Octaveの使い方

東京工業大学 計算工学専攻
杉山 将
sugi@cs.titech.ac.jp　*http://sugiyama-www.cs.titech.ac.jp/~sugi/*

1　インストール

Debian, SuSE, RedHat などの Linux ユーザは，それぞれのバイナリファイルを *http://www.gnu.org/software/octave/download.html* からダウンロードできる．Mac ユーザも上記のサイトからバイナリファイルが手に入る．Windows ユーザは，*http://www.cygwin.com/* からダウンロードできる Cygwin のインストーラを実行し，インストールするパッケージに math の octave-forge を含めればよい[1]．Cygwin とは，Windows 上で Linux をエミュレートするソフトウェアである．

2　Octaveの起動

コマンドラインから octave と打ち込めば，Octave が起動する．Windows で Cygwin を使っている人は，まず Cygwin を起動し，コマンドラインから startx と打ち込んで X window を立ち上げる．そして，X window 上のコマンドラインから octave と打ち込めばよい[2]．X window を立ち上げなくても octave は起動できるが，グラフが表示されないので注意すること．

まずは Octave を電卓のように使ってみよう．

```
> (1+2)*(3-4)*5^2
ans = -75
> exp(10)*log(5)*sin(0.5*pi)
ans =  3.5450e+04
> exit
```

exit は Octave を終了するコマンドである．

3　ベクトル，行列の演算

Octave の特徴は，ベクトルや行列を直接扱うことができるところである．ベクトルや行列は四角括弧で定義する．横方向の要素はスペースで区切り，縦方向の要素はセミコロンで区切る．

```
> A=[1 2;3 4]
A =
  1  2
  3  4

> b=[3;4]
b =
  3
  4
```

ベクトルや行列の演算はそのままスカラーの場合と同じように行う．

```
> A*b
ans =
  11
  25
```

行末にセミコロンをつけると値を表示しなくなる．

```
> C=A*b;
```

単に変数名を入力すれば，変数の値が表示される．

```
> C
C =
  11
  25
```

ベクトルや行列の転置はアポストロフィをつける．

```
> C'
ans =
  11  25
```

行列の要素を取り出すときは，丸括弧で要素を指定する．

```
> A(2,1)
ans = 3
```

また，行列の要素に直接値を代入することもできる．

```
> A(2,1)=5
A =
  1  2
  5  4
```

一列（一行）の要素を全て取り出したいときはコロンを用いる．

```
> A(:,1)
ans =
  1
  5
```

ベクトルや行列のスカラ倍は，そのまま記述する．

[1] math の＋ボタンをクリックするとパッケージ名のリストが表示される．octave-forge はデフォルトで Skip となっているが，これをクリックすれば，2006.3.17-1 と表示され選択される．

[2] あるいはインストールしたディレクトリの中にある usr/X11R6/bin/startxwin.bat を直接実行してもよい．

```
> D=3*A
D =
   3   6
  15  12
```

行列の要素同士の掛け算や割り算は以下のように記述する．

```
> A.*D
ans =
   3  12
  75  48

> A./D
ans =
  0.33333  0.33333
  0.33333  0.33333
```

数列のベクトルを作りたいときは以下のようにする．

```
> e=10:5:30
e =
  10  15  20  25  30
```

2 番目の引数を省略すれば，公差は 1 になる．

```
> f=1:4
f =
   1   2   3   4
```

4 数値演算関数

Octave には沢山の数値演算のための関数が組み込まれている．例えば，

```
> cos(2/3*pi)
ans = -0.50000
```

引数にベクトルや行列を指定すれば，要素ごとにその関数の値を計算する．

```
> sin(f)
ans =
   0.84147   0.90930   0.14112  -0.75680
```

他には例えば，sin, cos, tan, acos, atan, tanh, exp, log, sqrt などがある．各関数の説明は help で参照できる．

```
> help exp
```

また，詳細は help -i で info ファイルを参照できる．

```
> help -i exp
```

ベクトル要素の最大値，最小値，和，積は，max, min, sum, prod により得ることができる．

```
> max([1 3 5 2 4])
ans = 5
```

逆行列は inv で計算できる．

```
> A=[1 2;3 4]; inv(A)
ans =
  -2.00000   1.00000
   1.50000  -0.50000
```

行列の固有ベクトル，固有値は eig で求められる．

```
> [eigvec eigval]=eig(A)
eigvec =
  -0.82456  -0.41597
   0.56577  -0.90938
eigval =
  -0.37228   0.00000
   0.00000   5.37228
```

ベクトルの平均，標準偏差，分散は mean, std, var でそれぞれ計算できる．

```
> var([1 2 3 4])
ans = 1.6667
```

0 から 1 までの間の一様乱数は rand で，平均 0 分散 1 の標準正規分布に従う乱数は randn で生成できる．引数を指定すれば，その大きさの乱数行列を生成できる．

```
> randn(1,4)
ans =
   0.22797  -1.63055   0.38559  -1.66595
```

ソートは sort 関数で行なう．

```
>[sorted index]=sort([7 3 6 1 2])
sorted =
  1  2  3  6  7
index =
  4  5  2  3  1
```

5 描画関数

Octave には沢山のグラフ作成のための関数が組み込まれている．標準的な折れ線グラフは plot 関数で描画する．

```
> x=[-2:0.1:2]; y=sin(x); plot(x,y)
```

ヒストグラムは hist 関数で描画する．

```
> a=randn(1000,1); hist(a)
```

3 次元のグラフは surf, mesh で，等高線は contour で描画する．

```
> x=[-3:0.1:3]; y=stdnormal_pdf(x); surf(y'*y);
```

グラフの保存は print コマンドで行う．グラフを LaTeX などで取り込む場合は eps 形式で保存するとよい．

```
> print -deps graph.eps
```

jpeg ファイルを作るときは，

```
> print -djpg graph.jpt
```

とすればよい．

6 Octaveによるプログラミング

これまでは，コマンドラインに直接コマンドを打ち込むことにより計算を行ってきた．より複雑な処理をさせるときには，スクリプトや関数を用いる．

スクリプト: スクリプトは，これまでコマンドラインに打ち込んでいたものをあらかじめテキストファイルに記述したものである．例えば，適当なテキストエディタで次のようなスクリプトを作成し，test.m というファイル名で保存する．

```
A=[1 2; 3 4];
b=[5;6];
C=A*b;
```

そして，Octave のコマンドライン上で

```
> test
```

と入力することにより，スクリプトの内容が実行される．

繰り返しと条件分岐: C 言語などと同じように，繰り返しには for 文を使う．例えば，1 から 100 までの整数の和を求めるスクリプトは次のようになる．

```
n=100;
a=0;
for i=1:n
  a=a+i;
end
```

Octave でプログラミングするときに，非常に重要なことは**できる限り for 文を使わない**ということである．例えば，1 から 100 万までの整数の和を求めるとき，上記のプログラムで n=1000000 とすると非常に時間がかかるが，

```
a=sum(1:1000000);
```

とすれば，一瞬で計算が終わる．

条件分岐には if 文を使う．例えば，

```
x=-10:0.1:10;
for xx=1:length(x)
  if x(xx)>0
    y(xx)=x(xx);
  else
    y(xx)=-x(xx);
  end
end
plot(x,y)
```

このプログラムも，次のようにすれば非常に高速になる．

```
x=-10:0.1:10;
y=zeros(size(x));
y(x>0)=x(x>0);
y(x<=0)=-x(x<=0);
plot(x,y)
```

そのほか while 文や switch 文もある．

関数: スクリプトの先頭行で function と宣言することによって，関数を定義できる．ここでは，2 次元正規分布の確率密度関数の値を計算する関数を定義しよう．次のファイルを g2_pdf.m という名前で保存する．

```
function z=g2_pdf(x,y,Mu,Sigma)

d=sqrt(det(Sigma));
v=[x;y]-Mu;
z=1/(2*pi*d)*exp(-1/2*v'*inv(Sigma)*v);
```

そして，以下のスクリプトを実行すれば，2 次元正規分布の確率密度関数の 3 次元プロットと等高線プロットが表示・保存される．

```
clear all
Mu=[0;0];
Sigma=[2 1;1 2];
x=[-3:0.1:3];
y=[-3:0.1:3];
for xx=1:length(x)
  for yy=1:length(y)
    z(xx,yy)=g2_pdf(x(xx),y(yy),Mu,Sigma);
  end
end

figure(1);
clf
surf(x,y,z);
view(45,60)
print -deps gauss2d_pdf_surf.eps

figure(2);
clf
contour(x,y,z);
print -deps gauss2d_pdf_contour.eps
```