2012 年 07 月 27 日

計算機実習 IV (2012 年度)
第 14 回: gnuplot その 4

(http://takeno.iee.niit.ac.jp/%7Eshige/math/lecture/comp4/comp4.html)

目 次

1 画像ファイルなどへの出力

対話形式の wgnuplot で作成したグラフを画像ファイルとして残したり、ワープロソフトなどに取り込んで利用するには、以下のような方法がある。

1. クリップボード経由でコピー & ペースト (貼り付け) する
2. wgnuplot の機能を利用して EMF 形式で保存する
3. `set terminal` コマンドを利用して任意の出力形式として出力する

EMF 形式 (Enhanced Meta File 形式、拡張子 `.emf`)[1] は、MS–Windows で標準的な画像形式 (ベクトル画像形式) の一つで、この形式の画像は MS–Windows のワープロソフト、表計算ソフトなどに容易に取り込むことができる。EMF はベクトル画像形式なので、一般的には取り込んだ画像を拡大、縮小しても綺麗に表示される。

1. のクリップボード経由での画像の保存は、以下のいくつかの方法がある。

[1]「Enhanced」は「拡張された」という意味であり、WMF (Windows メタファイル) と呼ばれる形式を拡張したもの。

- グラフウィンドウ上で `Ctrl-C` とする
- グラフウィンドウ上部のクリップボードアイコン (Copy graph to clipboard) をクリックする
- グラフウィンドウ上部の「Options」アイコンをクリックするか、グラフウィンドウのタイトルバーを右クリックして、「Copy to Clipboard」を選択する

課題 14-1. 日本語のタイトルをつけた $\sin x$ のグラフを描き、それをクリップボードにコピーし、ペイント (「すべてのプログラム」 $\Longrightarrow$ 「アクセサリ」 $\Longrightarrow$ 「ペイント」、またはコマンドプロンプトで `mspaint` を実行) 内に貼り付け、それをビットマップファイル `test1.bmp` として保存せよ。

課題 14-2. 日本語のタイトルをつけた $\sin x$ のグラフを描き、それをクリップボードにコピーし、ワープロソフト上でペースト (貼り付け) を行え。

2. の EMF 形式で保存するには、以下のいくつかの方法がある。

- グラフウィンドウ上で `Ctrl-S` とする
- グラフウィンドウ上部の保存アイコン (Save graph as EMF) をクリックする
- グラフウィンドウ上部の「Options」アイコンをクリックするか、グラフウィンドウのタイトルバーを右クリックして、「Save as EMF...」を選択する

課題 14-3. 日本語のタイトルをつけた $\sin x$ のグラフを描き、それを EMF ファイル `test1.emf` として保存し、それをペイントで開け。

課題 14-4. `test1.emf` をワープロソフト上で取り込んでみよ。

3. の `set terminal` (省略形 `set term`) による出力形式の変更は gnuplot の標準的な方法で、EMF 形式以外にも多くの種類の形式のファイルを直接出力できる。`set term` は、`set output` (省略形 `set out`) と組で使うことが多い。

- `set term` <出力形式名> {<オプション>}　　: 出力形式の切り替え

- `set output <出力ファイル名>` : 出力ファイルの指定
- `set output` : 出力ファイルをクローズする

主な出力形式には、例えば表 1 のようなものがある。なお、gnuplot のヘルプ

`win`	Windows グラフ (default)	`windows`	`win` に同じ
`emf`	EMF 画像形式	`cgm`	CGM 画像形式
`corel`	CorelDraw 用	`dxf`	AutoCad 用
`mif`	Adobe FrameMaker 用	`gif`	GIF 画像形式
`jpeg`	JPEG 画像形式	`png`	PNG 画像形式
`svg`	SVG 画像形式	`canvas`	HTML5 canvas 要素
`dump`	テキスト文字グラフ	`post`	PostScript 形式

表 1: 主な出力形式一覧

ではさらに多くの出力形式の一覧が表示されるが、そのすべてが使用できるとは限らない。使用できる出力形式の正しい一覧は、対話ウィンドウで「`set term`」を実行すると表示される。

`set term` で指定できるオプションは、各出力形式毎に異なるが (詳しくは、各出力形式のヘルプを参照)、以下のようなオプションが使えるものが多い。

- `mono/color` : 白黒モード/カラーモード
- `solid/dashed` : 線種はすべて実線/線種で点線・破線を使用
- `size <x>,<y>` : 出力画像サイズを指定
- `enhanced/noenhanced` : 拡張文字列処理モードを On に/Off に

例えば、以下のようにすると GIF 画像ファイル file.gif が生成される。

```
set term gif size 400,300 # 出力形式指定
set out "file.gif" # 出力ファイル指定
set title "テスト" font "msmincho.ttc,17"
plot sin(x)
set out
set term win
```

GIF/JPEG/PNG 画像形式では、フォントは TrueType フォント (デフォルトでは `C:¥Windows¥Fonts` にあるもの) を使用するので、直接その TrueType フォントファイル名を指定する必要がある。(msmincho.ttc は「ＭＳ 明朝」フォントの TrueType フォントファイル名)。

`set out <ファイル名>` は、`set term` の直後に指定し、plot でグラフを出力した後に `set out` で画像ファイルをクローズする必要がある。最後の `set term win` で対話型出力形式に戻る。

`set term` を使えば、2. のような対話型ウィンドウの機能を使わなくても EMF 画像を作成することができるので、処理をスクリプト化できるし、フォントも容易に変更できるメリットがある。

```
set term emf size 400,300
set out "file.emf"
set title "テスト" font "MS Mincho,17"
plot sin(x)
set out
```

課題 14-5. `set term` を利用して、$\sin x$ と $\cos x$ のグラフを重ね描きした JPEG 画像ファイル kadai14-5.jpg を出力する gnuplot スクリプト kadai14-5.gp を作成せよ。なお、タイトルと y 軸のラベルに日本語を使用すること。

課題 14-6. 日本語のタイトルをつけた $\sin x$ のグラフを描き、3. の方法 (`set term emf` を利用) で EMF ファイルにしたものを `test2.emf` として保存し、`test1.emf`, `test2.emf` の両者をワープロソフトに取り込んで比較せよ。

課題 14-7. 1 列目が年令、2 列目が年収 (架空でよい) である CSV 形式のデータファイルを表計算ソフトで作成し、それを gnuplot で EMF 画像グラフに直せ。なお、CSV 形式 (コンマ区切り) の データを gnuplot で扱う場合は、「`set datafile separator ","`」とすればよい。

`set term win` で通常のグラフウィンドウに関する設定 (サイズや白黒、背景色、実線/点線など) も設定できるが、これは特に複数のグラフウィンドウを同時に立ち上げたい場合によく用いられる。`set term win` の最初のオプションに番号を指定することで、複数のグラフウィンドウを表示できる。例:

```
set term win 1 size 640,480
plot sin(x)
set term win 2 mono size 480,360
plot cos(x) w l lt 3 lw 2
```

課題 14-8. `set terminal windows` と `set terminal emf` のヘルプを読んでみよ。

課題 14-9. 大きさの違う 3 つのグラフウィンドウにそれぞれ $\sin x$, $\cos x$, $\tan x$ のグラフを描いてみよ。

課題 14-10. `set term win` に `mono`/`color`, `solid`/`dashed` のオプションをつけてみて (4 通り)、そのそれぞれで `test` コマンドの出力にどのような違いがあるか確認せよ (同時に立ち上げないと違いはわかりにくい)。

2 種々の設定 (set) その 3

set key

通常グラフウィンドウの右上に表示されるグラフの凡例 (key) の位置や並べ方などは、`set key` で設定できる。`set key` には多くのオプションがあるが、主なものは以下の通り (指定順は任意)。

オプション	意味
`off/on`	凡例の表示をやめる/表示する
`outside/inside`	凡例をグラフ領域の外に描く/内側に描く
`horizontal/vertical`	水平に並べる/垂直に並べる
`left/center/right`	凡例を左に/中央に/右に
`bottom/center/top`	凡例を下に/中央に/上に
`opaque/noopaque`	凡例範囲を透明化/凡例範囲を不透明化
`reverse/noreverse`	例、タイトルの順に/タイトル、例の順に
`box/nobox`	凡例の外枠を描く/描かない

いずれも最後のものがデフォルトになっている。例えば、

```
set key outside left bottom reverse box
```

とすれば、凡例をグラフ領域の外側、左下に外枠付きで描く。凡例は、線の例、タイトルの順に各グラフの説明を描く。

課題 14-11. グラフを複数重ね描きして、`set key` の上のオプションをすべて試して、デフォルトとどう変わるかを確認せよ。

課題 14-12. `set key opaque box` と `set key box` の違いを、`set grid` などと併用することで確認せよ。

3 システム関数の呼び出し

gnuplot 内部から外部プログラムを呼び出す仕組みが用意されている[2]。

- `!` : ! で始まる行は、コマンドプロンプト命令として実行される

この ! ではリダイレクションやパイプも書くことができる。例:

```
!  gawk -f file1.awk file1.dat > file2.dat
plot "file2.dat" w lp
pause -1
```

これは、gnuplot スクリプト内でデータファイルを作成/加工してそれを `plot` したり、別の gnuplot スクリプトを作成してそれを `load` したりするのに使える。なお、! 行で gawk の 1 行スクリプトなどを書く場合は、引用符のエスケープに注意が必要である。

```
!  gawk "BEGIN{ printf ¥"plot sin(x) not ¥¥¥n¥"; ¥
    for(j=1;j<=10;j++) ¥
        printf ¥",sin(x+%f) not ¥¥¥n¥",j/3; print}" ¥
    > file2.gp
load "file2.gp"
pause mouse
```

`¥` が 3 つ重なっているように見えるのは、最初の 2 つが引用符内での 1 つの `¥` を意味し、後ろの 1 つはその後ろの n とつながって改行を意味していて、実際には file2.gp の行末に `¥` を 1 つ置くことになっている。各行末の `¥` は ! 行自体が続いていることを示している。

課題 14-13. 上の例の gnuplot スクリプトを作成して、実際に実行し、`file2.gp` に実際に書き出される内容を確認せよ。

課題 14-14. 以下を行う gnuplot スクリプト kadai14-14.gp を作成せよ:

- $f(x) = e^{-x}\sin x$ を定義する

[2]現在は ! 以外にシステム関数 `system()` も利用できるが、説明は省略する。

- gnuplot スクリプトの ！ 行で、gawk を用いて $f(nx)$ のグラフ $(n = 1, 2, \ldots, 10)$ を重ね描きするような plot コマンドを tmp1.gp に書き出す
- そしてそれを `load` する

なお、凡例 (`set key`) はグラフの外に出し、凡例の各グラフのタイトルは「$n = 1$」「$n = 2$」のように書くようにせよ。

課題 14-15. 以下を行う gnuplot スクリプト kadai14-15.gp を作成せよ:

- $f_1(x) = 1$ と定義する
- gnuplot スクリプトの ！ 行で gawk を用いて、漸化式 $f_{n+1}(x) = f_n(x) + x^n/n!$ の $n = 1, 2, \ldots, 10$ に対する定義式を tmp2.gp に書き出す
- それを `load` して、その後で $y = e^x$ と $y = f_{11}(x)$ のグラフを重ね描きする

なお、$f_5(x)$ などの関数名は、スクリプト内では `f5(x)` のようにせよ。

4 インラインデータ

データ描画するデータは、通常 gnuplot スクリプトとは別に用意しそれを plot コマンドに読ませるが、データをスクリプト内、あるいは対話ウィンドウで直接与えることもできる。それを **インラインデータ形式** と呼ぶ。インラインデータ形式は、データファイル名として `"-"` を指定し、インラインデータの終わりは `e` を行頭に指定する。

対話ウィンドウで書く場合には例えば以下のようになる。

```
gnuplot> plot "-" w l
> 1 5
> 2 4
> e
gnuplot>
```

スクリプトでは以下のように書く。

```
plot "-" w l
1 5
2 4
e
pause -1
```

複数のデータ描画などの重ね描きも可能であるが、インラインデータ同士を重ね描きする場合は、e の後に次のインラインデータを続ける必要がある。これは同じデータであっても省略はできない。

```
plot "-" w l, "-" w i
1 5
2 4
e
1 5
2 4
e
pause -1
```

インラインデータは、gnuplot 命令とデータが同一ファイルに含まれるので、外部プログラムから gnuplot を呼び出す場合に重宝する。

課題 14-16. `plot "-" w l` で、1 列目が 2 から 20 までの整数、2 列目が 1 列目の数の約数の個数というグラフを描く gnuplot スクリプト kadai14-16.gp を作成せよ。

5 外部から変数を指定

gnuplot スクリプトを作成し、gnuplot (wgnuplot) をバッチファイルやコマンドプロンプトから呼び出す場合、gnuplot スクリプト内部の変数を変更して実行するには -e オプションを利用すればよい。

- `-e "<gnuplot 命令>"` : 指定した <gnuplot 命令> をその順に実行

これは、いわば gnuplot の「1 行スクリプト」であるが、スクリプトファイル名と `-e "<gnuplot 命令>"` を並べて指定した場合は、並べた順に実行される。

例えば、以下のような gnuplot スクリプト test1.gp に対して

```
ts = sprintf("総人口と未成年人口の推移 (%s 県)", s1)
set term emf size 400,300
set out f2
set title ts font "MS Mincho,20"
set xlabel "年" ; set ylabel "人数"
plot f1 u 1:2 t "総人口" w boxes lt 1 fs s 1, ¥
    "" u 1:3 t "未成年人口" w lp lt 3
set out
```

文字列 s1 (県名)、出力ファイル名 s2 の内部変数を、

```
Z:> gnuplot -e "s1=¥"新潟¥"; f1=¥"niigata.dat¥"
    f2=¥"niigata.emf¥"" test1.gp
    (↑実際には 1 行で実行する)
```

のように実行して外から与えることができる。

課題 14-17. 内部変数 x1 と x2 を xrange の最小値、最大値として、データファイル data の with points のグラフを描く gnuplot スクリプト kadai14-17.gp を作成せよ。x1, x2 は -e で外から与えられるようにし、実際にうまく動くかも確認せよ。

課題 14-18. 2 つの適当な gnuplot スクリプト (pause なし) を wgnuplot に指定して実行してみよ。次に 2 つのスクリプト指定の間に -e "pause mouse" をはさんで実行して動作を確認せよ。

コラム: 他のプログラムからの gnuplot の利用

gnuplot 内部から外部プログラムを呼び出すこともできるが、逆に C や AWK などの他のプログラムから gnuplot を呼び出してグラフを描かせることもできる。それには、以下のような 2 つの方法がある。

1. そのプログラムから gnuplot スクリプトとグラフデータを出力し、その言語に用意されているシェルコマンド実行命令 (C 言語なら system() 関数) で gnuplot (wgnuplot) を実行する
2. そのプログラムから、gnuplot へのパイプを開いて、直接そのパイプへ gnuplot 命令と描画データを流す

1. は容易だろうから、2. について紹介する。

MS–Windows 上で C のプログラム (VC++, Borland C++) から呼び出す場合、`fopen()`, `fclose()` に対応するパイプ関数 `_popen()`, `_pclose()` を使うことで、パイプコマンドとして gnuplot を呼び出せる。C 言語のサンプルや詳細は省略するが、以下の点に注意が必要。

- 標準入力はパイプ入力になり `pause -1` へはデータを送れなくなるので、代わりに `pause mouse` を使用する。
- `gnuplot.exe` を使用する (`wgnuplot.exe` はパイプ入力不可)。

gawk でパイプとして gnuplot を立ち上げるには、ファイルへのリダイレクトの場合と同様、printf 行, print 行の最後に「`| "gnuplot"`」を追加すればよい。gawk でのサンプルを一つ紹介する。

```
BEGIN {
    pipe = "gnuplot"
    print "plot sin(x)" | pipe
    print "pause mouse" | pipe
    fflush(pipe) # 出力を実際にパイプに吐き出す
    print "plot cos(x) w l lt 2" | pipe
    print "pause mouse" | pipe
    fflush(pipe)
    print "plot ¥"-¥" w lp lt 3" | pipe
    for (j=1; j<=10; j++)
        printf "%f %f¥n", j, j*j | pipe
    print "e" | pipe
    print "pause mouse" | pipe
    fflush(pipe)
    close(pipe)
}
```