物理学情報処理演習

１４. 数式処理

Maple10

参考文献

はやわかりMaple
赤間世紀著　共立出版社

Maple 10 User Manual
Maplesoft社

物理学のためのMathematica
ロバート・ジンマーマン他著　ピアソン・エディケーション

大久保晋

E-mail: buturi-johoshori@tiger.kobe-u.ac.jp

http://extreme.phys.sci.kobe-u.ac.jp/staffs/okubo/lectures/Programming/index.html

Maple ってなに？

- Maple は 数式処理（計算代数）システムです。

  数式処理は、コンピュータ代数とも呼ばれる「数式を数学的に操作することを研究するコンピュータサイエンス」の１分野。

  数式処理は、代数構造をコンピュータで扱うことが特徴。

  × 数値計算は計算誤差がある。

  ○ 数式処理の結果は厳密でかつ、正確。

  数式処理は、我々が通常行なっている数学の計算に非常に近い

  数式処理は、万能ではない

  × 記号的解法の計算が膨大な計算量となることがある

  × 記号的解法が知られていない問題が存在する

  しかし、科学や工学の問題のモデル化や経済学のモデル化に利用されている（今や経済学に数学とコンピュータは不可欠！）

数式処理の歴史

1960年代　人工知能(artificial Intelligence)の研究の一部として数式処理の研究始まる。（主にMITを中心として。1963年Slagelは不定積分を求めるシステムを発見的手法にて開発）

1970年代　数式処理システム REDUCE(1966)、MACSYMA(1969)の登場。当時はLISP言語で記述されていたため高性能計算機が必要。

1980年代　数式処理研究の進歩とアプローチの変化により、現代数学との関連が深まる。解法的アプローチにより、LISP言語から汎用言語で開発が可能になった。

1980年 カナダのWaterloo大学でMapleプロジェクトが開始

1981年 S. Wolframが数式処理システムSMPの商品化

1987年 Maple商品化(UNIX workstation向け)

1988年 S. Wolfram、Mathematica 発表

※S. Wolframは20歳でCaltechで理論物理学博士号を取得した物理学者です。自らの計算に必要なため数式処理システムを開発した。

演習１４－１：Mapleを使おう

Mapleの簡単なチュートリアルは以下にある。

http://extreme.phys.sci.kobe-u.ac.jp/extreme/staffs/okubo/lectures/Programming/MapleTut.pdf

これをみながら、Mapleを使ってみよう。（「方程式を解いて検算する」くらいまで、やってみよう）

重要なポイント：

1. 数値の扱いが、実数、有理数で与えることが可能。
2. 無理数（π）の近似値を与える。
3. 数式のまま、計算（微分・積分）ができる。
4. 数値計算も可能
5. くり返し計算ができる。
6. 方程式を数式のままとくことができる。

演習１４－２：数値の扱い

数値には、実数、有理数、無理数、虚数とある。それらの違いを見てみよう。

```
> 1+1;
                    2                    整数の２がでる
> 3/2 + 5/3;
                      19
                      --
                      6
> evalf(%);
                    3.166666667
```

evalfは浮動小数点近似を求める命令。%は直前の結果を意味する。

```
> 1.23e+15;
                    .123 10^16
> sqrt(3);
                      √3
```

演習１４－２：数値の扱い

近似値と厳密値との違いは、続けて次のように入力すると分かる。

```
> evalf(%);
                    1.732050808
> %^2;
                    3.000000001
> sqrt(3);
                        √3
> %^2;
                    3
```

三角関数も正確に計算されている

```
> sin(Pi);
                    0
```

演習１４－２：数値の扱い

複素数を入力するためには、虚数記号 I を用いる。

```
> I^2;
                    -1
> 1+2*I;
> %^2;
                    -3+4I
```

ただしく演算されていることを確認しよう。

算術演算子の種類

Maple で使用できる算術演算子の種類は次の通り。a, bは任意の数。nは整数

種類	記述	意味
+	a + b	加算
-	a - b	減算
*	a * b	乗算
/	a / b	除算
iquo	iquo(a, b)	商（整数）
irem	irem(a, b)	a/bの余り（整数）
^	a ^ b	べき乗（a^b）
**	a ** b	べき乗（a^b）
!	n!	階乗
isqrt	isqrt(n)	平方根整数化
signum	signum(a)	符号
abs	abs(a)	絶対値整数化
min	min(a, b)	最小値
max	max(a, b)	最大値

組み込み関数の種類

Maple で使用できる関数は多岐に渡る。下記に初等関数の一部を記載する。

種類	意味
sqrt	平方根
exp	指数関数
ln	自然対数関数
log10	常用対数関数
log	対数関数
sin, cos, tan	三角関数
sec, csc, cot	
sinh, cosh, tanh	双曲線三角関数
sech, csch, coth	
arcsin, arccos, arctan	逆三角関数
arcsec, arccsc, arccot	

演習１４－３：関数の扱い

これら初等関数の使用例の一部を示す。

```
> sqrt(4.0);
                    2.000000000
> sqrt(-5.0);
                    2.236067978I
```

log[b]は任意の底bの対数$\log_b x$を計算する。

```
> log[2](32);
```

$$\frac{\ln(32)}{\ln(2)}$$

整数関数の基本的な種類

整数関数を中心とした基本関数の一部を記載する。

種類	意味
trunc	引数の少数部を切り捨てた整数
round	引数のもっとも近い整数
frac	引数の少数部
floor	引数を越えない最大の整数
ceil	引数をこえる最小の整数
abs	絶対値
max	最大値
min	最小値
Re	複素数の実部
Im	複素数の虚部

種類	意味
conjugate	共役複素数
a mod b	aをbで割ったときの余り
iquo(a, b)	整数aを整数bで割ったときの商
irem(a, b)	整数a を整数bで割ったときの余り
ifactor	整数の素因数分解
igcd(a, b)	最大公約数
ilcm(a, b)	最小公倍数
isprime	素数判定
ithprime	引数番目の素数
nextprime	引数より大きい最初の素数

制御文

Mapleでももちろん制御文（if, for…)があり、構造化プログラミングを行える

条件分枝　if 文の書き方には次ぎの４つがある

if 式 then 文の並び fi;

if 式 then 文の並び 1 else 文の並び 2 fi;

if 式 1 then 文の並び 1 elif 式 2 then 文の並び 2 fi;

if 式 1 then 文の並び 1 elif 式 2 then 文の並び 2 else 文の並び 3 fi;

種類	意味
=	等しい
>	より大きい
<	より小さい
>=	以上
<=	以下
<>	等しくない
and	論理積
or	論理和
not	論理否定

制御文

Mapleでのくり返し構文

くり返しには for文と while文がある。c言語より複雑なので注意が必要
for 文には一定の回数をくり返す for-from 文

for 名前 from 式1 to 式2 do 文の並び od;

for 名前 from 式1 by 式2 to 式3 while 式4 do 文の並び od;

の２種類と、リスト構造などにおける一定回数くり返す for-in 文

for 名前 in 式1 while 式2 do 文の並び od;

がある。以下がその例

```
> for n from 1 to 4 do n^2 od;
```

これと同じく

```
> for n to 4 do n^2 od;
```

これを使うと偶数の階乗n!が800をこえる最初の整数を表わしてみよう

```
> for n from 0 by 2 while n!<800 do od;
> n;
```

制御文

for-in 文の例

```
> L := [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
> j := 0:
> for i in L while i <= 10 do j := j + L[i]; od:
> j;
```

繰り返し処理は while 文を用いても行える。

```
        while 式 do 文の並び od;
```

while文は、「式」が偽になるまで「文の並び」をくり返す。たとえば、

```
> i := 1:
> j := 0:
> while i <= 10 do j := j +i; i := i + 1; od;
> j;
```

制御文

繰り返しの制御には break文と next文がある。break 文は、繰り返しを中止する命令であり、next文は繰り返しの一部を中止し次の命令を実行する命令である。

```
> i := 1:
> while i <= 10 do i := i + 1; if i = 3 then break; fi; od:
> i;
```

i = 3 でループから抜けていることがわかる。

next文の使用例は次の通りである。

```
> for n to 5 do if n = 3 then next; fi; n; od;
```

n = 3 で繰り返し処理が中断され、次の繰り返し処理が行なわれている。

演習１４－４：様々なコマンド

Mapleには沢山のコマンドがあるので全てを説明することはできないので、ここではその一部を示す。コマンドの説明等は、メニューバーの右端にあるHelpをクリックすれば詳細なhelp windowがでるので、そこから検索してコマンドの利用方法を調べて下さい。

因数分解

```
> factor(x^2-3*x+2);
                    (x-1)(x-2)
```

100の階乗

```
> 100!;
```

演習１４ー５：プロット

複雑な式もMapleでは容易に可視化できる。

２次元プロット

```
> plot(tanh(x), x=-5..5);
> plot([exp(-x^2), -exp(-x^2), sin(12 x) exp(-x^2)],x=0..3, axes=boxed,
        color=[red, green, blue]);
```

plot(関数, 変数の範囲);である。

３次元プロット

```
> eq1:=sin(x)*cos(y);
> plot3d(eq1, x=-Pi..Pi, y=-Pi..Pi);
```

plot3dも plotと同じである。ここでは、１行目で関数を定義し、２行目のplot3dで定義した関数eq1を用いている。plotもplot3dも様々なオプションがある。枠を表示させるには

```
> plot3d(eq1, x=-Pi..Pi, y=-Pi..Pi, axes=boxed);
```

演習１４－６：方程式の解

solveで方程式の解がもとまる。

```
> _EnvExplicit := true
> eqset:={x+y=1, y=1+x^2};
> solve(eqset, {x, y});
```

_EnvExplicit := trueを入れないと方程式の解を可能な限り簡単な形式で表現した RootOf … の形で解答されてしまうので、これを入れておいた

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 がある。

```
> restart;
> f1:=x -> -x*ln(exp(-1/x)/(1-exp(-1/x)));
> fsolve(f1(x)=0,x);
```

矢印 -> は、ユーザー定義の関数であることを示す。

restart; は変数に入れていた値をクリアーにします。新しいウィンドウを開かずに連続して計算している場合、前の計算結果がそのまま引き継がれるのですが、それをクリアーしたい場合に使う。

演習１４－７：微分、積分

数式の微分
> diff(x^2, x);

関数の一般系のままでも形式的な微分を表示します。
> diff(y^2*x^2, x, x);
> c:=(x, t) -> X(x)*T(t);
> diff(c(x,t), x);
> diff(c(x,t), x, t);

数式の積分
> int(ln(x), x);
> int(sin(x), x=-Pi..0);
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);

> eq2:=exp(-x^2);
> int(eq2, x=2..5);　　定積分。積分記号だけ出したい場合は Intを使います

演習１４－８：微分方程式

Mapleの数式処理機能を用いて微分方程式を未知のパラメーターを含む形で解くことが可能です。

$$M \cdot \ddot{x} + c \cdot \dot{x} + k \cdot x = 0$$

を入力してみましょう。Mapleでは式の微分を diff コマンドで記述します。

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

さて、上の式をdeqとして入力してみましょう。

演習１４－９：ベクトル、行列演算

ベクトル、行列演算も可能です。数式パレットを使って入力することも、テキスト入力も可能です。

ベクトル、行列演算する際は、初めにLinearAlgebraパッケージを読み込む

```
> with(LinearAlgebra);
```

列ベクトル

```
> v1:=Vector([x,y,z]);
```

列ベクトルを行ベクトルへ

```
> v2:=Vector[row]([x,y,z]);
```

[ ]の代わりに< >を用いて列ベクトル、行ベクトルを生成できます。

```
> v1:=<x,y,z>;
> v2:=<x|y|z>;
```

Matrixコマンドでも行列の生成ができます。以下は2行3列の行列をlistから生成

```
> A0:=Matrix(2,3,[[1,2,3], [4,5,6]]);
```

< >と | を使っても生成できます。

```
> A1:=<<1,2,3>|<4,5,6>|<7,8,9>>;
```

演習１４－１０：ベクトル、行列演算

行列の演算
> A3:=[[1, 2], [3, 4]];　　list表示からの行列化
> A4:=convert(A3, Matrix);
> A5:=Matrix(2, 2, [[3, -1], [1, 2]]);
> a*A4+b*A5;
行列の和

行列の積は . で行ないます。
> A4.A4;
. は行列とベクトル、あるいはベクトル同士の内積にも使われます

ベクトルの外積は OuterProductMatrixをつかいます。
> OuterProductMatrix(v1, v2);

逆行列
> MatrixInverse(A4);
行列式
> Determinant(A4);

演習１４－１０：ベクトル、行列演算

固有値はEigenvaectorsでもとまります。
> Eigenvectors(A4);

演習１４ー１１：２つの点電荷と電位ポテンシャル

x軸上に２つの点電荷Q(c)があるとき、電荷の周りの電位分布と電気力線分布を
　　グラフで示す。

電荷Q(c)は規格化したあたいとする。

```
> vpn:=(x, y) -> 1/(sqrt((1-x)^2+y^2))+1/(sqrt((1+x)^2+y^2));
> plot3d(vpn(x,y), x=-2..2, y=-2..2, view=0..5, axes=boxed);
```

等高線で書くならば

```
> contourplot(vpn(x,y), x=-2..2, y=-2..2);
```

電気力線分布は電位ポテンシャルのgradientだから

```
> fieldplot([diff(vpn(x,y), x), diff(vpn(x,y), y)], x=-2..2, y=-2..2);
```