講座
コンピュータの利用技術

2．データの可視化

松　田　七美男
（東京電機大学）

Data Processing and Visualization

MATSUDA Namio
Faculty of Engineering, Tokyo Denki University, Tokyo 101-8457, Japan
(Received 1 November 2001)

Abstract

This article introduces how to process and visualize scientific data with using "Gnuplot" which is freely available plotting utility from the Internet. Not only a simple example of drawing a single plot on a page but also advanced feature of handling data stream and communication to the external application is explained.

Keywords:
scientific visualization, data processing, plotting utility, PC-Unix, gnuplot

2.1　はじめに

計測データをその場ですぐに確認することは，学生実験では基本事項として教わります．ところが本格的に研究をすすめると測定データをリアルタイムに表示するシステムができあがっていないような場合，ついデータ整理が後回しになってしまうことがあります．実験自体が複雑であるとか操作がマニュアル化されているような場合，人間がデータを合間にチェックすることは非常に重要です．そのために，機能は高くなくとも小回りの利く使いなれたツールが欲しいと思うことでしょう．

Gnuplot[1]はそのような気の利いたグラフ化ツールです．論文に掲載するような奇麗な図面を描くという用途にはあまり向きません（他の整形ツールと組み合わせればある程度可能です）．基本的ないくつかのコマンドを使って即座に図を描き，理論曲線と比較して実験結果を吟味するという目的を叶えてくれます．実例を通じて，この小粋なツールを使いこなす技を紹介することが本稿の目的です．

2.1.1　インターフェースの違い

一般に，最近のOSはユーザインターフェースにウィンドウシステム（MSは不可分，UnixならばX Window System）を備えているのが常識です．しかも，図面や画像の類いは，それを表示させながらマウスでクリックして処理を施すという作業がすぐに思い浮かびます．しかし，よく考えてみると前述のような用途では，マウス操作はわずらわしいと感じるようになるでしょう．ボタン操作は，コマンド入力の代替え以上の機能がなく，コマンドに馴れたユーザにとっては時間がかかるだけのものだからです（もちろん，打ち間違えが発生しないので初心者には親切であるには違いありません）．また，定型化された図を何枚も作成し，それをざっと眺めて実験全体の推移を把握するというような場合には，データを次々

author's e-mail: matuda@film.s.dendai.ac.jp
著者のご協力により，この講座の掲載内容についてのWebページが開設されています．本文中にあるデータファイルやC言語のソースファイルを入手可能ですので，ぜひ http://jspf.nifs.ac.jp/をごらんください．

に送り込んでバッチ処理しなければなりません．その場合には**データの流れ**を柔軟に制御できるシステムあるいはツールが求められるのです．

2.1.2 Gnuplot の特徴

以上のような理由から，コマンドライン指向のツール gnuplot を取り上げて，グラフ化の実際を紹介していきます．他にもいろいろなツール，特に MS-Windows 用の GUI を標準としたツールがあるのですが[2]，スクリプトでバッチ処理的に使うにはやや工夫が必要です．その点，gnuplot は Unix のシェルの**リダイレクト**や**パイプ**機能と組み合わせて，柔軟なデータ処理が行えます．この際，是非 PC-Unix[3]あるいは MS-Windows 用の Unix ツール[4]をインストールすることをお薦めします．インストール自体は，大変楽になっており，かつ特殊なハードウェアでない限り必ずインストールできるようになっています．

2.1.3 ドキュメント類

本稿だけで gnuplot のすべての機能を紹介することは無論不可能です．比較的ドキュメント類が充実していますから，随時それらを参照してください．起動中ならば，**help** コマンド（MS-Windows ならば Help メニューボタン）が使いやすいです．また，gnuplot.pdf はきれいに整形されており読みやすいです(英語ですが)．残念ながら配布されている入門書は古く実用的ではありません．そのかわり長大な demo（all.dem を実行してください）があります．ひととおり眺めれば gnuplot でどんなグラフを描くことができるかほぼわかります．もちろん，スクリプトファイルは大変良いコマンドの使用例の見本となっています．それらを読破するならば本稿も不要といえましょう．

2.1.4 擬似データファイルの準備

どのツールを使うにしても，実際に動かしてみないことには善し悪しがわかりません．ですからこの解説も是非 gnuplot を動かしながら読んでいただけると良いと思います．そのために，できれば擬似データを作成しておいてください．以下は C 言語で作成する場合の例です．List.1 の mksample.c をエディタで編集し，コンパイルしてできあがった実行ファイルを走らせます．本稿では，このような擬似データを MS-Windows でも実行できるように C 言語を用いて作成していきます．もちろん，他に馴染みのツールがあればそちらを使ってください．筆者はこのような簡単な数値処理には通常 awk を用いています．

```
cc -o mksample mksample.c -lm
./mksample
```

List.1 mksample.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define MAXDATA 100

main()
{
  int i;
  double x;
  FILE *fd;

  fd = fopen("sample.dat","w");
  for (i = 0; i < MAXDATA; i++) {
    x = M_PI*i/(double)MAXDATA ;
    fprintf(fd,"%.4g\t%.4g\t%.4g\t%.4g\n",
            x,exp(-x),1/(x*x+1),cos(x/2));
    printf("%.4g\t%.4g\t%.4g\t%.4g\n",
            x,exp(-x),1/(x*x+1),cos(x/2));
  }

  fclose(fd);
}
```

2.2 2 次元のグラフ：plot

2.2.1 基本的な使い方

擬似データファイル sample.dat は，次のように最も一般的なデータファイル構造を持っています．これを読み込んで 2 次元グラフを描く方法を説明します．

$$
\begin{array}{llll}
x_1 & y_1 & z_1 & \ldots \\
x_2 & y_2 & z_2 & \ldots \\
\ldots & & & \\
x_i & y_i & z_i & \ldots \\
\ldots & & &
\end{array}
$$

まず，gnuplot を起動しましょう．MS-Windows ならば gp371w32 のアイコンをクリック，PC-Unix ならば端末エミュレータ上で gnuplot をキー入力します．する

と，Fig. 1のようにgnuplotが立ち上がりプロンプト

```
gnuplot>
```

を表示して，コマンド入力待ちとなります．それに対して対話的に(interactive)コマンドをキー入力していくというのがPC-UnixとMS-Windowsの両者に共通の基本操作です．MS-Windowsでは，上部にメニューボタンがあり，コマンドをボタンの押し下げで実行できる点が初めてのユーザには心強く，親切設計といえましょう．

では，コマンドをキー入力してグラフを作成してみましょう．以下，紙面の節約のため行の先頭のプロンプトgnuplot>は省略します．もっとも単純な例は

```
plot "sample.dat"
```

というものです．データファイルの第1列を横軸，第2列を縦軸にした2次元プロットを描きます．ファイル名は，ダブルクォート，シングルクォートどちらでも括ることができます．ただし，MS-Windowsでは，必ずしもカレントディレクトリを認識しませんので，ファイルの**絶対パス**を指定したほうが無難です．さらに，ダブルクォートで括った場合，ファイルの先頭に付くバックスラッシュをエスケープしなければなりません．すなわち，c：ドライブのgp371w32ディレクトリにある場合には

```
'c:\gp371w32\sample.dat' あるいは
"c:\\gp371w32\\sample.dat"
```

となります．以降この違いは省略し，PC-Unixの記述法に従いますので，MS-Windowsで動作させている方は注意してください．

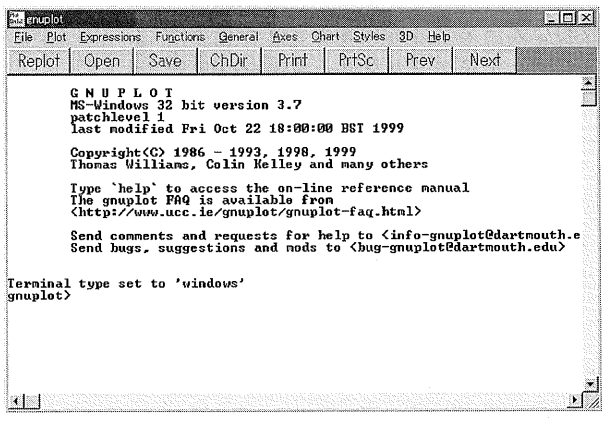

Fig. 1 Starting text-mode windows terminal on MS-Windows.

第1列を横軸，第3列を縦軸にしたグラフを描くには，**列指定子 using** を用います．

```
plot "sample.dat" using 1:3
```

データに対して演算を行うこともできます，その場合には列番号に**$**を付けて引用し，さらに括弧で囲む必要があります．

```
plot "sample.dat" using 1:(log($2))
```

複数描くには，','で区切って列挙します．sample.datのデータをすべてプロットさせてみたのが，Fig. 2です．コマンドは

```
plot "sample.dat", "" using 1:3, "" 1:4
```

です．同一のファイルに対しては，名前の省略が可能(ただしクオートは省略できません)です．

a 対数軸

軸を対数にするには,"set logscale *"コマンドを用います．

```
set logscale y
plot "sample.dat"
```

とすると，y軸が対数目盛りになって，直線が描かれます．

b 描画範囲

描画範囲はplotコマンド実行時の指定が最優先されますが，"set *range [min:max]"と予め設定することもできます．以下の実行例で確かめてください．

```
set xrange [0:2*pi]
set yrange [:3]
plot "sample.dat", "" using 1:3
plot [-pi:pi][:] "sample.dat", "" using 1:3
set autoscale xy
plot "sample.dat", "" using 1:3
```

自動で適切な範囲を設定するコマンド"set autoscale *"も覚えておくと便利です．

c ラベル

グラフ全体のタイトルは"set title "TITLE"", x, y軸の名前は"set x(y) label "X(Y) LABEL""で設定します．データ曲線の名前(gnuplotではkeyと呼ばれます)は，plotコマンドのオプションtitleで設定できます．また，任意のラベルを出力するコマンド"set label "LA-

BEL" at xp,yp"も用意されています．それぞれ，書体指定などのもっと細かいオプションがありますので，

```
help title
help label
help key
```

を実行してみてください．

d サイズ

グラフの大きさは"set size x,y"で設定できます．ただし，ラベルを含んだ全体の大きさですから，枠の大きさのみを思ったとおりにとはいきません．筆者の経験では，枠を正方形にしたい場合に悩みが深くなります．その場合には，大体でよければ"set size square"とします．

```
set logscale xy
set xrange[0.01:10]
set yrange[0.01:10]
set grid
set size square
plot "sample.dat" unsing 1:1, "" using 1:($1
**0.5)
```

e 画像出力

描画されたグラフを画像形式に保存するには，スクリーンダンプという方法がありますがそれは最終手段です．gnuplot は非常に多くの出力形式に対応していますから，例えば TeX に取り込む予定があるなら，拡大縮小が自由なEPSとしてファイルに出力するのが良いでしょう．

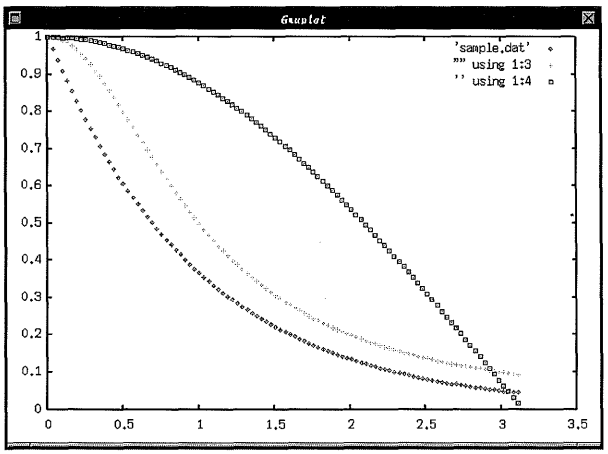

Fig. 2 Plotting all the data in sample.dat on X11 terminal.

```
set terminal postscript eps
set output "***.eps"
replot
```

とします．これ以外にどのような画像形式やプリンタ言語に対応しているかは"set terminal" とすると一覧が表示されます．

f 関数の描画

gnuplot は通常の用途には十分な数学関数を内蔵しており簡単にグラフを描くことができます．また，媒介変数モード (parametric mode) がありますので，多様な描き方が可能となります．次の例題を実行してみてください．

```
set grid
set size square
plot [-3:3] sin(x), exp(-x)
set parametric
set trange[0:10]
plot t,exp(-t), exp(-t),t
```

g 2つの x，y 軸

gnuplot では plot コマンドの axis オプションにより x,y 軸を 2 通り指定することが可能です．以下のコマンドで確認してください (Fig. 3)．

```
plot sin(x), 10*cos(x) axis x2y2 lt 0 lw 3
set x2tics
set y2tics
set xtics nomirror
set ytics nomirror
set y2range[-20:20]
set x2range[-pi:pi]
replot
```

最初のコマンドだけですと，$x2$, $y2$ 軸の目盛りが付きません (付いているのは $x1$, $y1$ 軸の目盛りです)．$x2$, $y2$ 軸に目盛りを付けるコマンドは"set *2tics" です．また $x2$, $y2$ 軸についてしまった余分な $x1$, $y1$ 軸の目盛りをを"set *tics nomirror" で抑止します．$x2$ 軸と $y2$ 軸の描画範囲は plot 時に指定することができない場合がありますから，あらかじめ設定しておいた方が良いでしょう．なお，軸の組み合わせは，$x1y1$ (既定)，$x1y2$, $x2y1$, $x2y2$ と 4 通り可能です．

h スタイル

描くグラフの種類，線種や記号を変えるにはオプショ

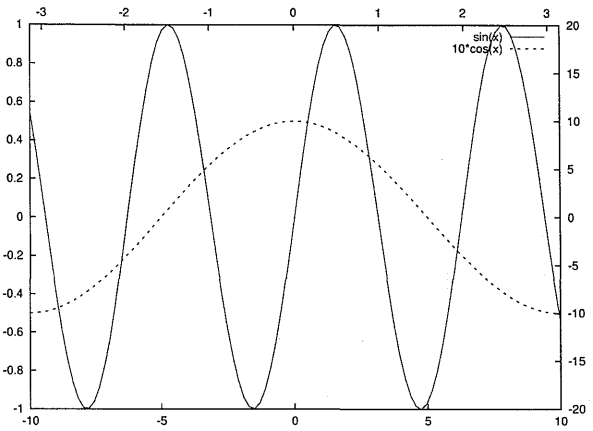

Fig. 3 Example of using first and second set of axes.

ン“with”以降に**順番を守って**記述します．例えば，

```
plot sin(x) with lp lt 0 lw 2 pt 6 ps 3
```

とすると，種類は折れ線と記号(lines-points)，線種(lt)は0，線幅(lw)が2，記号の種類(pt)が6，記号の大きさ(ps)が3のグラフを描きます．グラフの種類は，

```
lines（関数描画時のデフォルト），points（データ描画時のデフォルト），
linespoints,impulse,dots,steps,boxes
```

の他に，列が3列以上必要な

```
xerrorbars,yerrorbars,xyerrorbars,vector
```

などがあります．例題として，対話的にベクトル図を描いてみましょう．“with”は“w”と省略できます．

```
gnuplot> plot [0:1] [0:10] "-" w vector
0 0 1 1
2 2 0 1
2 4 1 0
e
```

特別なファイル名“-”を指定すると，標準入力（ここでは，キーボード入力）からデータを読み込みます．行の先頭で“e”を入力すると，読み込みが終了して，グラフが描画されます．vectorはベクトル $\vec{A}$ の開始位置の x，y 座標と，ベクトル自身の成分 A_x，A_y，すなわち4つの数値を必要とします．

2.2.2 multiplot

独立な複数のグラフを1つのページに描くこともできます．“set multiplot”で始まって，“set nomultiplot”で終了するネストの内部に，個々のグラフの原点の位置（set origin）と大きさ（set size）を指定しておき，それぞれplotを実行させます．ちょっと長いですが，Fig. 4は次のスクリプトで描いたものです．

```
set xrange [0:2*pi]
set xzeroaxis
set samples 25
f(x)=exp(-0.2*x)*cos(x)
set multiplot
set origin 0,0
set size 0.33,0.5
plot f(x) t "lines" w lines
set origin 0.33,0
set size 0.33,0.5
plot f(x) t "points" w points ps 3
set origin 0.66,0
set size 0.33,0.5
plot f(x) t "linespoints" w lp lw 2 ps 3
set origin 0,0.5
set size 0.33,0.5
plot f(x) t "boxes" w boxes
set origin 0.33,0.5
set size 0.33,0.5
plot f(x) t "steps" w steps
set origin 0.66,0.5
set size 0.33,0.5
plot f(x) t "impulses" w impulse
set nomultiplot
pause -1
```

2.2.3 関数曲線への当てはめ：fit

線形・非線形を問わず，最小自乗法によりデータを理論曲線に当てはめる機能が備わっています．パラメータp1, p2, p3を含む理論曲線を $f(x)$，その実測データを＊＊＊.datとすると，

```
fit f(x) "***.dat" via p1,p2,p3,...
```

により，最適パラメータの推定ができます．

2つのピーク（gaussianを仮定します）の，中心位置と高さと分散を求める簡単な例題を示しましょう．まず擬似データを準備します．List.2に示すgauss2.cを編集してコンパイルにかけ，擬似的な実験誤差を引数に与えてgauss2.datを作成してください．

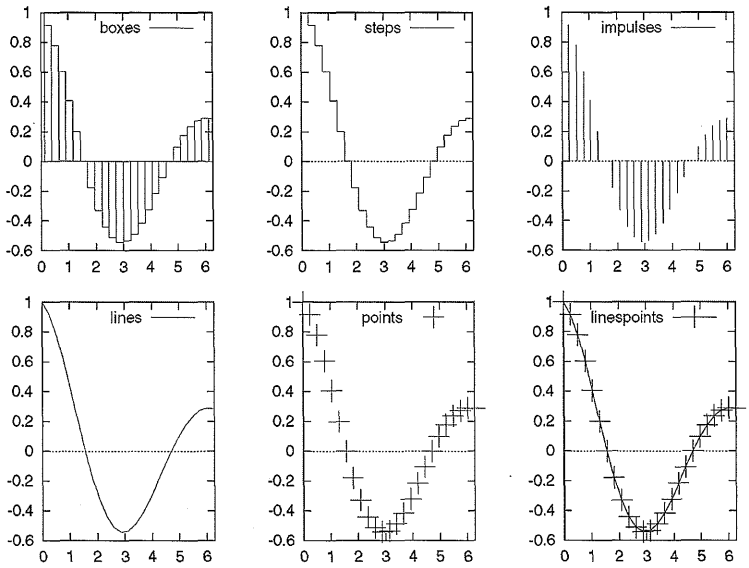

Fig. 4 Example of multiplot mode, in which several plots are placed on the same page.

```
gcc -o gauus2 gauss2.c -lm
gauss2 0.06
```

List.2 gauss2.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

#define MX   1.0
#define S    0.2
#define H    1.0
#define IMAX 200
double addrand(double x, double e)
{
  return x*(1 - e/2 + drand48()*e);
}

double gauss(double x, double mx, double s)
{
  return exp(-(x-mx)*(x-mx)/(2*s*s))
         /(s*sqrt(2*M_PI));
}

main(int argc, char **argv)
{

  int i;
  double x, g, eps;
  time_t *now;
  FILE *fd;

  fd = fopen("gauss2.dat", "w");
  now = NULL;
  srand48(time(now));

  eps = atof(argv[1]);

  for (i = 0; i < IMAX; i++){

    x = MX*2.0/IMAX*i;
    g = addrand(gauss(x,MX,S)
        + 0.2*gauss(x,MX/2,S), eps);
    fprintf(fd,"%f\t%f\n", x, g);
    printf("%f\t%f\n", x, g);
  }
  fclose(fd);
}
```

List.3 gauss2.plt

```
g1(x)=A1*exp(-(x-MX1)**2/(2*S1**2))/
/(S1*sqrt(2*pi))
g2(x)=A2*exp(-(x-MX2)**2/(2*S2**2))/
/(S2*sqrt(2*pi))
g(x)=g1(x)+g2(x)
MX1=1
MX2=0.2
S1=0.1
S2=0.1
A1=1
A2=0.1
fit g(x) "gauss2.dat" via A1,MX1,S1,/
A2,MX2,S2
plot g(x) lw 2 , "gauss2.dat" pt 6 ps 2
pause -1
```

続いて，List.3 に示す gnuplot のスクリプトファイル gauss2.plt を gnuplot で実行しますと，次のように繰り返し計算をしてパラメータ推定の最終結果が表示されます．途中経過が気になる方は，fit.log というファイルができますからそれを見てください．

```
gnuplot gauss2.plt
...

Final set of parameters   Asymptotic Standard Error
======================   ==========================
A1      = 0.997821       +/- 0.005714 (0.5726%)
MX1     = 1.0009         +/- 0.001176 (0.1175%)
S1      = 0.198563       +/- 0.0008375 (0.4218%)
A2      = 0.205105       +/- 0.005741 (2.799%)
MX2     = 0.505366       +/- 0.005866 (1.161%)
S2      = 0.201389       +/- 0.00425  (2.11%)
```

gauss2.c 内で定めたパラメータの値に近い推定値が得られました．Fig. 5 に擬似データとの一致具合を示します．

一般の実験結果の吟味においても，精密なパラメータ推定は大型計算機センターの統計パッケージに任せるとして，その場ですぐに検査できるという意味で，fit は大変重宝します．

2.3 3次元のグラフ：splot

3次元の俯瞰図を描くにはsplotを用います．要求されるデータは，$m \times n$ の格子状のデータで，具体的には m 行の x, y, z の並びを空白をはさんで n 個繰り返し記述します．もちろん Fig. 6 に示されるように，関数 $z=f(x,y)$ を描くこともできます．

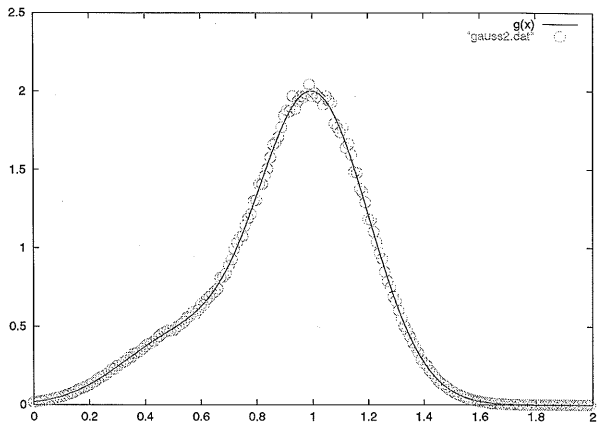

Fig. 5 Using "fit" command on a set of data point which seems like to have two gaussian peaks (very weak peak on the shoulder of strong one).

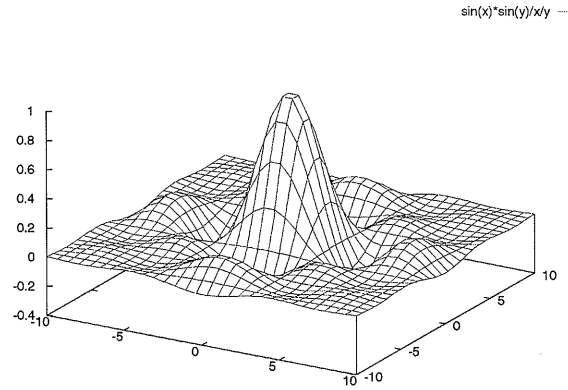

Fig. 6 Perspective plotting of a function $z(x,y)=\sin(x)\sin(y)/(xy)$ by "plot" command

2.3.1 視点

図に対する視点は set view コマンドで設定します．引数は4つあって，x 軸の回りの回転角，z 軸の回りの回転角，縮尺，z 方向の縮尺をそれぞれ指定します．"set view 0,0"の状態が，xy 平面を z 軸の正方向から観る視点となります．この状態から座標軸をまず x 軸まわりに回転させ，続いて回転後の新しい座標系の z 軸回りに回転させることになります．簡単なデモを動かしてみましょう．次のスクリプト rotmain.plt，rotsub.plt を用意してください．

```
gnuplot rotmain.plt
```

とすると，3次元グラフがx軸まわりの回転を一定にして，z軸まわりで回転を行います．少しスクリプトの説明をします．gnuplotはプログラミングをしようとすると制御構造が貧弱であまり実用的ではありません．ただし，いくつかのコマンドが用意されています．この例では，他のgnuplotスクリプトを呼び出すコマンド**load**を用いています．また，条件分岐**if**と**reread**（そのスクリプトを最初から実行させる）を組み合わせると繰り返し制御がある程度できますから，この例題のようなアニメーションが可能となります．詳しくはhelpで参照してください．

rotmain.plt

```
set isosamples 25
set hidden3d
set ticslevel 0
set size 0.7,1
f(x,y)=sin(x-3)*sin(y)/(x-3)/y
set title 'rx=0'; rx=0; rz=0
load 'rotate.plt'
set title 'rx=30';rx=30; rz=0
load 'rotate.plt'
set title 'rx=60';rx=60; rz=0
load 'rotate.plt'
set title 'rx=90';rx=90; rz=0
load 'rotate.plt'
set title 'rx=120';rx=120; rz=0
load 'rotate.plt'
```

rotsub.plt

```
set view rx,rz
splot f(x,y)
rz = rz+20
pause 1
if(rz<=360) reread
```

2.3.2 等高線図

等高線図を描くことも可能です．ただし，残念ながら等高線にラベルを割り込ませて表示する機能はありません．等高線を描く位置を，グラフの表面（surface）やxy平面（base：デフォルト）あるいは両方（both）に設定できます．次の例題で確かめてください．ただし，表面に描いた等高線は識別しにくいです．

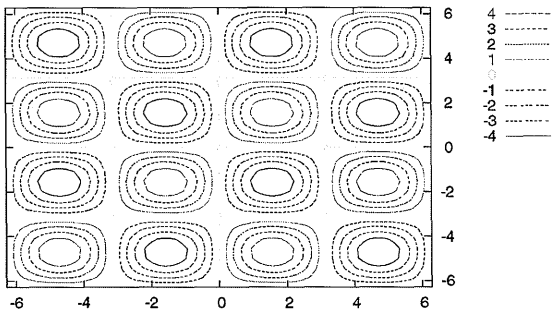

Fig. 7 Drawing a contour for the surface of the function $z\,(x,y) = 5\sin(x)\sin(y)$ by "splot" command.

```
set isosamples 30
set contour both
c=2*pi
splot [-c:c][-c:c] 5*sin(x)*sin(y)
set view 0,0
set noztics
replot
set nosurface
replot
```

俯瞰図ではなく，地図のようにxy平面をz軸から見たものの方が良く用いられます．それには視点を変更するだけではなく，グラフの表面を描かないようにしなければなりません（set nosurface）．Fig. 7に，この例題の$\sin(x)*\sin(y)$の等高線を示します．

2.3.3 非グリッドデータ

splotは基本的には格子状のデータを要求しますが，必ずしも測定データがそれに見合うものではない場合もあるでしょう．set dgrid3dを設定すると，各行のx,y,zを曲面上の座標とみなして，内部で補間により格子状データを生成して図を描きます．これも使い方によっては大変重宝します．List.6のような擬似データを作成するプログラムdgrid3dを実行し，dgrid3d.datを用意してください．この擬似データを基にしてdgid3dをセットして曲面を描きます．

```
set dgrid3d 20,20
set ticslevel 0
set hidden3d
set zrange [0:10]
splot "dgrid3d.dat" w lines
pause -1
```

List.6 dgrid3d.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

#define IMAX 800

main(int argc, char **argv)
{
  int i;
  double x, y, g;
  FILE *fd;
  time_t *now;

  fd = fopen("dgrid3d.dat","w");
  now=0;
  srand48(time(now));


  for (i = 0; i < IMAX; i++){
    x=3.0*(drand48()-0.5); y=3.0*(drand48
    ()-0.5);
    g = 1/(x*x+y*y);
    fprintf(fd,"%f\t%f\t%f\n",x,y,g);
    printf("%f\t%f\t%f\n",x,y,g);
  }
  fclose(fd);
}
```

2.4 非対話的な使い方

これより先は主としてPC-Unixのシェルの機能を使ってデータの流れを制御する例を紹介します．申し訳ありませんが，MS-Windowsでは実行しにくいものです．MSな方は通読して，“なるほどシェルは便利だ”と納得したならPC-Unixのインストールを試みてください．

2.4.1 パイプによるデータの受け入れ

gnuplotでグラフ化することに馴れてくると，いちいちgnuplotを起動してからコマンドを打ち込むのすら遅い．もっと素早くグラフを得る方法がないか考えるようになります．それには，データの流れ（これはどうしても省略できません）を直接gnuplotに渡して描画することを考えます．PC-Unixでは，シェルの機能を用いて実現可能です．例えば，

```
echo 'plot "sample.dat"' |gnuplot -persist
```

とすると，gnuplotに直接命令を送ることができます．この場合，オプション－persistを付けて起動する必要があります．以上は出来上がったファイルを利用していますが，sample.datを出力するプログラム，mksampleはまさにデータを流しますから，

```
(echo 'plot "-" using 1:3'; ./mksample) |
gnuplot -persist
```

という風に使うことも可能です．左側の二つのコマンドを括弧( )で括ると，両方の出力がパイプに接続されます．括弧がないと，最後のコマンドの出力だけがパイプに接続されてしまい期待どおりになりません．シェルの機能を，gnuplotの内部から利用することも可能で，

```
plot "< ./gauss2 0.06"
```

などとして，外部アプリケーション（この場合はgauss2）の標準出力を読み込むことができます．このように，シェルのデータ入出力の切り替え機能と組みあわせて，大変柔軟な処理が可能であることが理解できたと思います．

2.4.2 プログラムからの呼び出し

gnuplotはプログラミング言語ではありませんから，分岐制御(whileやforやswitchなど)がほとんどできません．また，文字列変数を持っていません．したがって，パラメータを変化させて，あるいはファイルの名前を変化させて次々と図を描くというような事は単独ではできません．このような要求に応えるには，他の言語で制御構造を記述し，そこから呼び出されるようにスクリプトを組むのが最も自然です．PC-Unixならば，シェル(bash，tcsh，zsh)やawk，perl，ruby，tcl/tk，pythonなど枚挙に暇がありません．ここでは，シェルを利用する例と，Cでpopen( )を使う例を示します．

まず，シェルを用いる例ですが，ディレクトリにある定型のデータファイル***.datを5秒置きに次々と表示するスクリプトを考えましょう．ファイル名を変数にして，gnuplotに与えれば良いので，素直には次のようになるでしょう．

```
#!/bin/sh

for dataname in `ls *.dat` ; do
   echo "plot  \"$dataname\"; pause 5" |
   gnuplot
done
```

しかしこれはグラフを1つ描く度にgnuplotも起動／終了しますから，少し繁雑です．gnuplotは1つだけ起動しておき，そこに次々とグラフを描かせたいものです．それには，

```
#!/bin/sh
for dataname in `ls *.dat` ; do
   CMD="$CMD plot \"$dataname\"; pause 5; "
done
echo $CMD |gnuplot
```

などが考えられます．ただあまりたくさんのファイル名を与えようとすると，非常に長いコマンド文字列となり，$CMDにおさまり切らないかもしれません．

C言語でpopen( )関数を用いると，gnuplotを1つだけ起動して，そこに次々と命令を送り込むことが簡単に実現します．gnuplotはデータをすべて読み込まないと描画しませんから，時系列データなどを表示するような場合には，データを適当にまとめる工夫が必要です．あるいは途中でバッファリングしておき，一定間隔でまとめてgnuplotに渡す方法も考えられます．List.5のibuffer.cまさにそのようなプログラム例です．wave.cは永遠に擬似データを送出するプログラムで，

```
./wave 10000 | ./ibuffer
```

と実行してみてください．

List.5 ibuffer.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define BUFFSIZE 64

main()
{
 int i;
 double x, y;
 double data[BUFFSIZE][2];
 FILE *GP;

 for (i=0; i < BUFFSIZE; i++){
  data[i][0]=0;
  data[i][1]=0;
 }

 GP=popen("gnuplot -geometry 384x384","w");
 fprintf(GP, "set yrange[-3:3]\n");
 fflush(GP);
 usleep(100000);

 for (i=1; i < BUFFSIZE; i++){
  scanf("%lf %lf", &x, &y);
   data[i][0]=x;
   data[i][1]=y;
 }

 while(1){
  fscanf(stdin,"%lf %lf", &x, &y);
  for (i=0; i < BUFFSIZE-1; i++){
   data[i][0]=data[i+1][0];
   data[i][1]=data[i+1][1];
  }
  data[BUFFSIZE-1][0]=x;
  data[BUFFSIZE-1][1]=y;

  fprintf(GP, "plot '-' with lp lt 1\n");
  fflush(GP);
  usleep(100000);
  for (i = 0; i < BUFFSIZE; i++){
   fprintf(GP, "%f %f\n",data[i][0],
   data[i][1]);
  }
  fprintf(GP,"e\n");
  fflush(GP);
  usleep(100000);
 }
 fclose(GP);
}
```

List.6 wave.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <math.h>

main(int argc, char **argv)
{
  double t;
  int i = 0;

  setlinebuf(stdout);
  while (1){
    usleep(atoi(argv[1]));
    i++; i = i % 1000;
    t=0.01*M_PI*i;
    printf("%f %f\n",t,
    sin(t)*exp(cos(M_PI*t)));
  }
}
```

a　他のツールからの利用

以上のようにライブラリ感覚で使うことができますので，数学ツールOctave[5]，Maxima[6]など，グラフィック機能をgnuplotの呼び出しで実現しているものが多数あります．

2.4.3　リモート観測

これはgnuplotの機能というよりも，X Window Systemの**ネットワーク透過性**を利用するもので，全てのXクライアントに共通の使い方です．すなわち，リモート先で測定データを蓄積し，それをgnuplotで随時グラフ化する際に画面をローカル側のXサーバーに描かせることができるのです．他にVNC[7]などでも可能ですが，X11の方が速いですし，ssh（Secure SHell）などを使えばデータが暗号化されますから安全です．

2.5　まとめ

その場でデータをグラフ化して検証するというような目的に最適の小さなツールgnuplotの基本操作と少し進んだ使い方を紹介しました．実験装置が巨大化し，得られたデータの解析も最終的には大型計算機でしかできないというような状況があればこそ，こまめに直にデータを眺めることが必要，というのは言い過ぎでしょうか．gnuplotに限らず，使いこなせるツールが一つあると，そのような作業が楽にもなるし，また楽しくもなるでしょう．

参考文献

[1] Gnuplotの公式サイト：http://www.gnuplot.org/は手違いで閉鎖中．臨時公式サイトhttp://www.gnuplot.vt.edu/

[2] 例えば，有名なMS-Windowsのソフトウェアを紹介するサイト“窓の杜”などで調べられます．http://www.forest.impress.co.jp/

[3] PC-Unixの2大潮流は，LinuxとFreeBSDです．雑誌も数多く出まわっています．http://www.linux.or.jp, http://www.FreeBSD.org/ja/

[4] MS-WindowsでUNIX環境を構築するためのツール群：http://www.cygwin.com/

[5] 高級数値計算ツールOctave:http://www.octave.org/

[6] シンボリック数式処理システムMaxima：http://www.ma.utexas.edu/users/wfs/maxima.html

[7] コンピュータの遠隔操作システム　VNC：http://www.uk.research.att.com/vnc/

松田七美男（まつだ　なみお）

研究分野：全般には，薄膜物性，現在のテーマは，「固体表面の二次電子放出係数の薄膜による制御」

経歴：1956　秋田県生まれ．1985　東京大学工学研究系博士課程満期退学，1985　高エネルギー物理学研究所助手，1989　東京電機大学工学部講師，2001　現在，東京電機大学工学部教授

趣味：Linuxで遊ぶこと．**家族：**妻1人，娘1人，犬1匹，**近況：**飼っているミシェルというダックスフントが，犬らしからぬ寒がり（朝，散歩したがらない）であることが判明した，**自己分析：**単なる食いしん坊の無神経なデブに見えるらしいが，本人はいたって繊細な神経の持ち主だと思っている．