自分用Mathematicaマニュアル

吉川周二 (Shuji Yoshikawa)

愛媛大学大学院理工学研究科生産環境工学専攻

Contents

Chapter 1

基本操作

(1) 実行: Shift+Enter

(2) ヘルプ: コマンドを入力した後にF1、または?コマンド名。UNIX と同様にワイルドカード「*」も使える、例えば「?Inte*」や「?*tegra*」なども使える。

(3) 割り込み: 無限ループに入ったりしてしまった場合、alt++(Ctrl+c) で割り込みによる計算途中での中断が可能。中断された場合の選択メニューとしては **continue**(計算続行)、**abort**(計算破棄)、**exit**(Mathematica の終了) がある。

(4) 長い関数にはCtrl+kで補完できる.

Chapter 2

よく使うコマンド

コマンド名とラフな使い方のみ記す. 細かい使用法はF1ヘルプを利用すべき.

2.1 計算

2.1.1 数の表示

数値 (近似値) で表示 m 桁の数値で表示	N[数] N[数, m], 数 ‘m
数値を分数で表示 与えられた精度での最も近い有理数を与える	Rationalize[数]
π, e, i, ∞	Pi, E, I, Infinity
a と b の間の整数乱数	Random[Integer, {a,b}]

Remark 2.1.1. 数の計算は厳密解, 数. にして計算をすると近似解.

```
1: a=2/3; b=2./3;
2: {a,b}  →{2/3, 0.66667}
```

ただより正確に精度も含めて近似値を求めたければ, N[2/3,100] などとすべき. ちなみにN[2./3,100] では結局, *0.66667* を返す. これはさきに *2./3* を計算して丸められたものを数値化しているからである.

2.1.2 整数の計算

素因数分解	FactorInteger[整数]
整数 a ÷ b の商	Quotient[a,b]
整数 a ÷ b の余り	Mod[a,b]

2.1.3 複素数

実部, 虚部	Re[複素数], Im[複素数]
共役複素数	Conjugate[複素数]
絶対値	Abs[複素数]
偏角	Arg[複素数]

2.1.4 多項式の計算

因数分解	Factor[式]
式の展開	Expand[式]
多項式 a(x) ÷多項式 b(x) の商	PolynomialQuotient[a,b,x]
多項式 a(x) ÷多項式 b(x) の余り	PolynomialRemainder[a,b,x]

2.1.5 分数式の計算

通分	Together[分数式+分数式]
約分	Cancel[分数式]
部分分数分解	Apart[分数式]
分母	Denominator[分数式]
分子	Numerator[分数式]
式を簡単にする (←曖昧な指示)	Simplify[分数式]

2.2 方程式

2.2.1 方程式

先にまとめる.

コマンド	Solve	Roots	Reduce	NSolve	FindRoot
解の種類	厳密解			近似解 (数値解)	
返される解の型	置き換え規則	方程式		置き換え規則	方程式
取り扱える方程式	(連立)整方程式	一変数整方程式	(連立)方程式	(連立)整方程式	(連立)方程式

厳密解 (無限精度) Solve，Roots，Reduce

- 「(多項式)=0」の方程式は Solve[{方程式 1，方程式 2，...}，{未知数 1，未知数 2，...}] でとく．解は置き換え規則{a → b}のリストで与えられる．一元一次方程式ならリストにしなくてよい．
- 「(多項式)=0」の一変数の方程式は Roots[方程式，未知数] でもとける．解は方程式の形で返す
- 「」Reduce[式，{未知数 1，未知数 2，...}] を解く．式は方程式に限らず, 連立方程式でも連立不等式でもよいがリストでなく, 論理式の形になり, 解は式の形で返す．

```
1: Reduce[x^2==1, x] →x==1 || x==-1
2: Roots[x^2==1, x] →x==1 || x==-1
3: Solve[x^2==1, x] →{{x->1}, {x->-1}}
4: Reduce[x^2==1 && x > 0, x] →x==1
5: Solve[{x+y==0, x^2+y^2==2},{x,y}] → {{x?-1,y?1},{x?1,y?-1}}
```

エラーが出る例:

```
1: Reduce[2*Sin[x]==x, x]  ←厳密解は (数値解ならわかる) 求まらない．
2: Roots[{x+y==1,x+2y==2}, {x,y}]  ←一元の方程式でない
3: Solve[2*Sin[x]==x, x] ←整方程式 (多項式=0 の形) でない
```

エラーが出なかった例:

```
4: Reduce[{x+y==1,x+2y==2}, {x,y}] ←２式は論理式&&で結んでいないのに...
```

近似解 (有限精度) NSolve，FindRoot

- Solve の有限精度の近似解もとめるコマンドが NSolve[] で使い方は Solve と同じ．5 次以上の方程式など解けない方程式に有効．ただし Solve も NSolve も多項式で表される方程式しか扱えない．
- 多項式の方程式でない一般の非線形方程式に対しては FindRoot を用いる．FindRoot[方程式，{x,a}] の形式で x についての方程式の a 近傍での近似解をみつける．FindRoot[方程式，{x,a,b,c}] として, $[b, c](\ni a)$ での解を探し, 見つからなければ計算をストップすることもできる．Solve と同様に連立にすることもでき, その場合は各変数をどのあたりでの解を探すかを指定しないとならない．FindRoot の解は置き換え規則{x → c}の形で与えられる．

```
1: NSolve[x^2 + 3 x + 1 == 0, x]  → {{x?-2.61803},{x?-0.381966}}
2: FindRoot[2*Sin[x]==x, {x,2}]  → {x?1.89549}
```

エラーが出る例:

```
1: Reduce[2*Sin[x]==x, x]  ←厳密解は (数値解ならわかる) 求まらない.
2: NSolve[2*Sin[x]==x, x]  ←整方程式でない.
```

2.2.2 微分方程式

詳しくは 3 章に. 基本的には, F1 ヘルプで使えばよい.

微分方程式の厳密解 (有限精度) DSolve

- (連立) 常微分方程式の一般解. 連立はリストにするだけ.
- (連立) 常微分方程式の初期値問題の解
- 線形偏微分方程式の一般解
- 線形偏微分方程式の初期値問題の解

微分方程式の近似解 (有限精度) NDSolve

- 常微分方程式の初期値問題
- 非線形偏微分方程式の初期値問題

2.3 線形代数の計算

行列とベクトルはリストで表されるので, 一部の操作はリストの操作になる.

2.3.1 ベクトルの計算

ベクトルの内積	a.b
ベクトルの i 成分を取り出す	a[[i]]
ベクトルの外積	Cross[a,b]

2.3.2 行列の表示

行列での表示	MatrixForm[リストでの行列]
行列の (i,j) 成分を取り出す	a[[i,j]]
行列の i 行目を取り出す	a[[i]]
n 次単位行列	IdentityMatrix[n]

2.3.3 行列 (ベクトル) の計算

行列の積	a.b
累乗	MatrixPower[a,n]
行列の指数乗	MatrixExp[a]
行列式	Det[a]
逆行列	Inverse[a]
固有値	Eigenvalues[a]
固有ベクトル	Eigenvectors[a]
固有値と固有ベクトル	Eigensystem[a]

Remark 2.3.1. 行列 a と b に対して積を計算するために a*b としてはいけない. これは各成分の積を成分とする行列 $(a_{ij}b_{ij})$ を返してしまう. 同様に a^n とすると各成分を単純に累乗した行列 $((a_{ij})^n)$ を返してしまうことに注意

2.4 微分積分の計算

極限 Limit

- 極限 Limit[式, x->x_0]
- 右極限 Limit[式, x->x_0, Direction->-1],
  左極限 Limit[式, x->x_0, Direction->1]
- 多次元の極限はなさそう.

微分 D, ', Derivative

- f_x =D[f[x],x], f_{xy} =D[f[x,y],x,y] のように偏導関数なども求められる. 一般の高階導関数は $\partial_x^n f$ =D[f[x],{x,n}] と表す.
- 一変数関数については f'[x] でも表現可能. これは [f[x],x] と同値. その導関数に値を代入するのも $f'(3)$ =f'[3] のように簡単に表現できる. D を使うと, D[f[x],x] /. x→3 と表現しなくてはならない. 二階導関数も f''[x] と表現できる.
- $\partial_x^{n_1}\partial_y^{n_2} f(x,y)$ =Derivative[n_1,n_2][f][x,y] のような表現もある.
- 全微分は df =Dt[f[x,y]] のように表される. この場合, dx =Dt[x], dy =Dt[y] のように表される.

積分 Integrate, NIntegrate

- 不定積分, 定積分 (無限精度).
  $\int f(x)dx$ =Integrate[f[x], x]
  $\int_a^b f(x)dx$ =Integrate[f[x],{x,a,b}]
  で表される.
- 数値定積分 (有限精度) NIntegrate[f[x],{x,a,b}] 重積分も同様.
- 重積分 (定積分) 積分の順番は外側から行われることだけ注意.
  $\int_a^b \int_c^d f(x,y)dydx$ =Integrate[f[x,y],{x,a,b},{y,c,d}]
  $\int_a^b \int_{c(x)}^{d(x)} f(x,y)dydx$ =Integrate[f[x,y],{x,a,b},{y,c[x],d[x]}]

テイラー展開 Series

- 一変数関数 $f(x)$ の x_0 中心に x 変数について, n 次までのべき級数展開をするのは Series[f[x], {x,x_0,n}].
- 多変数関数のときは, Series[f[x,y], {x,x_0,n},{y,y_0,m}] のようにすればよいが, これは x に関するべき級数展開を行い, 次に y に関するという具合に継続して展開するだけなのでそれほど使い勝手はよくない.

2.5 グラフの表示

2.5.1 二次元曲線

関数のグラフ Plot

- Plot[f[x], {x,a,b}] で f(x) の a〜b までのグラフが表示される. デフォルトで x 軸と y 軸の目盛りの比は黄金比 (GoldenRatio) になっている.
- Plot[f[x], {x,a,b}, AspectRatio->Automatic] とすると縦横の比は同じになる.
- PlotRange -> {c,d}をオプションでつけると, 値域が $[c,d]$ で表示してくれる.
- Plot[{f[x],g[x]}, {x,a,b}] のように関数をリストにすると複数のグラフを表示することもできる.

パラメータ表示された曲線のグラフ ParametricPlot

- ParametricPlot[{f[t],g[t]}, {t,a,b}] でパラメータ表示された曲線の a〜b までのグラフが表示される．デフォルトで x 軸と y 軸の目盛りの比は黄金比 (GoldenRatio) になっているので, 先と同様に AspectRatio->Automatic とすると縦横の比は同じにすればよい．Automatic のところは数字を入れて比率を変えることも可能.

2.5.2 三次元曲面

二変数関数のグラフ Plot3D

- Plot[f[x,y], {x,a,b},{y,c,d}] で f(x,y) の $[a,b] \times [c,d]$ のグラフが表示される.
- 等高線が書きたいときは ContourPlot[f[x,y],{x,a,b},{y,c,d}] とかく．よりなめらかな等高線が書きたければ, ContourSmoothing -> Automatic というオプションをつければよい.

パラメータ表示された曲面のグラフ ParametricPlot3D

- ParametricPlot3D[{f[t,s],g[t,s],h[t,s]}, {t,a,b}, {s,c,d}] でパラメータ表示された曲面のグラフが表示される.

Chapter 3

微分方程式関連

常微分方程式の一般解
ODE の初期値問題の解
ODE の数値解
PDE の数値解をグラフに
アニメに

-

```
1:
2:
3:
4:
5:
6:
7:
8:
9:
10:
```

```
a
```

3.0.3　分数式の計算

Together[**分数式＋分数式**]	分数式＋分数式を通分してたす
Simplify[**分数式**]	分数式を簡単にする
Cancel[**式**]	式を約分する.
Cancel[**(1-x)/(1-**x^2**)**]	$\frac{1}{1+x}$

Chapter 4

プログラム

主に [1] に従った.

4.1 ループ, 繰り返し処理

以下のプログラムは, すべて計算の式が複数ある場合は, 「式 1; 式 2; ··· 」とセミコロンでつなげばよい.

4.1.1 Do

Do[式, {i,imin,imax,di}]

- i を imin から imax まで刻み幅 di で式を繰り返す.
- デフォルトで di=1, imin=1, i=1 なので, **Do[式, {imax}]** としたら, imax 回式を繰り返すことになるし, ほかにも {i, imax } や { i, imin, imax} などを用いることもできる.
- 式のところには+=, -=, *=, /=, などプログラムでよく用いられる表現も利用可能.

```
1: n=100; s1=0; s2=0;
2: Do[s1=s1+i; s2=s2+, \{ i,1,n,1 \} ]}
3: s1  →5050
4: s2  →388350
```

4.1.2 While

While[真偽文, 式]

- 真偽文が正しい間, 式の計算を続ける.
- 真偽文には, x==y, x!=y, x¿y, x¿=y のほか, !p(否定), pq(論理積), p——q(論理和), Xor[p,q,...](排他的論理和), True, False などが入る.

```
1: n=100; s1=0; s2=0; i=1;
2: While[i<=n, s1+=i; s2+=i^2; i++]
3: s1 →5050
4: s2 →388350
```

4.1.3 For

For[初期設定, 真偽文, 増加式, 式]

- 初期設定から始まり, 真偽文が真なら, 増加式, 式の順に実行する.

```
1: n=100;
2: For[s1=1; s2=0; i=1, i<=n, i++, s1+=i; s2+=i^2]
3: s1 →5050
4: s2 →388350
```

Remark 4.1.1. 基本的にこれらの手続き型プログラムよりは, 関数型プログラムを使った方が時間が早くなることが, *[?]* にも記載されているが, 極端な例は *[?, P264]* が参考になる.

4.2 分岐

4.2.1 If

If[真偽文, 式 1(真), 式 2(偽), 式 3(どちらでもない)]

- 真偽文が, 真なら式 1 が, 偽なら式 2 が, どちらでもないときは式 3 が実行される. 式 3 は C 言語などにはない特徴ともいえる.

```
1: Remove["Global‘*"];
2: x=2; y1=If[x==2,0,1,-1];
```

```
3: x=3; y2=If[x==2,0,1,-1];
4: x=a; y3=If[x==2,0,1,-1];
5: {y1,y2,y3}  →{0,1,-1}
```

4.2.2 Which

Which[真偽文 1, 式 1, 真偽文 2, 式 2, 真偽文 3, 式 3, . . .]

- 真偽文 1 が, 真なら式 1 が, 偽なら真偽文 2 をチェックし真なら式 2 が, . . . , 真であるものがなければ何も返さない.

```
1: x=2;
2: Which[x==0, a=0, x==1, a=10, x==2, a=20, x==3, a=30]
3: a →20
```

4.2.3 Switch

Switch[計算式, 値 1, 式 1, 値 2, 式 2, 値 3, 式 3, . . .]

- 計算式の結果が, 値 1 に等しければ式 1 が, 偽なら値 2 と比較し等しければ式 2 が, . . . , 真であるものがなければ何も返さない. すなわち値の比較をするのが Switch.

```
1: x=2;
2: Switch[x, 0, a=0, 1, a=10, 2, a=20, 3, a=30]
3: a →20
```

Remark 4.2.1. 分岐文には *Break[]* や *Continue[]* を組み合わせることで繰り返しループ *(While* や *For)* から抜けたり続けたり出来る.

```
1: x=0; n=1
2: While[True, x=n^2; n++; If[x>20, Break[]]];
3: {x,n} →{25,6}
```

```
1: s=0;
2: Do[If[Mod[i,2]==0, Continue[]]; s+=i, {i,1,10}];
3: s  → 25 (1+3+5+7+9 を計算している)
```

4.3 関数の定義

4.3.1 遅延割り当てと即時割り当て

:=

- **遅延割り当て**. `f[x_]:= a*x^2` などが基本.
- 引数にデフォルトを定める時は, `f[x_:10, y_:100]` のように**コロン**を用いる.]
- 関数は複合表現でもよくその場合は (式 1; 式 2;\ldots) のように括弧でくくる. この表現を用いることでより複雑な処理をする関数を定義出来る.
- 引数は数でなく, 関数などのオブジェクトでもよい.
- 場合分けには, `If` を使うだけでなく, `/;` を使うのも便利.
- `[f[x_Integer]]` のようにして変数の型を指定することもできる.

```
1: Remove["Global‘*"];
2: f[n_:10] :=(
3:    m=n^2;
4:    l=Sin[m];
5:    {m,l} )
6: f[]   →{100, Sin[100]}
7: f[3]  →{9, Sin[9]}
```

```
1: Remove["Global‘*"];
2: ingr[g_]:= Integrate[g[x], x];
3: ingr[Sin]  → -Cos[x]
```

次の関数

$$f(x) = \begin{cases} -1, & x < 0, \\ 1, & x \geq 0, \end{cases}$$

はもちろん `If` を用いても定義出来るが, `/;` を用いると次の用に定義できる.

```
1: f[x_]:=-1 /; x<0
2: f[x_]:=1  /; x>=0
```

=

- **即時割り当て**.

Remark 4.3.1. 遅延割り当てより即時割り当ての方が速いことがある.

```
1: n=1000;
2: f[x_]:=N[Sin[2*Pi*x/n]];
3: Timing[Table[f[x],{x,0,n-1}];]  →{0.047, Null}
```

に対して

```
1: n=1000;
2: fx=N[Sin[2*Pi*x/n]];
3: Timing[Table[fx,{x,0,n-1}];]  →{2.0331 × 10^{-15}, Null}
```

と天文学的に速さが違う. これは, 前者ではいちいち `Sin[2*Pi*x/n]` の計算をしているが, 後者は関数の定義の段階で `Sin[0.00628319*x]` となってから計算をしているからである.

4.3.2 純関数

一度しか利用しない関数はいちいち名前をつけないで定義することが出来る. このような関数を**純関数**という.

Function[変数, 関数]

- たとえば `f[x_]:=Sin[x]+x^2; f[t]` とかく場合, この関数を二度と使わない時はいちいち名前をつけるのは無駄. このようなとき `Function[x,Sin[x]+x^2][t]` と書くと便利. この場合 Function[] の全体が関数になることに注意. あえて名前をつける必要がないときにつかう.
- 多変数の時は `Function[{x1,x2,...}, 関数]` とする.
- 複合表現も勿論 OK. `Function[変数, ○; ○; ○]` .

関数 & (引数には#)

- Function が長ったらしく感じるときはこの表現は便利. `(Sin[#]+#^2)&[t]` とすればよい.
- 多変数の場合は引数を`#1, #2,`. 具体的には, `(#1^2 + #2^2)&[x,y]` とすると $x^2 + y^2$ が返される.
- 複合表現も勿論 OK. `(○; ○; ○)&[数 or 変数]`

Chapter 5

テクニック

5.1 略記号

主に [1] に従った.

5.1.1 よく使うもの

(* *)	コメント
\hline \verb;@	複合表現
○=.	変数○の値を消去. 特定の変数の値を消去している. Remove["Global`*"] はすべての変数そのものを消去する.

5.1.2 代入操作

//	関数に適用する.「t//f」でf[t] をかえす. f//t でないことに注意.
@@	リスト t の各成分を多変数関数 f の引数に使いたいとき 「f@@t」で. Apply[f,t] と同値
/@	一変数関数 f にリストの各成分を作用させてリストを返したいとき 「f/@t」で. Map[f,t] と同値

```
1: t={a,b,c};
2: t//f   →f[{a,b,c}]
3: f@@t   →f[a,b,c]
4: f/@t   →{f[a],f[b],f[c]}
```

数値計算結果の値を具体的にちょっとみたいときなどは次のように使うと便利.

```
1: Pi     →π
2: %//N   →3.14159 (N[%] と書くのと同値)
```

5.1.3 置き換え規則に対する操作

/.	置き換え規則{a → b}に対して用いる. 「a の関数/.{a → b}」でf[b] を返す. ReplaceAll[a の関数, a → b] の短縮表現.
t[[i]]	リスト t の i 番目の要素を取り出す.

```
1: sol=Solve[x^2==1,x]   →{{x->1}, {x=>-1}}
2: sol1=sol[[1]]         →{x->1}
3: x1=x/.sol1            → 1
```

5.2 その他のテクニック

主に [1] に従った.

5.2.1 キーボードからの入力

Input, InputString

- Input["入力せよ:"] などと打ち込むと,「入力せよ」の文章つきのウィンドウが立ち上がり入力待ちに. 入力した数字がその値になる. 計算をする必要がなく, 文字列の入力の時は, InputString を用いる.

```
1: n=Input["入力して下さい"];
2: n!  →(10 を入力すると)3628800
```

Chapter 6

ファイルの読み書き

主に [1] に従った.

6.1 入力・書き込み

OpenWrite, WriteString, Close

- OpenWrite[”パスも含めたファイル名”] でファイルを開いて書き込み可能な状況にする.
- WriteString[オープンしたストリーム, 文字列 1, 文字列 2, . . .] で文字列 1 文字列 2.
- Close[”パス”] でファイルをクローズする.
- 改行したければ, 文字列 i に"\n"を入れればよい.
- タブを挿入したければ同様に"\t"を使う.

```
1: t={1,2,3,4}
2: stmp = OpenWrite["C:\\a.txt"];
3: Do[WriteString[stmp, t[[i]], " "], {i,1,4}]
4: Close[stmp];
```

とすると, ファイル a.txt に

1␣2␣3␣4␣

と書き込まれる.

OpenAppend

- OpenWrite の代わりに OpenAppend を用いると, ファイルを開いて書き込み可能な状況にするが, その書き込みは 追加書き込み の形になる.

```
1: t={5,6,7,8}
2: stmp = OpenAppend["C:\\a.txt"];
3: Do[WriteString[stmp, t[[i]], " "], {i,1,4}]
4: Close[stmp];
```

とすると, ファイル a.txt は

1␣2␣3␣4␣5␣6␣7␣8␣

と追加して書き込まれる.

6.2 出力・読み込み

ReadList["場所", データの型]

- 場所でパスも含めたファイル名を指定する.
- データの型は「Number」なら数のリスト,「String」なら行ごとのリストを返す.

```
1: t = ReadList["C\a.txt", Number]  →{1,2,3,4,5,6,7,8}
```

Chapter 7

コマンド集

[2] 参照。必要があればF1ヘルプで調べて使えばよい. 説明はより詳しくできるならよりくわしくするべき.

7.1 数について

7.1.1 数の頭部

整数	Integer
有理数	Rational
実数	Real
複素数	Complex

7.1.2 定数

円周率	Pi
ネピア数, 自然対数の底	E
度	Degree
黄金比	GoldenRatio
虚数単位	I
無限大	Infinity
複素無限大	ComplexInfinity
不定	Indeterminate

7.1.3 精度

小数点以下の精度を調べる	Accuracy
精度を調べる. a の有効桁数を返す.	Precision[a]
小数点以下の精度を指定する	SetAccuracy
精度を指定する	SetPrecision

7.2 数学関数

7.2.1 基本演算

加算	Plus, +
減算	Minus, -
乗算	Times, *
除算	Divide, /
べき乗	Power, ^

7.2.2 数値関数

絶対値. 複素数にも使用可能	Abs
符号	Sign
四捨五入	Round
下回らない最小の整数	Floor
超えない最大の整数	Ceiling
小さい数をゼロにする	Chop
最大値	Max
最小値	Min
区間	Interval
区間の和集合	IntervalUnion
区間の共通集合	IntervalIntersection
複素数の実部	Re
複素数の虚部	Im
共役複素数	Conjugate
複素数の偏角	Arg
整数の除算の剰余	Mod
整数の除算の商	Quotient
整数の桁表現	IntegerDigits
実数の桁表現	RealDigits
有理数に変換する	Rationalize

7.2.3 乱数

乱数	Random
乱数の種を設定する	SeedRandom
疑似乱数発生器の状態を調べる	RandomState

7.2.4 初等関数

対数関数	Log
指数関数	Exp
べき関数	Power,^
平方根関数	Sqrt
正弦	Sin
余弦	Cos
正接	Tan
余割	Csc
正割	Sec
余接	Cot
	ArcSin
	ArcCos
	ArcTan
逆余割	ArcCsc
逆正割	ArcSec
逆余接	ArcCot
	Sinh
	Cosh
	Tanh
双曲線余割	Csch
双曲線正割	Sech
	Coth
	ArcSinh
	ArcCosh
	ArcTanh
逆双曲線余割	ArcCsch
	ArcSech
	ArcCoth

7.2.5　階乗に関する関数

階乗	Factorial, !
二重階乗	Factorial2, !!
二項係数 ${}_aC_b$	Binomial[a,b]
多項係数	Multinomial
ガンマ関数	Gamma
ベータ関数	Beta
対数ガンマ関数	LogGamma
二重ガンマ関数	PolyGamma

7.2.6　数論に関する関数

素因数分解	FactorInteger
素数	Prime
素数	PrimePi
素数の判定	PrimeQ
整数のモジュロ剰余	Mod
整数の商	Quotient
数 a〜c の最大公約数	GCD[a,b,...,c]
数 a〜c の最小公倍数	LCM[a,b,...,c]

7.3　数学計算

7.3.1　数値化

数値化する	N
桁数を指定して数値化する	N[expr,n]

7.3.2 式の操作

式を簡単にする	Simplify
式を展開する	Expand
論理式を展開する	LogicalExpand
有理式を展開する	ExpandAll
部分分数に展開	Apart
分子のみ展開する	ExpandNumerator
分母のみ展開する	ExpandDonominator
約分する	Cancel
分子を取り出す	Numerator
分母を取り出す	Denominator
べき乗に関して展開する. x^2 y^2	PowerExpand[(xy)^2]
複素変数に関して展開する	ComplexExpand
多項式を因数分解する.	Factor
式の部分を取り出す	Part, [[]]
因子を取り出す	FactorTerms
多項式の係数を取り出す	Coefficient
べきを取り出す	Exponent
通分する	Together
べきに関して整理する	Collect
三角関数に関するオプション	Trig

7.3.3 多項式

多項式の変数を調べる	Variables
多項式の係数を調べる	CoefficientList
多項式のモジュロ剰余	PolynomialMod
多項式の商	PolynomialQuotient
多項式の剰余	PolynomialRemainder
最大公約多項式	PolynomialGCD
最小公倍多項式	PolynomialLCM
多項式の割り算	PolynomialDivision
多項式の終結式	Resultant
整数多項式を因数分解する	Factor
多項式の二乗因数	FactorSquareFree
多項式の因数をリストで返す	FactorList
分周等分多項式	Cyclotomic
多項式の分解. $f(x) = g(h(x))$ となる g, h を見つける	Decompose[f[x]]

7.3.4 行列計算

行列, 配列を作る	Array
単位行列	IdentityMatrix
対角行列	DiagonalMatrix
積	Dot, .
外積	Outer
逆行列	Inverse
一般化逆行列	PseudoInverse
転置行列	Transpose
行列式	Det
固有値	Eigenvalues
固有ベクトル	Eigenvectors
固有値と固有ベクトル	Eigensystem
特性多項式	CharacteristicPolynomial
線形方程式を解く	LinearSolve
ゼロ空間	NullSpace
行既約行列	RowReduce
一般化余因子	Minors
行列のべき乗	MatrixPower
行列の指数関数	MatrixExp
特異値分解	SingularValues
QR 分解	QRDecomposition
Schur 分解	SchurDecomposition
LU 分解	LUDcomposition
LU 分解後の解	LUBackSubstitution
正規形式	HermiteNormalForm
Jordan 分解	JordanDecomposition
整数の既約基	LatticeReduce

7.3.5 方程式

方程式を解く	Solve
方程式を数値的に解く	NSolve
代数方程式を数値的に解く	NRoots
超越方程式を数値的に解く	FindRood
微分方程式を解く	DSolve
微分方程式を数値的に解く	NDSolve
変数を消去する	Eliminate
指定された変数について解く	SolveAlways
解いた方程式を簡単にする	Reduce
逆関数	InverseFunction
方程式を等しいとおく	Equal, ==
変数の置き換え規則を与える	AlgebraicRules

7.3.6 微分積分

級数を求める	Sum
級数を数値的に求める	NSum
乗積を求める	Product
乗積を数値的に求める	NProduct
極限	Limit
導関数	Derivative, ’
微分, 偏微分	D
全微分	Dt
積分	Integrate
数値的に積分する	NIntegrate
留数	Residue
べき級数展開	Series
逆級数	InverseSeries
級数の合成	ComposeSeries
高次の項を捨てる	Normal
級数の係数を取り出す	SeriesCoefficient

7.3.7 最適化

線形計画	LinearProgramming
最小点を求める	FindMinimum
制約付き最小値問題をとく	ConstrainedMin
制約付き最大値問題をとく	ConstrainedMax

7.3.8 データ処理

補間	Interpolation
多項式による補間	IntrpolatingPolynomial
関数による補間	InterpolatingFunction
関数の当てはめ	Fit
フーリエ変換を求める	Fourier
逆フーリエ変換を求める	InverseFourier
ラプラス	

7.3.9 オプション

コンパイルする	Compiled
小数点以下の精度を指定する	AccuracyGoal
全桁の精度を指定する	PrecisionGoal
許容誤差を指定する	Tolerance
内部精度を指定する	WorkingPrecision
最大繰り返し数を指定する	MaxSteps
繰り返しにおける最大ステップを指定する	MaxStepSize
アルゴリズムを指定する	Method

7.4 リスト

7.4.1 リストを作る

リスト	{e1,e2,...}
リストを作る. {f[1],f[2],...,f[k]}	Table[f[n],{n,1,k}]
リストを作る. a〜b までの整数を x とばしで羅列. a と x のディフォルトは 1	Range[a,b,x]

7.4.2 成分を取り出す

頭部を求める	Head
指定された成分を取り出す	Part, [[]]
第一成分	First
最後の成分	Last
第二成分以降	Rest
部分を取り出す	Take
リストから要素を選ぶ	Select
照合して取り出す	Cases

7.4.3 リストを調べる

位置	Position
成分数	Count
長さ	Length
次元	Dimensions
深さ	Depth
ベクトルかどうかの判定	VectorQ
行列かどうかの判定	MatrixQ
指定した成分をもつかどうかの判定	MemberQ
指定した成分をもたないかどうかの判定	FreeQ
同じ長さの部分をもつ部分の深さ	TensorRank

7.4.4 リストを加工する

先頭に成分を付け加える	Prepend
先頭に成分を付け加えて割り当てる	PrependTo
最後に成分を付け加える	Append
最後に成分を付け加えて割り当てる	AppendTo
途中に成分を付け加える	Insert
成分を取り除く	Delete
照合した成分を取り除く	DeleteCases
部分を取り除く	Drop
成分を置き換える	ReplacePart

7.4.5 リストを結びつける

連結する	Join
和集合をとる. $a \cup b$	Union[a,b]
共通部分をとる. $a \cap b$	Intersection[a,b]
補集合をとる	Complement

7.4.6 リストの中身を並べ替える

大きさの順に並べ替える	Sort
逆順に並べ替える	Reverse
左に回転する	RotateLeft
右に回転する	RotateRight
行と列を入れ替える (行列の転置と同じ)	Transpose
平らにする	Flatten
平らにする	FlattenAt
中身をまとめる	Partition

7.4.7　構造を操る

一般化された外積	Outer
一般化された内積	Inner
成分のすべての順列	Permutations
成分の順序を調べる	OrderedQ
順序の奇遇を判定する	Signature
指定されたレベル以下の部分表現のリスト	Level
表現を縫い込む	Thread

7.5　グラフィックス

7.5.1　描画

曲線を描く	Plot
データから曲線を描く	ListPlot
パラメータ表示された曲線を描く	ParametricPlot
曲面の描く	Plot3D
データから曲面を描く	ListPlot3D
パラメータ表示された曲線や曲面を描く	ParametricPlot3D
等高線を描く	ContourPlot
データから等高線を描く	ListContourPlot
密度を描く	DensityPlot
データから密度を描く	ListDensityPlot
グラフィックス要素を表示する	Show

7.5.2 基本オプション

描画範囲の指定	PlotRange
縦横比	AspectRatio
ラベル	PlotLabel
描画のスタイル	PlotStyle
描画に用いる点の数	PlotPoints
座標軸	Axes
座標軸のラベル	AxesLabel
目盛り	Ticks
座標軸の交点	AxesOrigin
座標軸のスタイル	AxesStyle
枠	Frame
枠のラベル	FrameLabel
枠の目盛り	FrameTicks
枠のスタイル	FrameStyle
格子線	GridLines
点と線をつなぐ	PlotJoined
等高線を指定	Contours
等高線の表示を指定	ContourLines
等高線の陰影を指定	ContourShading

7.5.3 三次元描画のオプション

視点を定める	ViewPoint
描画の中心となる点を定める	ViewCenter
描画の垂直方向を指定する	ViewVertical
枠の箱を描くかどうかの指定	Boxed
箱の縦横高さの比を指定する	BoxRatios
箱のスタイルを指定する	BoxStyle
箱の麺に描く格子線を指定する	FaceGrids
座標軸をおく辺を指定する	AxesEdge
疑似照明を使うか否かを指定する	Lighting
光源の位置と色を指定する	LightSources
背景照明のレベルを指定する	AmbientLight
メッシュのスタイルを定める	MeshStyle
メッシュを描く範囲を定める	MeshRange
切断面の描き方の指定	ClipFill
陰影をつけるかいなかを指定する	Shading
隠れた表面を表示するか否かを指定する	HiddenSurface
箱の外接球が収まるように描くかどうかを指定	SphericalRegion

7.5.4 高度なオプション

オプションを得る	Options
ディフォルトのオプションを定める	SetOptions
完全なオプションを得る	FullOptions
グラフィックスオブジェクトの詳細を得る	FullGraphics
背景の色を指定	Background
ディフォルトの色を使うか否かを指定	DefaultColor
表示領域を定める	PlotRegion
縦軸のラベルの向きを指定	RotateLabel
ディフォルトの文字を使うか否かを指定	DefaultFont
色を定める関数を指定	ColorFunction
描画を表示する関数を指定	DisplayFunction
色出力を指定	ColorOutput

7.5.5 グラフィックスの要素

2次元グラフィックスオブジェクトを作る	Graphics
3次元グラフィックスオブジェクトを作る	Graphics3D
曲面グラフィックスオブジェクトを作る	SurfaceGraphics
等高線グラフィックスオブジェクトを作る	ContourGraphics
密度グラフィックスオブジェクトを作る	DensityGraphics
グラフィックス要素の配列を作る	GraphicsArray
点を描く	Point
線を描く	Line
長方形を描く	Rectangle
立方体を描く	Cuboid
多角形を描く	Polygon
円・楕円・弧を描く	Circle
塗りつぶされた円・楕円・弧を描く	Disk
数値の配列を描く	Raster
グラフィックス指示の配列を描く	RasterArray
文字列を描く	Text
文字の種類を指定	FontForm
オブジェクトの位置を指定	Scaled
グレイレベルを規定	GrayLevel
色を規定	RGBColor
色を規定	Hue
色を規定	CMYKColor
線の太さを指定	Thickness
線の太さを指定	AbsoluteThickeness
点の大きさを指定	PointSize
点の大きさを指定	AbsolutePointSize
点線・破線などを指定	Dashing
点線・破線などを指定	AbsoluteDashing
表面の色を指定	SurfaceColor
多角形の前面・後面を規定	FaceForm
多角形の辺を規定	EdgeForm

7.6　プログラミング

7.6.1　代入

変数に値を割り当てる	Set, =
変数に値を遅延的に割り当てる	SetDelayed, :=
変数の値を消去する	Unset, =.
変数を消去する	Clear
記号を消去する	Remove
値を１増加し、増加前の値を返す	Increment, x++
値を１増加し、増加後の値を返す	PreIncrement, ++x
値を１減少し、減少前の値を返す	Decrement, x--
値を１減少し、減少後の値を返す	PreDecrement, --x
値を加えて割り当てる	AddTo, +=
値を引いて割り当てる	SubtractForm, -=
値をかけて割り当てる	TimesBy, *=
値を割って割り当てる	DivideBy, /=

7.6.2 チェック

等しいかどうかを判定	Equal, ==
等しくないかどうかを判定	Unequal, !=
形が同じかどうかを判定	SameQ, ===
形が異なるかどうかを判定	UnsameQ, =!=
小さいかどうかを判定	Less, <
大きいかどうかを判定	Greater, >
	LessEqual, <=
	GreaterEqual, >=
数かどうかを判定	NumberQ
整数かどうかを判定	IntegerQ
偶数かどうかを判定	EvenQ
奇数かどうかを判定	OddQ
素数かどうかを判定	PrimeQ
正かどうかを判定	Positive
負かどうかを判定	Negative
非負かどうかを判定	Nonnegative
ベクトルかどうかを判定	VectorQ
行列かどうかを判定	MatrixQ
多項式かどうかを判定	PolynomialQ
指定された形かどうかを判定	MemberQ
指定された形でないかどうかを判定	FreeQ
指定された形が表現に適合するかどうかを判定	MatchQ
真かどうかを判定する	TrueQ
原子要素かどうかを判定	AtomQ
値が与えられているかどうかを判定	ValueQ

7.6.3 論理演算

真 (定数)	True
偽 (定数)	False
否定	Not, !
論理積	And, &&
論理和	Or, ||
排他的論理和	Xor
含意	Implies
論理式を展開する	LogicalExpand

7.6.4 流れの制御

分岐	If
分岐	Which
分岐	Switch
繰り返し	Do
繰り返し	While
繰り返し	For
繰り返しの中断	Break
繰り返しの継続	Continue
複合表現	CompoundExpression, ;
一次計算停止する	Pause

7.6.5 関数プログラミング

純関数	Function, &
成分に関数を施す	Map, /@
すべての成分に関数を施す	MapAll, //@
指定された位置にある成分に関数を施す	MapAt
インデックス付きで成分に関数を施す	MapIndexed
成分を縫い込んで関数を施す	MapThread
頭部を置き換える	Apply, @@
何回も関数を施す	Nest
履歴を残しながら何回も関数を施す	NestList
不変になるまで関数を施す	FixedPoint
履歴を残しながら不変になるまで関数を施す	FixedPointList
パラメータを持つ関数を何回も施す	Fold
パラメータを持つ関数を履歴を残しながら施す	FoldList
関数たちを施す	ComposedList
条件を満たす成分を取り出す	Select
関数を合成する	Composition
関数サブルーチンを定義する	Module

7.6.6 ルールの適用

ルールを定める	Rule, ->
遅延的ルールを定める	RuleDelayed, :>
ルールを適用する	Replace
すべての部分にルールを適用する	ReplaceAll, /.

7.6.7 文字列の計算

文字列の頭部	String
文字列をつなぐ	StringJoin, <>
文字列の長さ	StringLength
文字列のバイト数	StringByteCount
文字列を逆順に並べる	StringReverse
文字列の部分を取り出す	StringTake
文字列の部分を取り除く	StringDrop
文字列を挿入する	StringInsert
文字列の部分の位置を求める	StringPosition
文字列の部分を億替える	StringReplace
文字列をストリームに変換する	StringToStream
文字列を文字のリストに変換する	Characters
文字を ASCII コードに変換する	ToCharacterCode
ASCII コードを文字に変換する	FromCharacterCode
すうじかどうかを判定	DigitQ
文字かどうかを判定	LetterQ
上段の文字かどうかを判定	UpperCaseQ
下段の文字かどうかを判定	LowereCaseQ
上段の文字に変換する	ToUpperCase
下段の文字に変換する	ToLowerCase
表現を文字列に変換する	ToString
文字列を表現に変換する	ToExpression
文字列を評価しない表現に変換する	ToHeldExpression

7.6.8 属性

属性を調べる	Attributes
属性を設定する	SetAttributes
属性を消去する	ClearAttributes
リストの成分に適用できるという属性	Listable

7.7 入出力

7.7.1 入出力

計算中に表示する	Print
キーボードから表現を読み込む	Input
キーボードから文字列を読み込む	InputString

7.7.2 ファイルの入力

ファイルを読み込みモードでオープン	OpenRead
ファイルを閉じる	Close
ファイルからストリームを読み込む	Read
ファイルをリストの形で読み込む	ReadList
パッケージを読み込む	Needs

7.7.3 ファイルの出力

ファイルを書き込みモードでオープン	OpenWrite
ファイルを追加書き込みモードでオープン	OpenAppend
ファイルを閉じる	Close
ファイルへストリームを書き込む	Write
ファイルへ文字列として書き込む	WriteString

7.7.4 出力形式

入力形式に変換	InputForm
出力形式に変換	OutputForm
C 形式に変換	CForm
Fortran 形式に変換	FortranForm
TeX 形式に変換	TeXForm
詳細な形式に変換	FullForm
木状の形式に変換	TreeForm
行列形式に変換	MatrixForm
表形式に変換	TableForm
表形式の向きを指定	TableDirections
短い形式に変換	Short

7.7.5 数の出力形式

指定された桁数に変換	NumberForm
科学表記に変換	ScientificForm
工学表記に変換	EngineeringForm
経理表記に変換	AccountingForm
n 進表記に変換	BaseForm
桁をそろえた表記に変換	PaddedForm
列方向にそろえた表記に変換	ColumnForm

7.7.6 出力形式の要素

文字列に表現を混ぜた表記に変換	StringForm
１行に並ぶような表記に変換	SequenceForm
表現を評価しない表記に変換	HoldForm

7.7.7 入力に関するオプション

各オブジェクトを独立したリストにする	RecordLists
レコードの区切りを指定する	RecordSeparators
ワードの区切りを指定	WordSeparators
ASCII コードとして読み込み	Byte
文字として読み込む	Character
表現として読み込む	Expression
数として読み込む	Number
実数として読み込む	Real
レコード単位の文字列として読み込む	Record
改行で区切られた文字列として読み込む	String
ワードで区切られた文字列として読み込む	Word

7.8 システム関連

7.8.1 ファイルシステム

ディレクトリを調べる	Directory
ディレクトリを設定する	SetDirectory
ディレクトリを元に戻す	ResetDirectory
親ディレクトリを求める	ParentDirectory
ホームディレクトリを求める	HomeDirectory
ファイル名を求める	FileNames
ファイルを複製する	CopyFile
ファイル名を変更する	RenameFile
ファイルを消去する	DeleteFile
ファイルの大きさを調べる	FileByteCount
ファイルの日付を調べる	FileDate
ファイルの日付を設定する	SetFileDate
ディレクトリの作成	CreateDirectory
ディレクトリの消去	DeleteDirectory
ディレクトリの名前を変更する	renameDirectory
ディレクトリをコピーする	CopyDirectory
パス名 (大域変数)	$Path

7.8.2 時刻と日付

表現を計算するに要した時間を求める	Timing
計算に要した CPU 時間を調べる	TimeUsed
現在の日付と時間を調べる	Date
1990 年 1 月 1 日から現在までの秒数	AbsoluteTime
日付を 1990 年 1 月 1 日から現在までの秒数に	FromDate
1990 年 1 月 1 日から現在までの秒数を日付に	ToDate

7.8.3 メモリー

使用されているメモリを調べる	MemoryInUse
使用されたメモリの最大値を調べる	MaxMemoryUsed
指定されたメモリ内で実行する	MemoryConstrained
実現を記憶するのに必要なメモリ	ByteCount

References

[1] 上坂吉則, Mathematica 数値数式プログラミング, 牧野書店, 1997.

[2] 白石修二, 例題で学ぶ Mathematica 数学編, 森北出版, 1995.

[3] 日本 Mathematica ユーザー会 編, 入門 Mathematica, 東京電機大学出版局, 2009.