数値計算：数式処理システム Reduce の使い方

インストール

数式処理システム REDUCE は SourceForge からダウンロードできる．
`http://sourceforge.net/projects/reduce-algebra/files/`

Windows 用は（reduce-windows32-20110414.zip （32 ビット版）と reduce-windows64-20110414.zip（64 ビット版））が用意されているので，これ，をダウンロードして展開すれば，インストールは終了する．展開したフォルダ中に実行ファイルがあるので，これをクリックして実行することで REDUCE が起動する．(32 ビット版の方が安定しているらしい.）

Linux/Mac については，加古富志雄「REDUCE のすすめ」
`http://www.jssac.org/Editor/Suushiki/V18/No2/V18N2_130.pdf` 参照.

日本語マニュアル:`http://kako.ics.nara-wu.ac.jp/~kako/software/reduce/reduce37.pdf`

使い方

Reduce は対話的に入力するように作られている．

```
1:
```

に

```
1: (x+y+z)^2;
```

と入力して Enter を押すと，展開された式が出力される．セミコロンは文の終わりのマークである．

```
2: u:=(x+y+z)^2;
```

とすれば，u に $(x+y+z)^2$ が代入された状態になり，

```
3: df(u,x);
```

とすると，$u=(x+y+z)^2$ を x に関して偏微分した式が返される．

ループ処理

```
x=1;
for k:=1:10 do <<x:=x*k>>;
x;
```

とすると，$10!=362800$ が得られる．同じ結果は

```
for i:= 1:10 product i;
```

や

```
factorial 10;
```

としても得られる．

行列

```
m := mat((a,b),(c,d));
```

とすると，m に $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ が代入される．$1/m$，m^2 などとすると，それぞれ逆行列，2 乗などが求まる．

2 変数ニュートン法

以下は，連立方程式

$$\begin{cases} x^2+y^2-1 &= 0 \\ 4x^2+y^2-2 &= 0 \end{cases}$$

をニュートン法により，初期値 $(1,1)$ から解くプログラムである．

```
on rounded;
f:=x^2+y^2-1;
g:=4x^2+y^2-2;
j:=mat((df(f,x),df(f,y)),(df(g,x),df(g,y)))$

vk:=mat((1),(1))$
for k:=1:10 do <<x:=vk(1,1); y:=vk(2,1);
 vk:=vk-(1/j)*mat((f),(g))>>$
x:=vk(1,1); y:=vk(2,1);
{f,g};
clear x, y;
```

ここで，`on rounded` は結果を小数点表示するための命令，`$`は結果を出力させないためのマーク．`{f,g}`はリスト $\{f,g\}$ を表し， `clear x, y;` は x と y に代入されている値をクリアする命令である．また，for 節は, `do <<`文 `1;` 文 `2;` 文 `3 >>`という形式になっていることに注意せよ．

本来は while 文で収束を判定するべきだが，うまく書かないとプログラムが暴走するのでここでは for 文で予め反復回数を指定している．

Jacobi 法

連立方程式

$$\begin{bmatrix} 5 & 2 & 3 \\ -1 & 4 & 3 \\ 1 & 3 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 3 \end{bmatrix}$$

を Jacobi 法で解くプログラム．初期値は $(0,0,0)$ としている．

```
on rounded;
n:=3;
a:=mat((5,2,3),(-1,4,3),(1,3,6));
b:=mat((-1),(2),(3));
x:=mat((0),(0),(0));
matrix y(n,1);

for k:=1:10 do <<
  for i:=1:n do <<
    s1:= for j:=1:n sum  a(i,j)*x(j,1);
    s2:= b(i,1) - s1 + a(i,i)*x(i,1);
    y(i,1):=s2/a(i,i)>>;
  x:=y>>;
x;
clear x,y;
```

ここで，`matrix y(n,1)` は y を $n \times 1$ 行列として用意する命令．

差分法

常微分方程式の境界値問題 $y'' = p(x)y' + q(x)y + r(x)$, $y(a) = \alpha$, $y(b) = \beta$ （ただしここでは $p(x) = -x$, $q(x) = 5$, $r(x) = -20x^3 - 4x$）に対する差分解法は以下のようになる．連立１次方程式は Jacobi 法を用いる．

```
% Jacobi Method
procedure jacobimethod(a, b);
begin scalar n, k, i, j, s1, s2;
n:=part(length a,1);
matrix jacobi_x(n,1);
matrix jacobi_y(n,1);
for k:=1:100 do <<
  for i:=1:n do <<
    s1:= for j:=1:n sum  a(i,j)*jacobi_x(j,1);
    s2:= b(i,1) - s1 + a(i,i)*jacobi_x(i,1);
    jacobi_y(i,1):=s2/a(i,i)>>;
  jacobi_x:=jacobi_y>>;
return jacobi_x;
end;
```

```
% Main
on rounded;
procedure p x; -x$
procedure q x; 5$
procedure r x; -20x^3-4x$
n:=10$
a:=0$
b:=1$
alpha:=0$
beta:=0$
h:=(b-a)/(n+1)$
array x(n+2)$
for i:=1:n+2 do << x(i):=a+h*(i-1)>>;
matrix mm(n+2,n+2), bb(n+2,1)$
for i:=2:n+1 do <<mm(i,i):=2+h^2*q(x(i));
  mm(i,i+1):=-1+1/2 *h*p(x(i));
  mm(i,i-1):=-1-1/2 *h*p(x(i));
  bb(i,1):=-h^2*r(x(i))>>;
mm(1,1):=1$
mm(n+2,n+2):=1$
bb(1,1):=alpha$
bb(n+2,1):=beta$
u:=jacobimethod(mm,bb);

%Plot
v:={}$
for i:=1:n+2 do <<v:= append(v,{{x(i),u(i,1)}})>>;
load_package gnuplot;
plot v;
```

ここで procedure は関数を定義する命令で,

procedure 関数; 定義;

または

procedure 関数; begin; return 値; end;

の形式で書かれる．array は配列を宣言する命令で，append はリストを操作する（末尾に付け加える）命令である．