J. Plasma Fusion Res. Vol.84, No.5 (2008) 243-251

講座 オープンソースソフトウェアを使った実践データ解析

6．Octaveを使ったデータ解析

松田七美男
東京電機大学

（原稿受付：2008年2月18日）

6.1 はじめに

皆さんは，複素数の数値を簡単に計算するときにどうしますか？　昔からのUNIXのツール**bc**や**awk**は複素数をサポートしていません．そんなときには，数値計算ツールを使ってみましょう．**Octave**[1-3]，**Scilab**[4]，**R**[5]などはいずれも複素数計算ができます．今回紹介するOctaveは，OSS（Open Source Software）の数値計算ツールとしてかなり有名なものです．商用のツールではMathematicaと並んで有名な**MATLAB**[6-9]の互換を標榜しています（完全ではありません）．例えば，$\sin(1+i)$の値を計算したいなら，ktermやgnome-termなどの端末エミュレータで走っているシェルのコマンドライン上で（MS-WindowsやMacユーザは，次節で説明しているようにoctaveを立ち上げ対話モードで実行してみてください．ただし，二重引用符内のoctaveの命令だけをキー入力します）

```
$ echo "sin(1+i)" |octave -q
```

のようにキー入力すれば，以下のような答えが返ってきます．

```
$ ans = 1.29846 + 0.63496i
```

RやScilabはGUIの完成度が高く，試してみるとわかりますが逆にこのようなコマンド入力での使いかたが難しくなってしまっています．octaveはコマンドラインで仕事する古き良き時代の特徴を今でも残しています．

さて，octaveの特徴は古いインターフェースというあまり自慢にならないものだけではありません．例えば，$\sin(1+i)$，$\sin(3+4i)$を以下のように一気に計算できます．

```
$ echo "sin([1+i 3+4i])" |octave -q
$ ans =
   1.2985 + 0.6350i -7.6192 - 6.5481i
```

この例では，**入力に行列を与えると同じ次元の行列が出力されること**を示したかったのです[1]．C言語などでは入力には複数の引数を与えることが許されますが，返り値（＝出力）はたった1つの変数しかとれません．複数の値を戻す場合には構造体という変数のコンテナを使わないと実現できないので，それに比べてOctaveの柔軟な入出力はとても便利です．

C言語との比較でいうと，octaveは行列の扱いがとても楽なのも特徴です．C言語では，変数は宣言しないといけません．さらに2次元配列（＝行列）は大きさが宣言時に設定された領域しか保証されませんので，たとえばAという2次元配列を$n \times m$で確保した場合に，その大きさを越えた要素に代入を行う

```
A[n+10][m]=5
```

などの演算は**絶対に避けなければならないご法度**です．なぜなら，このような操作を含むソースのコンパイルは成功して実行ファイルができてしまうので，実行時に深刻な間違いが発生するというとても質の悪いバグを潜ませることになるからです．したがって，行列の大きさは常にプログラマが管理しなければならないのです．octaveでは

```
$ echo "A=zeros(2,2), A(2,4)=1+i" |octave -q
A =
   0  0
   0  0
A =
   0 + 0i  0 + 0i  0 + 0i  0 + 0i
   0 + 0i  0 + 0i  0 + 0i  1 + 1i
```

という実行例からわかるように，宣言した大きさを越える行列の要素に書き込みをしようとすると，行列を拡大して再定義しなおしてくれるのです．これはC言語でいうところの**動的メモリの確保**を自動的に行ってくれるという，メモリの管理からプログラマを開放してくれる涙物の機能です．

この例では行列の大きさの自動確保に加えて，複素数行列への拡張も自動であることがわかります．実は，octaveでは扱う数値がdefaultで倍精度の複素数行列であるという設計になっています．数論では，数は（かなり大雑把ですが）

1　実行画面の例示は実際の表示を一部変えており，空行を詰めています．紙面の節約のためですのでご了承ください．

Practical Data Analysis Using Open Source Software 6. Data Analysis Using Octave

MATUDA Namio
author's e-mail: matuda@film.s.dendai.ac.jp

自然数 → 整数 → 有理数 → 実数 → 複素数

と収まりきれない性質を取り込んで拡張されてきましたが，逆に複素数ならばどんな数値も包含して表現できます．同様にスカラも 1 × 1 の行列とみなすことで，行列の中に包含できます．したがって，default を複素数行列とすれば，演算の規約に関して後から追加する必要はなく，統一して扱うことができるわけです．

バージョンについて 2007年度末に10年ぶりに安定版3.0.0が公開されました（それまでは2.0.17）．この解説は基本的に，その3.0.0に基づいて行います．直前の開発版2.9.xx からの変更もかなりあったようですから，iMac や MS-Windows ユーザはバイナリ版を入手してインストールしてください．Linux ユーザはバイナリパッケージができてないようですが，雑誌が刊行される頃にはパッケージを入手できると思います．

6.2 対話モードで使う

GUI がお好みのユーザは専用のウィンドウが開いてボタンがあってということであると思いますが，数値計算の記述は文章を書く作業の一種ですから，実はボタンに配置すべきコマンドはあまりないのです．数学関数などはむしろキー入力した方が早いです．しかし，普段仕事をしている端末とは別の端末を開いて，そこで常に octave を使えるようにして置くことは，まんざら悪くない使い方です．octave

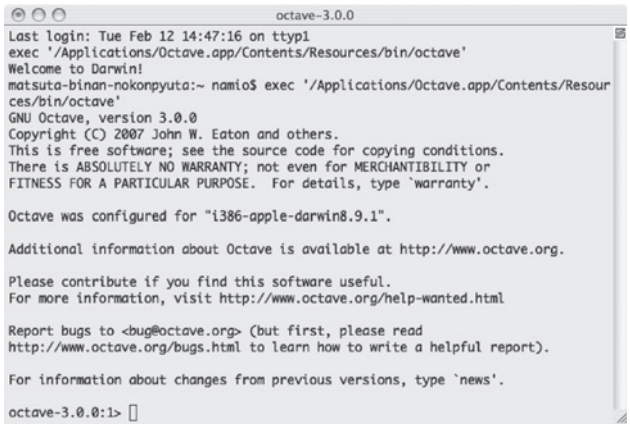

図 1 iMac での octave3.0.0 の起動画面．

が端末を占有した状態で対話的に実行させるには，単に

```
$ octave
```

とキー入力します[2]．すると**図 1** のように簡単な挨拶（オプション-q で挨拶は抜きになります）の後，プロンプト（カスタマイズ可）が表示され，octave の命令待ちとなります．ここで一つ一つ主要な命令を実行し，結果を確かめながら octave に馴染んでいきましょう．手始めに，複素数行列を定義して適当な関数値を表示させましょう．

```
octave:1> Z=[1+i 3+4j ; -1+I 4-3J], arg(Z), abs(Z), Z*Z, Z.^2
Z =
   1 + 1i   3 + 4i
  -1 + 1i   4 - 3i
ans =
   0.78540   0.92730
   2.35619  -0.64350
ans =
  1.4142  5.0000
  1.4142  5.0000
ans =
   -7 +  1i   23 + 14i
   -3 +  7i    0 - 25i
ans =
    0 +  2i   -7 + 24i
    0 -  2i    7 - 24i
```

行列は鈎括弧で括ります．その列内の要素はスペースあるいはカンマを区切りにして列挙します．行と行の区切りはセミコロンを用います．虚数単位 i が i, j, I, J と 4 通りに定義されていることも覚えておきましょう．このため，ループのカウンタ変数名として i, j は使えません．数学関数は（入力）変数が複素数行列に対応していて，出力値も複素数行列となりその要素は元の行列の要素の関数値となります．すなわち，行列 Z の要素を Z_{ij} で表現すると，

$$f(Z)_{ij} = f(Z_{ij}) \qquad (1)$$

です．Z*Z は数学の教科書どおりの行列の掛算です．Z.^2 は 2 乗を表しますが，Z*Z ではありません．ピリオド記号は行列の要素ごとの演算という**数値計算ツールに独特な演算**を示します．この場合，次のように 2 乗を要素毎に行うことを意味します．すなわち，

2 iMac や MS-Windows ではアイコンクリックで端末が開いて octave が起動するインターフェースです．

$$[Z.^2]_{ij} = [Z_{ij}]^2 \qquad (2)$$

これは次元が同じ行列間の算術演算（+, -, *, /）についてしばしば用いられる重要な演算です．加減（+, -）については元来の定義そのものですから，ピリオドは省略するのが普通です．

数値計算ツールに独特な演算をもう一つ紹介します．行列の除算についてです（これは要素毎ではありません）．数学の教科書には行列の除算は載ってません．スカラであれば除算は（大雑把にいって）**逆数を掛ける**ことで定義できますから，行列でも**逆行列を掛ける**ことで定義すれば良いと思いますが，少し問題があります．行列の乗算は一般に可換ではありません．

$$\text{スカラ：} xy = yx, \quad \text{行列：} AB \neq BA \qquad (3)$$

そこで，左から逆行列を掛ける場合を**左除算**，右から逆行列を掛ける場合を**右除算**として区別します．

$$\text{左除算：} \texttt{A\B} \longleftarrow A^{-1}B, \quad \text{右除算：} \texttt{B/A} \longleftarrow BA^{-1} \qquad (4)$$

この演算を用いれば，A を係数行列，B を定数ベクトル，X を解ベクトルとする線形連立一次方程式の解は

$$AX = B \Leftrightarrow X = A^{-1}B \longrightarrow \texttt{X = A\ B} \qquad (5)$$

のように求まります．具体的には以下の実行例で確かめてください．

```
octave:2> A=[1 2; 3 5], B=[1;1], X=A\B
A =
  1  2
  3  5
B =
  1
  1
X =
  -3.0000
   2.0000
```

6.3 スクリプトを書く

対話モードで長い計算を実行するのは厄介です．ヒストリー機能があるので，記述を間違えた場合に過去に遡って修正することはできますが，修正した後に実行をすべてやり直さなければならず，非効率的です．一般には，命令文を並べたスクリプトを作成することになります．UNIX ユーザは emacs や vi や gedit などの使い慣れたエディタで編集をしてください[3]．保存する場合には拡張子を .m とするのが一般的です．私は汎用性のないものには .oct という拡張子を使うことにしています．最初に，行列計算の実行時間計測スクリプト（**リスト 1**）を作成してみましょう．セミコロン';'やコロン':'単引用符''に注意して打ち込んでください．

リスト 1（exetime.oct）

```
N=100;

A1=zeros(N);
tic()          # for ループを用いた行列の生成
for k=1:N
  for m=1:N
    A1(k,m)=k*m;
  endfor
endfor
toc()

A2=zeros(N);
tic()          # 行列演算を用いた行列の生成
K=[1:N]';
M=[1:N];
A2=K*M;
toc()
```

記述に違いがない（#から行末まではコメントですので入力しなくとも結構です）ことを確かめて，次のように実行します[4]．

```
$ octave -q exetime.oct
ans = 0.51885
ans = 0.0033690
```

関数 tic(), toc() は，間に挟まれた命令文の実行時間を表示します．コメントにあるように行列を生成する際に，C 言語ではそれしか方法がない for ループを用いる方法と，行列演算のみ（この方法で生成できない場合も多いのですが）を用いる方法の実行時間を計測しています．実行結果から，明らかに後者が優っておりその差（比？）はざっと150倍です．$N = 1000$ にするともっと顕著になります．理由は明らかで，for ループの方法では，代入演算が N^2 回行われるのに対して，行列演算のみの方法では見た目は 3 回だからです．このように，**行列の生成時には，可能ならば for ループを避ける**ことが，octave での計算テクニックの一つです．もちろん，時間を気にしないのであれば，呪文のような行列演算に拘る必要はありません．

その呪文の方の説明に移ります．コロンは範囲指定演算子と呼ばれるものです．有り体にいえば等差数列ベクトルを生成します．

$$\texttt{X}_1\texttt{:X}_\texttt{d}\texttt{:X}_2 \longleftarrow \text{初期値 } X_1, \text{終了値 } X_2, \text{差 } X_\mathrm{d} \text{ の等差数列} \qquad (6)$$

差の項 $\texttt{X}_\texttt{d}$ が省略された場合には差が 1 と解釈されます．しかし，終了値 X_2 が初期値足す差の整数倍 $X_1 + nX_\mathrm{d}$ とは限りません．その場合は範囲内で打ち切られます．それが嫌だというなら，差の替りに個数を引数に等差数列ベクトルを

3 iMac や MS-Windows のユーザーは，OS 付属のエディターで十分用が足ります．
4 iMac や MS-Windows では保存したファイルを octave に関連づけて，クリックにより起動するお馴染みのインターフェースです．

生成する

linspace(X1,X2,N) ←X_1 と X_2 間を $N-1$ 等分割した個数 N の等差数列 （7）

を使います．

セミコロンは行列内部では行の区切りでしたが，命令文の末尾にある場合には結果の**表示を抑止**します．octave では結果を表示することが default であり，print 命令は必要ありません．したがって逆に表示が不要な場合にはセミコロンを末尾に付けて抑制する必要があります．単引用符は転置を表しますが，注意が必要です．複素数の場合には共役を取って転置するからです．単に転置を行う場合には，ピリオド演算子を用いて Z.' などとします．

最後に，for ループについて補足します．**リスト 1** では，for ループを endfor で終端してますが，本来は end のみで終端できます．実は他にも end で終端できるブロック（if, while, function, switch）があるので，明確に区別するために endfor を使いました．また，for のループ変数には任意のベクトルを与えられます．したがって，

```
for k=[1 3 4 6 8 10 12] k endfor
for a=[{"NTV"} {"TBS"} {"NHK"}]
  if !strcmp(a,"NHK") printf("%s:FREE\n",a); endif
endfor
```

のような柔軟な制御が可能です．

6.4 データのファイル入出力

タイトルがデータ解析ということですから，外部とのデータの入出力とりわけファイル入出力について説明します．保存と読み込みにはそれぞれ，save と load 命令があります．

```
save  オプション  ファイル名  v1 v2 ...
load  オプション  ファイル名  v1 v2 ...
```

6.4.1 保存：save

v1 v2 ...は対象とする変数名で，省略するとすべてのユーザ変数が保存されます．オプションは主として数値の保存形式（-text,-binary,-mat,-hdf5[5]）の指定を指示するものです．ここでは汎用性の高い文字列(-text)を用いることにします．**リスト 2** に示すスクリプトで確かめてください．作成されたファイル octave.dat を見ると，'#' で始まるヘッダーの中に変数の定義が記述されています．この情報は load の際に用いられます．また，データはフィールド幅が一定でなく見にくいかもしれません．それが気になる方のために，リストの後半に fopen() を使った C 言語と同様のファイル保存方法を示します．printf() で書式を指定してあるので，作成された plain.dat の中のデータはきれいに整列しています．

リスト 2（testsave.oct）

```
N=10;
x=linspace(0,pi,10);
y=sin(x);
Z=[x' y'];

save -text "octave.dat" Z              # octave 独自

fd=fopen("plain.dat", "wt");           # C 言語と同様
for k=1:N
  fprintf(fd,"%.15f\t%.15f\n",x(k),y(k));
endfor
fclose(fd);
```

6.4.2 読み込み：load

前節で生成した 2 種類のデータファイルを読み込むスクリプトを**リスト 3** に示します．読み込む場合にも，ファイル内のデータ構造が既知ならば読み込むべき変数名の指定 v1 v2 ...ができます．もし変数名が既に使われていた場合にはファイルのデータで上書きされてしまいます．したがって，使われてない変数に代入する書式がよいかもしれませんが，次のように構造体[6]の要素として読み込まれてしまいます．

Z1 = { Z }

このような構造体の要素を参照するには Z1.Z と記述します．plain.dat のようなヘッダがない(変数の定義情報がない）ファイルを読み込んだ場合には，構造体を生成しません．fscanf() を用いた C 言語風の読み込みも可能ですが，微妙に異なるので使わない方が無難でしょう．

5 バイナリデータの記述モデルで広く普及している：http://hdf.ncsa.uiuc.edu/products/hdf5/

6 data structure と名付けられており，要素は文字列で識別される．octave には他に cell array と呼ばれるオブジェクトがある．これは配列であり要素が添字（整数）で管理される．

リスト 3（testload.oct）

```
Z=zeros(20,2);
load "octave.dat" ;                  # octave 独自 ヘッダあり
Z1=load("octave.dat");               # octave 独自 ヘッダあり
Z2=load("plain.dat");                # octave 独自 ヘッダなし
fd=fopen("plain.dat", "rt");         # C 言語風
Z3=fscanf(fd,"%f %f",[10,2]);
fclose(fd);

Z, Z1, Z2, Z3
Z1.Z
```

6.5 グラフィック

octaveはグラフィックを内部処理せず外部ツールに任せています．現在は，gnuplot[10]が default ですが，gnuplot は独自の命令体系を有してますから，MATLAB のプロット命令系と完全互換とまではいかず，どうしても細かな差が生じます．ただし，2007年度末に公開された安定版3.0は開発版の2.1.x に比べてかなり MATLAB 風のグラフィック命令が使えるようになりました．特に，印刷命令 print がサポートされたことは喜ばしいです．mesh() を使った 3 次元表面形状プロットの例（**リスト 4**）を示します．

リスト 4（testmesh.oct）

```
u=linspace(-10,10,32);
zy=sin(u)./u;
zx=sin(u)./u;
z=zy'*zx;                            # z(i,j)=f(x(j),y(i))
mesh(u,u,z);                         # z=f(x,y) の 3 次元表面形状プロット
title("{/Helvetica=28 sin(x)*sin(y)/x/y}");
xlabel("{/Helvetica=20 X-axis}");
ylabel("{/Helvetica=20 Y-axis}");
print -depsc2 sincxy.eps
input("Quit?");                      # すぐに終了しないため
```

図 2 にiMacの画面出力（AquaTermというグラフィック端末）を，**図 3** に MS-Windows での PostScript 出力（これは OS に依存しない）を示します．

6.6 ユーザー関数

octave のユーザー関数の特徴は，扱うオブジェクトの構造がかなり自由に設定できることにあると思います．したがって定義の際に，行列に対応するようにピリオドで表記する要素毎の算術演算（'.^', '.*', './'）を適宜用いることが肝要です．

リスト 5 に，球の半径 r を引数にして，表面積 S と体積 V を出力する関数の例を示します．

実行結果は以下のようになっていて，複数の出力および行列に対応している様子がおわかりになると思います．画面出力の表示を整えるために format コマンドを用いて書式を6桁（short）の浮動小数点数表示（E）に設定していま

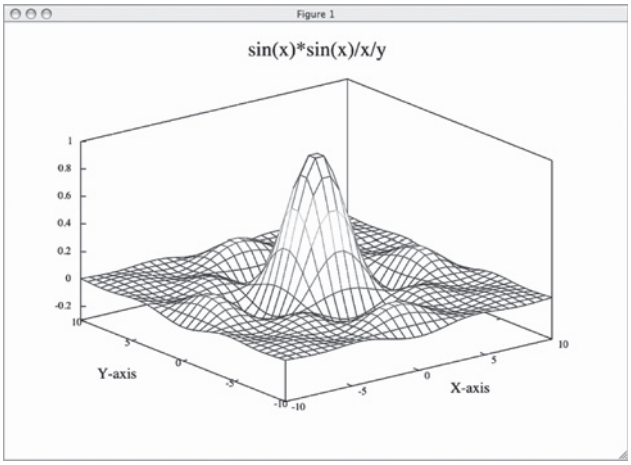

図 2　octave3.0.0/iMaC でのグラフ出力画面（AquaTerm）例．

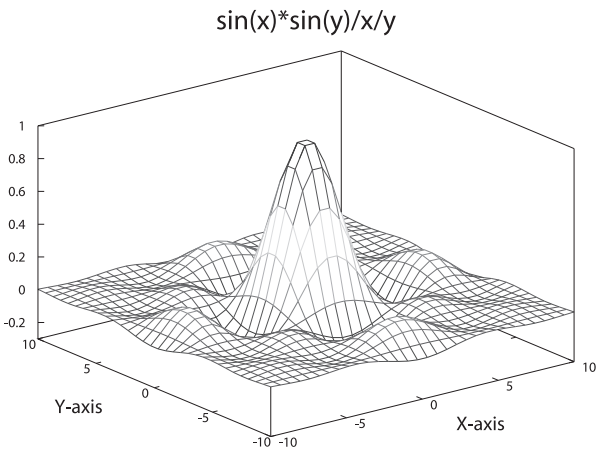

図 3　octave3.0.0/MS-Windows でのグラフ印刷（PostScript）例．

す．

リスト5

```
$ octave -q testfunc.oct
s1 =  5.0265E+01
v1 =  3.3510E+01
s2 =
   1.2566E+01   5.0265E+01   1.1310E+02   2.0106E+02
   3.1416E+02   4.5239E+02   6.1575E+02   8.0425E+02
v2 =
   4.1888E+00   3.3510E+01   1.1310E+02   2.6808E+02
   5.2360E+02   9.0478E+02   1.4368E+03   2.1447E+03
```

6.7 応用例

6.7.1 最小2乗法による線形パラメータの推定

ある実験の背景にある理論式が変数x，yと定数パラメータ$\boldsymbol{p}$を含んで$y=f(x:\boldsymbol{p})$のように陽に表現されとしましょう．このパラメータの最適推定値$\boldsymbol{p}_{\rm est}$を，測定値x_k，y_kに対して残差2乗和

$$\sum_k^N |y_k - f(x_k:\boldsymbol{p}_{\rm est})|^2 \tag{8}$$

が最小になるように決定する方法が，**最小2乗法**です．学部の実験学の講義では一次関数への回帰くらいしか学習しませんが，もう少し適用範囲は広く，結構役立つと思いますので，例にとりあげます．yを表す関数が，変数xの関数$c_j(x)$を係数とするn個のパラメータの線形和からなる式

$$y=f(x:p_1,p_1,\cdots,p_n)=\sum_{j=1}^{n} c_j(x)p_j \tag{9}$$

であるとします．このとき，N組の測定点についての2乗和

$$S=\sum_{k=1}^{N}\left(y_k-\sum_{j=1}^{n} c_j(x_k)p_j\right)^2 \tag{10}$$

を最小にするための必要条件は，

$$\frac{\partial S}{\partial p_i}=0 \qquad (i=1,2,\cdots,n) \tag{11}$$

であり．具体的には，

$$\sum_{k=1}^{N}\left(y_k-\sum_{j=1}^{n} c_j(x_k)p_j\right)c_i(x_k)=0 \tag{12}$$

のような連立方程式で与えられます．行列の形に書き換えると，

$$C\boldsymbol{p}=\boldsymbol{Y} \tag{13}$$

$$C=\begin{pmatrix} \langle c_1c_1\rangle & \langle c_1c_2\rangle & \cdots & \langle c_1c_n\rangle \\ \langle c_2c_1\rangle & \langle c_2c_2\rangle & \cdots & \langle c_2c_n\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle c_nc_1\rangle & \langle c_nc_2\rangle & \cdots & \langle c_nc_n\rangle \end{pmatrix} \tag{14}$$

$$\boldsymbol{p}=\begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix}, \qquad \boldsymbol{Y}=\begin{pmatrix} \langle yc_1\rangle \\ \langle yc_2\rangle \\ \vdots \\ \langle yc_n\rangle \end{pmatrix} \tag{15}$$

$$\langle c_ic_j\rangle=\frac{1}{N}\sum_{k=1}^{N} c_i(x_k)c_j(x_k) \tag{16}$$

$$\langle yc_i\rangle=\frac{1}{N}\sum_{k=1}^{N} y_kc_i(x_k) \tag{17}$$

となり，係数行列Cが正則ならば以下のようにパラメータ$\boldsymbol{p}$が求まります．

$$\boldsymbol{p}=C^{-1}\boldsymbol{Y} \tag{18}$$

では具体的に，半値幅σとピーク位置x_1，x_2がわかっている2つのピークの高さh_1，h_2を推定する場合を考えます（**図4**参照）．パラメータh_1，h_2を含む理論式は以下のように表されます．

$$f(x:h_1,h_2)=h_1\times\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{(x-x_1)^2}{2\sigma^2}\right] + h_2\times\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{(x-x_2)^2}{2\sigma^2}\right] \tag{19}$$

まず，正規分布乱数（`stdnormal_rnd()`）に従う誤差を理論関数に加えることで測定結果の疑似データ（`fit2gauss.dat`）を作成するスクリプトを**リスト6**に示しま

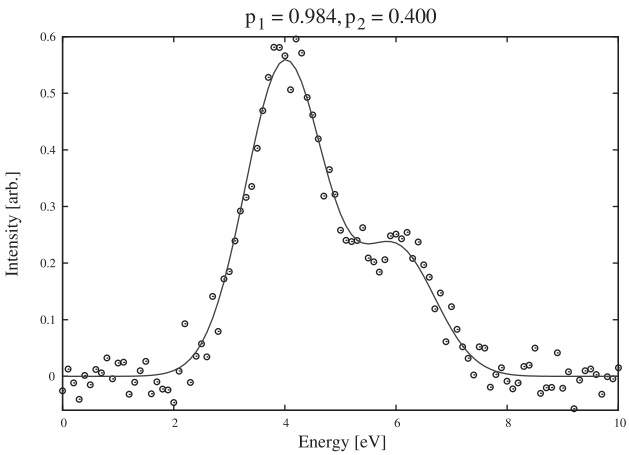

図4　近接する2つのガウシアンピークの和．

す．input()を用いてパラメータ値をキー入力できるようにしています．

リスト6（mk2gauss.oct）

```
N=101; err=0.03; s=0.7;
p1=input("p1 = ");
p2=input("p2 = ");
x=linspace(0,10,N);
y = p1*normal_pdf(x,4,s**2) + p2*normal_pdf(x,6,s**2)\
    + err*stdnormal_rnd(1,N);
z=[x' y'];
plot(x, y, "o1;;");
drawnow();
input("Save Now?");
save "fit2gauss.dat" z;
```

疑似データ fit2gauss.dat を式(19)にあてはめて，最小2乗法によりパラメータを推定するスクリプトを**リスト7**に示します．保存時に変数名 z を用いましたので，読み込み後も z を用いる必要があります．また sprintf()を使って，パラメータ値を文字列に変換してタイトルに用いています．最後の system()は，C言語にもありますが，外部コマンドを呼び出す関数です．

リスト7（fit2gauss.oct）

```
load "fit2gauss.dat"
x=z(:,1); y=z(:,2); N=length(x);
C=zeros(2); Y=zeros(2,1);
mx1=4.0; mx2=6.0; s=0.5;
c1=normal_pdf(x,mx1,s); c2=normal_pdf(x,mx2,s);

C(1,1)=mean(c1.**2); C(1,2)=mean(c1.*c2);
C(2,1)=C(1,2);       C(2,2)=mean(c2.**2);
Y(1,1)=mean(y.*c1); Y(2,1)=mean(y.*c2);
p = C\Y

xlabel("{/=20 Energy [eV]}");
ylabel("{/=20 Intensity [arb.]}");
tstr=sprintf("{/=24 p_1 = %.3f, p_2 = %.3f}", p(1), p(2));
title(tstr);
hold on
plot(x,y,"o1;;");
plot(x,p(1)*normal_pdf(x,4,s)+p(2)*normal_pdf(x,6,s),"-2;;");
% drawnow();
print -depsc2 -FTimes-Roman fit2gauss.eps
system("ggv fit2gauss.eps");
```

6.7.2 真空コンダクタンスの計算

真空においては，一般に気体流量 Q と圧力差 Δp の間に比例関係

$$Q = C\Delta p \tag{20}$$

が成立し，定数 C はコンダクタンスと呼ばれる最も重要な物理量です．真空機器を構成する部品のうちダクトはほんとんどの場合円筒ですが，分子流領域における円筒管の真空コンダクタンスの正確な値を積分方程式によって数値的に求める例を紹介します．

分子流領域においては気体分子は円筒壁間を直接往来し，途中の空間での分子同士の散乱は無視されます．このような状況下では，壁の幾何形状により通過確率 K（円筒の一端に入射した気体が多端から出ていく確率）が決定されます．円筒全体の真空コンダクタンスは，円筒の一端の開口コンダクタンス C_0（円に空間から分子が飛び込む頻度）と通過確率 K の積として，

$$C = KC_0 \tag{21}$$

と表現されます．そこで円筒壁を微小な円環に分割し，そこにおける気体分子の放出量と他の壁や開口部からの入射量が平衡になっているとして，微小円環面からの気体分

子の放出速度$q(x)$に関する以下の積分方程式を導くことができます[11].

$$q(x)=\Gamma(x)+\int_0^{\mu} q(\xi)G(\xi,x)\mathrm{d}\xi \tag{22}$$

$$\Gamma(x)=\frac{x^2+1/2}{\sqrt{x^2+1}}-x \tag{23}$$

$$G(\xi,x)=1-|\xi-x|\frac{2(\xi-x)^2+3}{2\{(\xi-x)^2+1\}^{3/2}} \tag{24}$$

ここに，μは円管の長さLと円管の直径Dとの比すなわち$\mu=L/D$，$\Gamma(x)$は開口端からの入射量，$G(\xi,x)$は他の円環(位置ξ)からの入射確率を表します．この積分方程式を解くにあたって，式(22)中の積分を単純な求積法で計算するならば，それは次のような行列方程式の計算に変換できます．

$$\boldsymbol{q}=\boldsymbol{\Gamma}+h\tilde{G}\boldsymbol{q} \quad \left(h=\frac{\mu}{N}\right) \tag{25}$$

$$\boldsymbol{q}=\begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \end{pmatrix} \quad \boldsymbol{\Gamma}=\begin{pmatrix} \Gamma_1 \\ \Gamma_2 \\ \vdots \\ \Gamma_N \end{pmatrix} \tag{26}$$

$$\tilde{G}=\begin{pmatrix} G_{11} & G_{12} & \cdots & G_{1N} \\ G_{21} & G_{22} & \cdots & G_{2N} \\ \vdots & \vdots & \ddots & \cdots \\ G_{N1} & G_{N2} & \cdots & G_{NN} \end{pmatrix} \tag{27}$$

$$x_i=\left(i+\frac{1}{2}\right)h, \quad q_i=q(x_i), \quad \Gamma_i=\Gamma(x_i), \quad G_{ij}=G(x_i,x_j) \tag{28}$$

G_{ij}，Γ_iが既知ですから，式(25)を整理して簡単にベクトルq_i（すなわち数値解）が求められます．

$$(\tilde{I}-h\tilde{G})\boldsymbol{q}=\boldsymbol{\Gamma}_i \Leftrightarrow \boldsymbol{q}=(\tilde{I}-h\tilde{G})^{-1}\boldsymbol{\Gamma} \tag{29}$$

行列計算に持ち込めれば，octaveが活きてきます．**リスト8**のような簡単なスクリプトで計算できてしまうからです(C言語ではかなり面倒です)．

リスト8（conductance-cylinder.oct）

```
function ret = Kw(u)
  ret = 1.0 - u.*(u.^2+1.5)./((u.^2+1).^1.5);
endfunction

format long g;
MU=10; DIV=2048; h=MU/DIV;
x = ([1:DIV]-0.5)'*h;                    # 縦ベクトル生成
g=(x.^2+0.5)./sqrt(x.**2+1) - x;
V=x*ones(1,DIV);                         # V(i,*)=x(i)
G = h*Kw(abs(V-V'));                     # [V-V'](i,j) = x(i)-x(j)
IG=eye(DIV)-G;
q=IG\g;                                  # 左除算によるシンプルな解法
K = 1+2*MU^2-2*MU*sqrt(MU^+1)+4*g(DIV:-1:1,1)'*q(:,1)*h
```

1 GBのメモリですと，分割数の最大値は5000程度です．行列演算だけで$\tilde{G}$を生成しています．試しにforループを使うスクリプトを書いてみてください．かなり実行が遅くなります．求めた$q(x)$を用いて通過確率Kは以下の式で算出します．

$$K=1+2\mu^2-2\mu\sqrt{\mu^2+1}+4\int_0^{\mu} q(x)\Gamma(\mu-x)dx \tag{30}$$

単にKの値が表示されるだけ(ただし，多少時間がかかります)ですが実行例を示します．

```
$ octave -q conductance-cylinder.oct
K =  0.109365346712448
```

真空工学の分野で正しいとされる値は0.109284～0.109323ですから，4桁までは一致している結果となっています．表示が6桁では不足なので，`format long g`として書式を15桁のg形式(浮動小数点数と固定少数点数を自動切替え)に設定しています．

6.8 おわりに

限られた誌面でoctaveの機能をすべて解説することはもちろんできません．なにしろマニュアルは印刷すると570ページにもなるのですから．ただ，私が普段使っていてちょっと直観とは違うぞとか，癖があるなと感じた部分を紹介しました．それ以外はほぼ直観で操作できます．そういう意味で，気軽に数値計算をしてみるには丁度良い機能のツールであることを感じていただければ幸いです．

参 考 文 献

［1］Octave の公式サイト：http://www.octave.org/
［2］大石進一：Linux 数値計算ツール（コロナ社，2000）．
［3］松田七美男：雑誌 Linux Japan の連載記事「もう少しだけ Linux」アーカイブ（http://ayapin.film.a.dendai.ac.jp/ matuda/TeX/lj.html）
［4］Scilab の公式サイト：数学の王国フランスの国立研究所 INRIA（Institut National de Recherche en Informatique et en Automatique）が開発しており，octave よりも完成度が高い（http://www.scilab.org/）
［5］R の公式サイト：ベル研で開発された統計解析言語 S に一歩及ばないので R と名付けたとか（http://www.r-project.org/）
［6］Matlab を開発している Mathworks 社のサイト（http://www.mathworks.com/）
［7］大石進一：MATLAB による数値計算（培風館，2001）．
［8］アレジェンドロ・ガルシア 著，畑崎隆雄 訳：MATLAB/C++で学ぶ物理学のための数値法 上・下（ピアソン・エデュケーション，2004）．
［9］芦野隆一／Rémi Vallancourt：はやわかり MATLAB（共立出版，1997）．
［10］gnuplot の公式サイト：http://www.gnuplot.info/
［11］松田七美男：真空 **47**, 690 (2004).