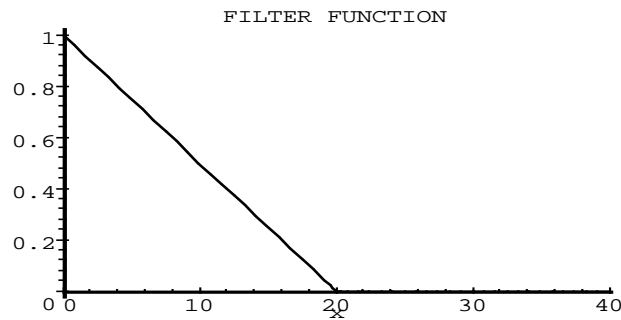
```
> datapoint:=[seq( [i,T[i]],i=1..256)];
> plot(datapoint,style=POINT);
```



## 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).周波数空間の位相 $\tan^{-1}(b_n/a_n)$ のデータ128チャンネルをプロットせよ .はじめのデータの面積強度とフーリエ逆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

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

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関数に似ており、そのフーリエ変換は全周波数成分を一樣に含んでいます。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

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

することによってフィルタ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 (平坦部のある 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('DATA101',1);
> Idata:=array([seq(0,i=1..256)]);
> Rdata:=array([seq(T[i],i=1..256)]);
```

全パワーを求めておきましょう。

```
> temp:=add(i^2,i=T);
512973.3434
```

実際に FFT を実行します。

```
> readlib(FFT);
proc(m,x,y) ... 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;

```

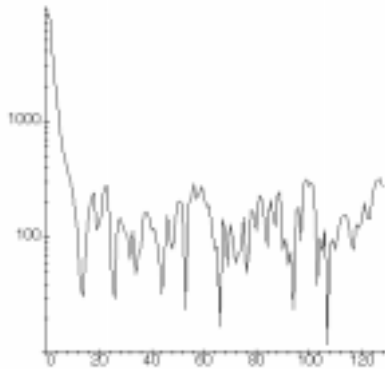
1	8373.62739	0.00000		
2	-4114.57495	-32.01013	-4114.57495	32.01014
3	2535.41239	114.76556	2535.41239	-114.76556
4	-1608.19700	-82.14135	-1608.19700	82.14135
5	936.67621	66.38215	936.67621	-66.38215
		.....		
124	-7.72419	232.68948	-7.72419	-232.68948
125	108.89917	-262.61238	108.89917	262.61238
126	-153.40139	262.48981	-153.40139	-262.48980
127	280.10101	-152.14228	280.10101	152.14228
128	-261.51641	68.62209	-261.51641	-68.62209

波数空間での強度をログで見てください。

```

> Adata:=[seq([i,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;
512973.34

```

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。前章の最後に述べた注意を参考にし、

```

> filter:=x->Heaviside(x)*Heaviside(20-x)*(1-x/20);

```

$$filter := x \rightarrow Heaviside(x) Heaviside(20 - x) \left( 1 - \frac{1}{20} x \right)$$

```

> 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、

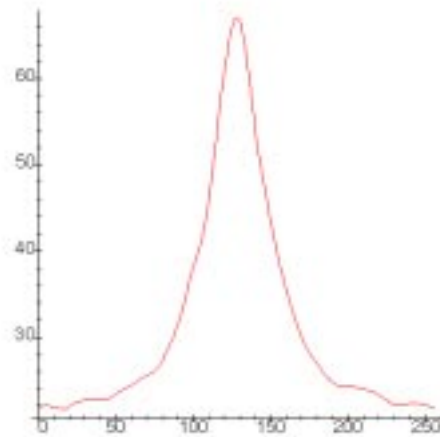
```

> iFFT(8,FRdata,FIdata);

```



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



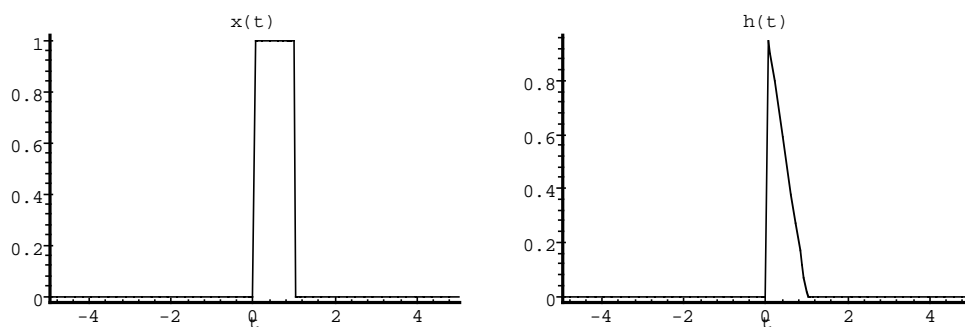
ノイズが取り除かれているのが分かるでしょう。

### 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 $h(t)$ で表される時、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。畳み込みを問

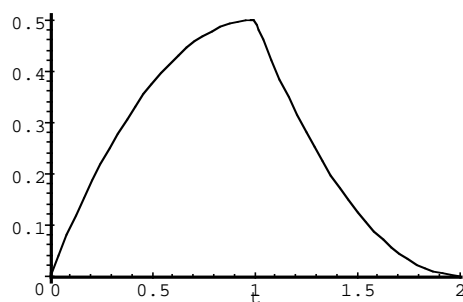
単に  $f = x * h$  と表すと, そのフーリエ変換は

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

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->Heaviside(x)*Heaviside(1-x);
      fun := x → Heaviside(x) Heaviside(1 - x)
> h:=x->(1-x)*Heaviside(x)*Heaviside(1-x);
      h := x → (1 - x) Heaviside(x) 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,

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

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,

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

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、

$$\begin{pmatrix} \Delta f_1 \\ \Delta f_2 \\ \vdots \\ \Delta f_{256} \end{pmatrix} = \mathbf{J} \begin{pmatrix} \Delta a_1 \\ \Delta b_1 \\ \Delta c_1 \\ \Delta d_1 \end{pmatrix} \tag{6.3}$$

となります。ここで4行256列の行列

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

はヤコビ行列と呼ばれる行列です。逆行列 $\mathbf{J}^{-1}$ は転置行列 $\mathbf{J}'$ を用いて

$$\mathbf{J}^{-1} = (\mathbf{J}'\mathbf{J})^{-1}\mathbf{J}'$$

と表されます。従って真値からのずれは

$$\begin{pmatrix} \Delta a_2 \\ \Delta b_2 \\ \Delta c_2 \\ \Delta d_2 \end{pmatrix} = (\mathbf{J}'\mathbf{J})^{-1}\mathbf{J}' \begin{pmatrix} \Delta f_1 \\ \Delta f_2 \\ \vdots \\ \Delta f_{256} \end{pmatrix} \tag{6.4}$$

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('DATA101',1):`

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

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);`

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

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

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);
```

$$dfda := 1$$

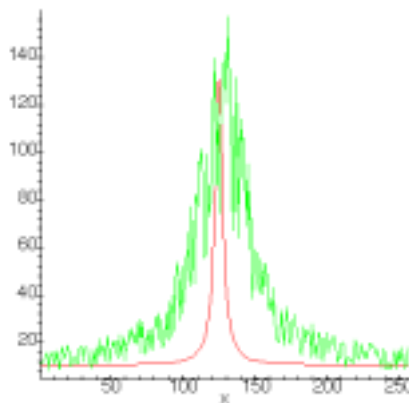
$$dfdb := \frac{1}{x \rightarrow c + (x - d)^2}$$

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

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

初期値を仮定します .

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



$\Delta f_i$  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 , 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 := array(sparse, 1 .. 256, 1 .. 4, [ ])

```

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));

```

最後に新たなパラメータの組を求めます .

```

> guess2:=multiply(tJac,df);
guess2 := [12.69373203, 4132.316868, 42.34612274, .6167914678]
> 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 := { d = 125.6167915, c = 52.34612274, b = 5332.316868, a = 22.69373203 }

```

```

> guess1:=guess3:

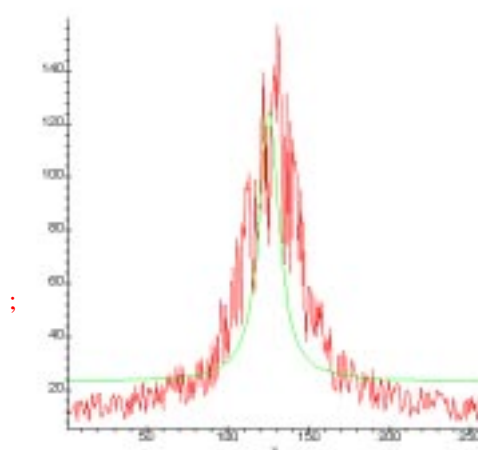
```

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);

```



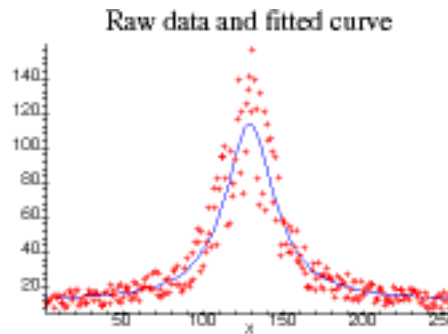
$\Delta f_i$  以下の操作を4回繰り返した後の結果です . データにほぼフィットした結果が得られています . 見やすくするために出力を少し工夫してあります .

```

> print(guess1);
{ d = 128.1222151, c = 339.5709538, b = 36778.33839, a = 10.31258466 }
> 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<sup>2</sup>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  $x$  がデータの横軸、 $a$  がパラメータの組とすると、

$$y = f(x; a)$$

です。最小二乗法の<sup>2</sup>関数は

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

です。後の式を単純にするために  $y_i - f(x_i; a) \Rightarrow f(x_i; a)$  と置き換え、あらかじめ実験値を引いておきます。S が安定であるための条件は

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

ヤコビ行列を  $J_i(a) = [\partial f_i / \partial a_j]$  と表すと

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

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

$$\nabla^2 S(a) \Delta a^{(k)} = -\nabla S(a^{(k)})$$

となります。S(a) のあらわな 2 次微分の表現は

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

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

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

が得られます。鉤括弧内の第2項を無視すると、先程の(6.4)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  $f(x_i; \mathbf{a})$  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はなく、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。

---



---

## 第7章 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、

```
input.data > maple -f action.txt >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。

```
Challenge# cat test.txt
readlib(readdata):
T:=readdata(`DATA101`,float,1):
n:=nops(T):
oldinter:=interface(quiet,quiet=true);
writeto(`out.data`);
for i from 1 to n do printf(`%10.5f\n`,T[i]); od;
writeto(terminal);
interface(quiet=oldinter);
quit:
```

ここでのキーはwritetoとinterfaceです。writeto(file\_name)でそれ以降の出力をファイルへ書きだし、writeto(terminal)でterminalへ出力するもとの状態へ戻しています。これだけですとスクリプト自身もout.dataへ出力されてしまいます。これを抑制するためにinterface(quiet=true)で結果だけを出力するモードに切り替えています。

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ました。上記のスクリプトにはバグが潜んでいます。

```
Challenge# mint test.txt
```

```

  |^^|      Maple V Release 3 Diagnostic Program
._|\|    |/_|. Copyright (c) 1981-1994 by Waterloo Maple Software
 \  MINT  /   and the University of Waterloo. All rights reserved.
 <____ >     Maple and Maple V are registered trademarks of
 |           Waterloo Maple Software.
9: quit:
  ^ syntax error
This name on line 2 does not need string quotes(``): DATA101
```



少し分かりにくいですが, mint はひとつの syntax error と warning を出しています. syntax error は printf 文の後ろ側のちょんがバックコートではなくシングルコートになっている事による error です. warning の方は, この場所のファイルを読むのにバックコートはいらないよという注意です. これらを訂正して,

```
Challenge# cat test.txt
readlib(readdata):
T:=readdata(`./DATA101`,float,1):
n:=nops(T):
oldinter:=interface(quiet,quiet=true);
writeto(`out.data`);
for i from 1 to n do printf(`%10.5f\n`,T[i]); od;
writeto(terminal);
interface(quiet=oldinter);
quit:
```

```
Challenge# mint test.txt
```

```
This name on line 2 does not need string quotes(``): DATA101
  | ^ |      Maple V Release 3 Diagnostic Program
._| | _   | / | _ . Copyright (c) 1981-1994 by Waterloo Maple Software
 \ MINT /   and the University of Waterloo. All rights reserved.
 < ___ ___ > Maple and Maple V are registered trademarks of
   |
   |      Waterloo Maple Software.
```

と全くエラーを返しません. 走らせてみると,

```
Challenge# maple < test.txt
```

```
  | ^ |      Maple V Release 3 (Kyoto University)
._| | _   | / | _ . Copyright (c) 1981-1994 by Waterloo Maple Software and the
 \ MAPLE /   University of Waterloo. All rights reserved. Maple and Maple V
 < ___ ___ > are registered trademarks of Waterloo Maple Software.
   |
   |      Type ? for help.
```

```
> readlib(readdata):
> T:=readdata(`./DATA101`,float,1):
> n:=nops(T):
> oldinter:=interface(quiet,quiet=true);
                                oldinter := false
```

```
> quit:
bytes used=254200, alloc=262096, time=0.10
```

```
Challenge# wc out.data
```

```
      256          256          2816 out.data
```

めでたく出力されています.

(Release 4 では write も改善されて writedata(`filename`,data) だけで出力されるようになりました.)

## 2) C, LATEX への出力

Maple で作った式や関数などは, Fortran や C, LATEX 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:=limit(normal(int(f1(x),x=-alpha..alpha)),alpha=infinity);

```

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

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

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

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

```

> readlib(C);
proc() ... end
> C(f1(x),optimized);
t2 = x*x;
t9 = exp(-beta*c*t2)*(1.0+beta*g*t2*x);
> latex(eqn1);
\lim_{\alpha \rightarrow \infty} \frac{\sqrt{\pi} \operatorname{erf}(\sqrt{\beta c} \alpha)}{\sqrt{\beta c}}

```

---



---

## 実践演習

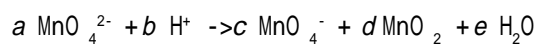
---



---

実践演習[ 1 ] 化学反応式の係数決定( 連立方程式の整数解 )

次の反応式



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;

$$\begin{aligned} Mn &:= a = c + d \\ OO &:= 4a = 4c + 2d + e \\ H &:= b = 2e \\ Q &:= -2a + b = -c \end{aligned}$$

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

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

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

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 .

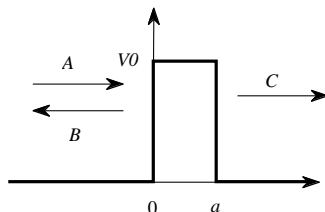
> subs(\\_N1=1,ans);

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

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—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させていただき , 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

$$-\frac{\hbar^2}{2m} \frac{d^2\phi(x)}{dx^2} = \epsilon \phi(x)$$

ですから , 波動関数は

$$\begin{aligned} x \leq 0 \text{ では} & \quad \phi(x) = A \exp(ikx) + B \exp(-ikx) \\ x \geq a \text{ では} & \quad \phi(x) = C \exp(ikx) \end{aligned}$$

ここで  $k = \sqrt{2m\epsilon}/\hbar$  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  $\epsilon \leq V_0$  によって事情が変わってきます  $\epsilon \geq V_0$  では  $\kappa = \sqrt{2m(\epsilon - V_0)}/\hbar$  と定義すると , 波動関数は

$$0 \leq x \leq a \text{ では} \quad \phi(x) = F \exp(ikx) + G \exp(-ikx)$$

となります . 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 という条件を  $x = 0$  と  $x = a$  に適用すると , 条件はそれぞれ

$$x = 0 \text{ で } \phi(x) \text{ が連続 : } A + B = F + G$$

$$x = 0 \text{ で } \phi'(x) \text{ が連続 : } k(A - B) = \kappa(F - G)$$

$$x = a \text{ で } \phi(x) \text{ が連続 : } F \exp(ika) + G \exp(-ika) = C \exp(ika)$$

$$x = a \text{ で } \phi'(x) \text{ が連続 : } \kappa F \exp(ika) - \kappa G \exp(-ika) = k C \exp(ika)$$

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

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

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

となります。

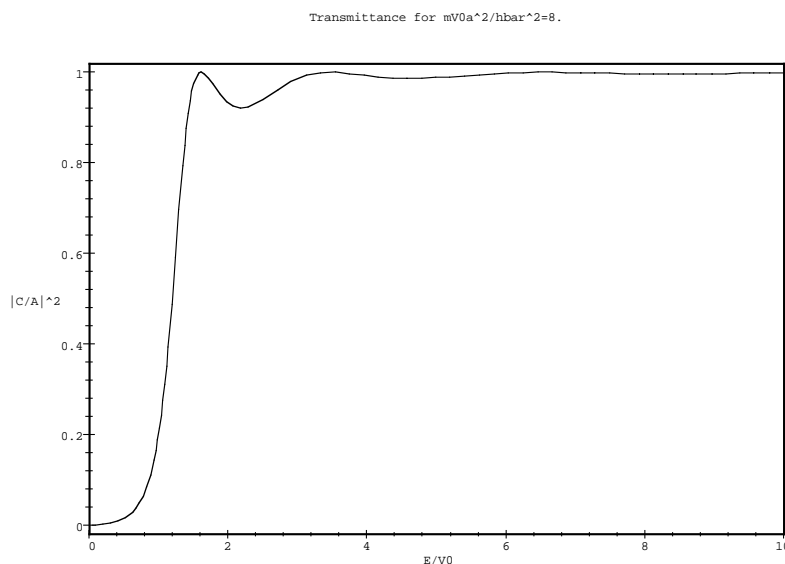
$0 < \epsilon < V_0$  では  $\alpha = \sqrt{2m(V_0 - \epsilon)}/\hbar$  として、波動関数は

$$0 \leq x \leq a \text{ では } \varphi(x) = F \exp(\alpha x) + G \exp(-\alpha x)$$

です。同様な計算によって

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

となります  $mV_0 a^2 / \hbar^2 = 8$  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、

```
> 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);
```

$$\begin{aligned} eq1 &:= A + B = F + G \\ eq2 &:= k(A - B) = \kappa(F - G) \\ eq3 &:= F e^{(I \kappa a)} + G e^{(-I \kappa a)} = C e^{(I k a)} \\ eq4 &:= \kappa F e^{(I \kappa a)} - \kappa G e^{(-I \kappa a)} = k C e^{(I k a)} \end{aligned}$$

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});

> 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;

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

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

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

> assign(sol2);

> A/C;

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

> A/B;

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

でほぼ求まっています．ここからの式の単純化は苦勞します．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．

まずはB/Aで計算を進めていきます．

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

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

分母と分子を別々に扱います．

> num\_BB:=numer(BB);

den\_BB:=denom(BB);

$$\begin{aligned} \text{num\_BB} &:= \kappa^2 \sin(a \kappa) + 2 I k \kappa \cos(a \kappa) + k^2 \sin(a \kappa) \\ \text{den\_BB} &:= \sin(a \kappa) (-\kappa^2 + k^2) \end{aligned}$$

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(num\_BB);

$$\text{conjugate}(\kappa^2 \sin(a \kappa) + 2 I k \kappa \cos(a \kappa) + k^2 \sin(a \kappa))$$

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，虚数項と実数項を分けて計算を進めます．

> Rnum\_BB:=coeff(num\_BB,I,0);

Inum\_BB:=coeff(num\_BB,I,1);

$$\begin{aligned} \text{Rnum\_BB} &:= \kappa^2 \sin(a \kappa) + k^2 \sin(a \kappa) \\ \text{Inum\_BB} &:= 2 k \kappa \cos(a \kappa) \end{aligned}$$

絶対値の二乗を求めます．

> BB2:=(Rnum\_BB+I\*Inum\_BB)\*(Rnum\_BB-I\*Inum\_BB);

$$\text{BB2} := \begin{pmatrix} \kappa^2 \sin(a \kappa) + 2 I k \kappa \cos(a \kappa) + k^2 \sin(a \kappa) \\ \kappa^2 \sin(a \kappa) + k^2 \sin(a \kappa) - 2 I k \kappa \cos(a \kappa) \end{pmatrix}$$

cos^2=1-sin^2 を適用し，分母を二乗した数で割っておきます．

> final:=collect(subs(cos(a\*kappa)^2=1-sin(a\*kappa)^2,expand(BB2))/den\_BB^2,sin(a\*kappa));

$$\text{final} := \frac{\kappa^4 - 2 \kappa^2 k^2 + k^4}{(-\kappa^2 + k^2)^2} + 4 \frac{k^2 \kappa^2}{\sin(a \kappa)^2 (-\kappa^2 + k^2)^2}$$

第一項を取りだし，簡単化します．

> coeff(final,sin(a\*kappa),0);

$$\frac{\kappa^4 - 2 \kappa^2 k^2 + k^4}{(-\kappa^2 + k^2)^2}$$

> normal(expand(""));

1

で大体終わりです．

別の解答を以下で示します．前の実践例で問題になるのは 絶対値の二乗を求める conjugate が効かないところです．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．しばらくは同じ入力ですが，出力に ~ がついており，変数に assume がかかっていることを教えてくれます．

> eq1:=A+B=F+G;

eq2:=k\*(A-B)=kappa\*(F-G);

$$\text{eq3} := F * \exp(I * \kappa * a) + G * \exp(-I * \kappa * a) = C * \exp(I * k * a);$$

$$\text{eq4} := \kappa * F * \exp(I * \kappa * a) - \kappa * G * \exp(-I * \kappa * a) = k * C * \exp(I * k * a);$$

$$\text{eq1} := A + B = F + G$$

$$\text{eq2} := k (A - B) = \kappa (F - G)$$

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

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

> sol1:=solve({eq1,eq2},{F,G});

$$\text{sol1} := \left\{ F = \frac{1}{2} \frac{k A - k B + \kappa A + \kappa B}{\kappa}, G = -\frac{1}{2} \frac{-\kappa A - \kappa B + k A - k B}{\kappa} \right\}$$

> assign(sol1);

> sol2:=solve({eq3,eq4},{B,C});

$$\text{sol2} := \left\{ C = -4 A e^{(I \kappa a)} k e^{(-I \kappa a)} \kappa \left/ \left( e^{(I k a)} \left( -2 e^{(I \kappa a)} k \kappa + e^{(I \kappa a)} \kappa^2 - e^{(-I \kappa a)} \kappa^2 - 2 e^{(-I \kappa a)} k \kappa + e^{(I \kappa a)} k^2 - e^{(-I \kappa a)} k^2 \right) \right) \right. \right. \\ \left. \left. \left( -e^{(I \kappa a)} \kappa^2 + e^{(I \kappa a)} k^2 - e^{(-I \kappa a)} k^2 + e^{(-I \kappa a)} \kappa^2 \right) \right) \right. \\ \left. \left/ \left( -2 e^{(I \kappa a)} k \kappa + e^{(I \kappa a)} \kappa^2 - e^{(-I \kappa a)} \kappa^2 - 2 e^{(-I \kappa a)} k \kappa + e^{(I \kappa a)} k^2 - e^{(-I \kappa a)} k^2 \right) \right. \right\}$$

> assign(sol2);

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

$$\text{BB2} := 4 \frac{k^2 \kappa^2 \cos(\kappa a)^2}{\sin(\kappa a)^2 (-\kappa^2 + k^2)^2} + \frac{\kappa^4}{(-\kappa^2 + k^2)^2} + 2 \frac{\kappa^2 k^2}{(-\kappa^2 + k^2)^2} \\ + \frac{k^4}{(-\kappa^2 + k^2)^2}$$

> normal(subs(cos(kappa\*a)^2=1-sin(kappa\*a)^2,BB2));

$$\left( 4 \kappa^2 k^2 - 2 \kappa^2 k^2 \sin(\kappa a)^2 + \kappa^4 \sin(\kappa a)^2 + k^4 \sin(\kappa a)^2 \right) \left/ \left( \sin(\kappa a)^2 (-\kappa^2 + k^2)^2 \right) \right.$$

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



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

> CC2:=expand(conjugate(CC)\*CC);

$$CC2 := \frac{1}{4} \frac{\kappa \sim^2 \cos(k \sim a \sim)^2 \sin(\kappa \sim a \sim)^2}{k \sim^2} + \frac{1}{4} \frac{k \sim^2 \cos(k \sim a \sim)^2 \sin(\kappa \sim a \sim)^2}{\kappa \sim^2} \\ + \frac{1}{4} \frac{k \sim^2 \sin(k \sim a \sim)^2 \sin(\kappa \sim a \sim)^2}{\kappa \sim^2} + \frac{1}{4} \frac{\kappa \sim^2 \sin(k \sim a \sim)^2 \sin(\kappa \sim a \sim)^2}{k \sim^2} \\ + \cos(k \sim a \sim)^2 \cos(\kappa \sim a \sim)^2 + \frac{1}{2} \cos(k \sim a \sim)^2 \sin(\kappa \sim a \sim)^2 \\ + \sin(k \sim a \sim)^2 \cos(\kappa \sim a \sim)^2 + \frac{1}{2} \sin(k \sim a \sim)^2 \sin(\kappa \sim a \sim)^2$$

> subs(cos(k\*a)^2=1-sin(k\*a)^2,cos(kappa\*a)^2=1-cos(kapp\*a)^2,CC2);

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

> normal(subs(cos(k\*a)^2=1-sin(k\*a)^2,cos(kappa\*a)^2=1-cos(kappa\*a)^2,CC2));

$$\frac{1}{4} \left( \kappa \sim^4 \sin(\kappa \sim a \sim)^2 + k \sim^4 \sin(\kappa \sim a \sim)^2 + 4 \kappa \sim^2 k \sim^2 - 4 \kappa \sim^2 k \sim^2 \cos(kapp a \sim)^2 \right. \\ \left. + 2 \kappa \sim^2 k \sim^2 \sin(\kappa \sim a \sim)^2 \right) / (\kappa \sim^2 k \sim^2)$$

> normal(subs(cos(k\*a)^2=1-sin(k\*a)^2,cos(kappa\*a)^2=1-cos(kappa\*a)^2,CC2));

$$\frac{1}{4} \left( \kappa \sim^4 \sin(\kappa \sim a \sim)^2 + k \sim^4 \sin(\kappa \sim a \sim)^2 + 4 \kappa \sim^2 k \sim^2 - 4 k \sim^2 \kappa \sim^2 \cos(\kappa \sim a \sim)^2 \right. \\ \left. + 2 \kappa \sim^2 k \sim^2 \sin(\kappa \sim a \sim)^2 \right) / (\kappa \sim^2 k \sim^2)$$

> normal(subs(cos(k\*a)^2=1-sin(k\*a)^2,cos(kappa\*a)^2=1-sin(kappa\*a)^2,CC2));

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

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);

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

これだけではプロットできません．つぎに m を消去することを試みます．

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

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

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

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

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

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

これを単純化するうまい方法が見つけれませんでした．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)))));

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

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));

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

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;

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

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

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 関数でユーザ関数とします．

> CC3:=unapply((1+num\_f1/den\_f1)^(-1),x);

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

> 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．labels 内の "ちゃん" はバッククォートです．

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められます . 平均の位置  $x$  は , 熱力学的な確率で重みづけられ ,

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

で計算できます . ここで  $\beta = 1 / (k_B T)$  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);

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

- から 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(normal(int(f1(x),x=-alpha..alpha)),alpha=infinity);

$$\lim_{\alpha \rightarrow \infty} \frac{\sqrt{\pi} \operatorname{erf}(\sqrt{\beta c} \alpha)}{\sqrt{\beta c}}$$

うまく計算してくれません . これは  $\beta c$  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よう . 先ず分母は ,

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

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

次に分子は

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

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

です．これを取りだして

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

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

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

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

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 .

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

さらに

$$\left( \frac{\partial}{\partial t} n1(t) \right) + (\lambda1 + K12) n1(t) = 0 \tag{2a}$$

$$\left( \frac{\partial}{\partial t} n2(t) \right) + \lambda2 n2(t) = K12 n1(t) \tag{2b}$$

の連立微分方程式を  $n1(0) = n0, n2(0) = 0$  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 "PositroninSolids", edbyP.Hautojarvi, Springer-Verlag, Berlin1979 )

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

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

> dsolve(deq,n(t));

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

> 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};

$$deq := \left\{ n1(0) = n0, \left( \frac{\partial}{\partial t} n1(t) \right) + (\lambda1 + K12) n1(t) = 0, n2(0) = 0, \left( \frac{\partial}{\partial t} n2(t) \right) + \lambda2 n2(t) = K12 n1(t) \right\}$$

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

$$\left\{ n1(t) = - \frac{(-K12 + \lambda2 - \lambda1) n0 e^{-(\lambda1 + K12) t}}{K12 - \lambda2 + \lambda1}, n2(t) = \frac{n0 K12 e^{(-\lambda2 t)}}{K12 - \lambda2 + \lambda1} - \frac{n0 K12 e^{-(\lambda1 + K12) t}}{K12 - \lambda2 + \lambda1} \right\}$$

## 実践演習[ 5 ] Hamiltonian の解 (線形代数)

異核 s 価電子ポンド 2 量体の原子軌道の線形結合から求めた永年方程式は

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

である . ここで  $\Delta E = E_A - E_B$  は原子エネルギー準位のずれ  $\bar{E} = \frac{1}{2}(E_A + E_B)$  ,  $h$  と  $S$  とはそれぞれ軌道間のポ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

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

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);**

$$sol1 := -\frac{1}{4}DE^2 + E1^2 - 2E1Eav + Eav^2 - h^2 + 2hSE1 - 2hSEav - S^2E1^2 + 2S^2E1Eav - S^2Eav^2$$

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

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

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

> **Eplus:="[1];**

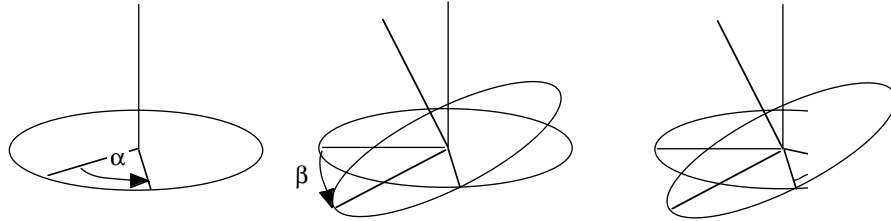
$$Eplus := \frac{1}{2} \frac{2Eav - 2hS - 2S^2Eav + \sqrt{DE^2 + 4h^2 - S^2DE^2}}{1 - S^2}$$

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

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

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]);

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

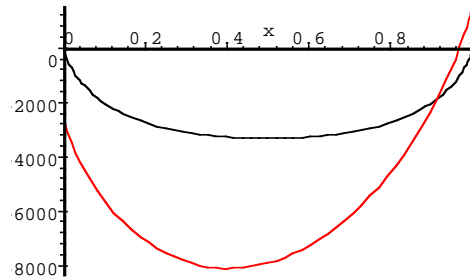
> Y:=matrix(3,3,[ct,0,st,0,1,0,-st,0,ct]);

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

> 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成を求める．
- 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

$$> G_s := (x_b, temp) \rightarrow (1-x_b) \mu_{as} + x_b \mu_{bs} + dG_{ms}(x_b, temp);$$

$$> G_l := (x_b, temp) \rightarrow (1-x_b) \mu_{al} + x_b \mu_{bl} + dG_{ml}(x_b, temp);$$

$$G_s := (x_b, temp) \rightarrow (1-x_b) \mu_{as} + x_b \mu_{bs} + dG_{ms}(x_b, temp)$$

$$G_l := (x_b, temp) \rightarrow (1-x_b) \mu_{al}(temp) + x_b \mu_{bl}(temp) + dG_{ml}(x_b, temp)$$

です．この中で  $dG_{ms}$ ,  $dG_{ml}$  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，

$$> dG_{ms} := (x_b, temp) \rightarrow \Omega_{gas} (1-x_b) x_b + RR \cdot temp (x_b \ln(x_b) + (1-x_b) \ln(1-x_b));$$

$$> dG_{ml} := (x_b, temp) \rightarrow \Omega_{gal} (1-x_b) x_b + RR \cdot temp (x_b \ln(x_b) + (1-x_b) \ln(1-x_b));$$

$$dG_{ms} := (x_b, temp) \rightarrow$$

$$\Omega_{gas} (1-x_b) x_b + RR \cdot temp (x_b \ln(x_b) + (1-x_b) \ln(1-x_b))$$

$$dG_{ml} :=$$

$$(x_b, temp) \rightarrow \Omega_{gal} (1-x_b) x_b + RR \cdot temp (x_b \ln(x_b) + (1-x_b) \ln(1-x_b))$$

となります．これらの式の中で使われている定数と変数を以下のようにとります．

$$> RR := 8.314;$$

$$\Omega_{gas} := 16.7 \cdot 1000; \mu_{as} := 0; \mu_{bs} := 0;$$

$$\Omega_{gal} := 0;$$

$$dH_a := 8.37 \cdot 1000; dH_b := 12.55 \cdot 1000;$$

$$dS_a := dH_a / 1000; dS_b := dH_b / 1500;$$



mu<sub>al</sub> と mu<sub>bl</sub> 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性を仮定します .

`mual:=temp->dHa-temp*dSa;`

`mubl:=temp->dHb-temp*dSb;`

$$mu_{al} := temp \rightarrow dHa - temp dSa$$

$$mu_{bl} := temp \rightarrow dHb - temp dSb$$

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 .

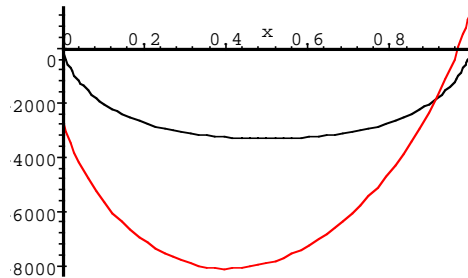
`> Gl:= $(x_b, temp) \rightarrow (1-x_b) * mu_{al}(temp) + x_b * mu_{bl}(temp) + dG_{ml}(x_b, temp)$ ;`

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 .

`> temp1:=1300;`

`plot({Gl(x,temp1),Gs(x,temp1)}`

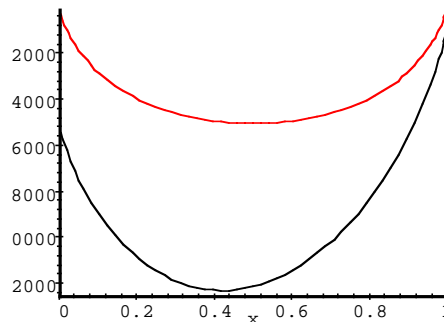
`,x=0..1);`



`> temp1:=1600;`

`plot({Gl(x,temp1),Gs(x,temp1)},x=0..1);`

*temp1 := 1600*



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);`

$$4180.00 + .0033333333 temp + 8.314 temp (\ln(y) - \ln(1 - y))$$

まず傾きの差を

`> Fl:= $(x,y,temp) \rightarrow \text{subs}(x1=x,y1=y,\text{diff}(G_s(x1,temp),x1)-\text{diff}(G_l(y1,temp),y1))$ ;`

$$F_l := (x, y, temp) \rightarrow \text{subs} \left( x1 = x, y1 = y, \left( \frac{\partial}{\partial x1} G_s(x1, temp) \right) - \left( \frac{\partial}{\partial y1} G_l(y1, temp) \right) \right)$$

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));
```

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

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});
{x = .9685370576, y = .8308884633 }
```

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 -> fsolve( { F1(x, y, temp) = 0, F2(x, y, temp) = 0 }, { x, y },
{x = .5 .. .99, y = .5 .. .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 800
do
  print(temp);
  FindXY(temp);od;
1400
{x = .9843946402, y = .9164387422}
1300
{y = .8308884633, x = .9685370576}
1200
{y = .7434269425, x = .9524858191}
1100
{y = .6542285850, x = .9363612791}
1000
{y = .5636183410, x = .9204118951}
900
```

*Error, (in fsolve/genroot) cannot converge to a solution*

ちゃんと求まっているのでしょうか? 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-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 DATA10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（以下はrelease4 での入出力例です）

> readlib(readdata):

> with(linalg):

> readlib(FFT);

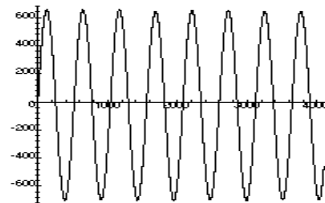
> T:=readdata('Data102',1):

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

> n:=vectdim(T);

n := 4096

> with(plots):listplot(T);



>readlib(FFT);

proc(m, x, y) ... end

> Rdata:=array(sparse,1..n):

> Idata:=array(sparse,1..n):

> for i from 1 to n do

> Rdata[i]:=T[i];

> od:

>

> FFT(12,Rdata,Idata);

4096

datapoint:=array(sparse,1..50,1..2);

> for i from 1 to 50 do

> datapoint[i,1]:=i:

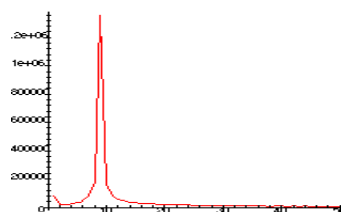
> datapoint[i,2]:=sqrt(Rdata[i]^2+Idata[i]^2):

> od:

*datapoint := array(sparse, 1 .. 50, 1 .. 2, [ ])*

datapoint1:=convert(datapoint,listlist):

> plot(datapoint1);



---

---

## 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?indexと?packageです。?packageにはMaple Vに含まれている各数学分野の関数パッケージのリストが含まれています。

### 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。

# 索引

## 記号

.(ピリオド) 26  
:(コロン) 7  
;(セミコロン) 7  
`(バッククォート) 26, 29, 50  
(シングルクォート) 10, 20, 28  
"(ダブルクォート) 10  
[] 18  
{ } 18  
# 26

## A

add 25  
adjoint 22, 61  
angle 22, 61  
args 26  
array 20  
assign 13, 46  
assume 13, 47, 51, 60

## B

break 24

## C

C 41  
Chebyshev 多項式 26  
coeff 13, 47, 60  
coldim 22, 61  
collect 13, 47, 60  
combine 13, 60  
complex 14  
conjugate 47  
convert 13, 21, 60  
convolution 33  
crossprod 22, 60

## D

debug 27  
denom 13, 46, 60  
det 22, 54, 61  
diff 15, 53  
display 38  
dotprod 22, 60  
dsolve 15, 53

## E

eigenvals 22, 61  
eigenvects 22, 61  
evalf 11  
expand 13, 49, 60

## F

factor 13, 60  
FFT 30, 59  
filter 7  
for 23  
Fortran 41  
fsolve 58

## G

gcd 13, 60

## H

Hamiltonian 54  
Heaviside 28

## I

I 10  
if 23  
infinity 10, 14, 51  
int 16  
interface 40  
intersect 18  
inverse 22, 61  
isolve 43

## L

LATEX 41  
lcm 13, 60  
limit 14, 51  
linalg 21  
list 18  
listplot 29  
logplot 32

## M

Mandelbrot 集合 3  
map 21  
matadd 22, 60  
matrix 21, 54, 55  
mint 40  
minus 18  
multiply 22, 55, 60

## N

nargs 26  
next 24  
nops 19  
normal 13, 47, 60  
numer 13, 46, 60

## O

op 19

## P

package 21  
Parseval の定理 32  
Pi 10  
piecewise 28  
plot 30  
plot3d 12  
plots 12  
print 19  
printf 26  
printlevel 27  
procedure 24  
procname 26

## R

readdata 29  
readlib 11  
recursive. 再帰的 を参照  
remember 26  
rowdim 22, 61

## S

seq 19, 32  
series 51, 54  
set 18  
simplify 13, 46, 52, 60  
solve 13, 46, 54  
sort 13, 60  
subs 10  
sum 14

## T

table 19  
Tayler 級数展開 16  
trace 22, 61  
transpose 21, 22, 61  
type 21

## U

unapply 11, 50  
union 18

## V

vectdim 22, 61  
vector 21

## W

while 23  
with 11, 21  
writedata 41  
writeto 40

## イ

因数分解 13, 60

## エ

永年方程式 54  
エラー処理 26

## オ

オイラー角 55

## カ

解 13  
改行 7  
外積 22, 60  
階段関数 28  
角度 22, 61  
関数 11  
関数の plot 7, 28  
関数の定義 11  
簡単化 13, 60

## キ

逆行列 22, 37, 61  
級数 14, 16  
行列 11, 60  
行列式 22, 61  
極限 14  
近似 51

## ク

組み合わせ 11, 60  
組み込み関数 11

## ケ

係数 13, 60  
結合 19  
減衰振動 59

## コ

高速フーリエ変換 59  
コメント 26  
固有値 22, 61  
固有ベクトル 22, 61  
コロソ 7

## サ

差 18  
再帰的 26  
最小公倍数 13, 60  
最大公約数 13, 60  
三角関数 11

## シ

式の変形 13, 44, 60  
実行の中断 8  
    中断 8  
自由エネルギー曲線 56  
集合 18  
出力の抑止 7  
初期化 10  
初期条件 15

## ス

随伴 22, 61  
数式処理 10  
複素数 10

## セ

制御構造 23  
整数関数 11, 60  
整数論 11, 60  
積 18, 22, 60  
積分 16, 51  
セミコロソ 7  
線形代数 21, 54, 55

## ソ

ソート 13, 60  
総和 14

## タ

対角和 22, 61  
代入 10  
多項式 11, 60  
畳み込み 33

## チ

調和振動子 51  
直交多項式 11, 60

## テ

データ構造 18  
データ処理 29  
データフィット 34  
定義関数の plot 27  
テイラー級数展開. Taylor級数展  
    開を参照  
手続関数 24  
デバッグ 27, 40  
展開 13, 60  
転置 22, 61

## ト

統計学 11, 60  
同類項 13, 60  
トレース 22, 61  
トンネル効果 44

## ナ

内積 22, 60

## ニ

入出力 29, 40  
入力 7

## ネ

熱膨張 51

## ハ

パーシバルの定理 32  
配列 20

## ヒ

微積分 14  
非線形最小二乗法 34  
微分 15  
微分方程式 15  
表 19

## フ

フーリエ変換 30  
フィルタ 31  
複素関数 16  
プログラミング 23  
プロット 10, 12  
分子 13, 60  
分数の結合 13, 60  
分母 13, 60

## へ

ベクトル関数 11, 60  
ヘルプ 8

## ホ

方程式の解 13  
ボルツマン分布 51

## マ

間違い修正 8

## ム

無限大 10

## ヤ

ヤコビ行列 35, 36

## ユ

有理関数 11, 60

## ヨ

要素 19  
陽電子消滅寿命 53  
呼び出し 11

## リ

リスト 18

## ル

ループ 24

## レ

連結作用素 26  
連立微分方程式 53  
連立方程式の数値解 56  
連立方程式の整数解 43

## ロ

ローレンツ型関数 34

## ワ

和 18  
和(行列) 22, 60