設計計算機演習 １－３ Fortran による図形描画 その１ (2002/10/18)

1 Calcomp/GKS ライブラリについて

標準の Fortran には作図のための機能は含まれないが，非標準で様々なライブラリが作成されている．ここでは，比較的よく使われる Calcomp/GKS の命令体系に基づくライブラリを用いて作図を行う．用いるライブラリは，森正武，Fortran77 図形処理プログラミング（岩波書店）によるものである．

ライブラリ関連ファイルは `/usr/freeware/bin`，`/usr/freeware/lib` 以下にある．サブルーチンの細かな説明は `/usr/freeware/doc/GKS/gks.doc` を参照すること．

2 サンプルプログラム・その 1（描画ルーチンの例）

2.1 サンプルプログラム `sample.f` の説明

```
c23456
      program sample
      real x(10), y(10), x2(2), y2(2), x4(4), y4(4), xx
      data x/1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0/
      data y/5.0,3.0,8.0,2.0,6.0,4.0,3.0,6.0,8.0,7.0/
      data x4/0.5,11.5,11.5, 0.5/
      data y4/3.5, 3.5, 6.5, 6.5/
      integer j

      call plots

* マーカを描く
      call gsmksc(0.8)     ! マーカの高さ 0.8cm
      do j=1, 8
         xx = float( j )
         call gsmk(j)        ! マーカの種類を指定（1〜8 まで）
         call gpm(1,xx,15.0) ! (1.0,15.0) から (8.0,15.0) までマーカを描いていく
      enddo

* 斜線を描く
      do j=1, 4
         xx = float(j)
         call gsln(j)       ! 線種は 1〜4 まで
         call gslwsc(xx)    ! 線太さは 1.0 〜4.0 まで
         x2(1) = xx+1.0     ! 右にずらしながら...
         y2(1) = 10.0
         x2(2) = xx+5.0
         y2(2) = 13.0
         call gpl(2,x2,y2) ! gpl を用いて線を引いていく
      enddo

* 文字を書く（書ける文字はアルファベット，数字，記号のみ）
      call gsln(1)          ! 実線に戻す
      call gslwsc(1.0)      ! 細線に戻す
      call gschh(1.0)       ! 字の高さ 1cm
      call gtxs(8.0,10.0,5,'ABCde') ! (8.0, 10.0) に ABCde の 5 文字を描く

* 四角形（多角形）を描く
```

```
      call gsln(4)          ! 破線
      call gslwsc(2.0)      ! 線太さ 2
      call gsfais(1)        ! 多角形を描くときに、中を塗りつぶさず周囲の線を描く
      call gfa(4,x4,y4)     ! 多角形を描く

* 一次元配列で一度に描く
      call gsln(1)          ! 実線に戻す
      call gslwsc(1.0)      ! 細線に戻す
      call gsmk(1)          ! マーカの種類 1
      call gsmksc(0.5)      ! マーカの高さ 0.8cm
      call gpm(10,x,y)      ! 配列に代入された x, y 座標を用いて、一度にマーカを描く。

* クリッピングして描く
      call clipon(0.5,11.5,3.5,6.5) ! この領域に囲まれた範囲が描画可能になる（クリッピング）
      call gpl(10,x,y)      ! 配列に代入された x, y 座標を用いて、一度に線を描く。
      call clipof           ! クリッピング終了

*
      call plotv

      stop
      end
```

2.2 コンパイルと実行，画面表示の手順

```
hmogi@onyx3400[1]:ls
Gnuplot/    sample.f    sankaku*    sankaku.f   wave.out
hmogi@onyx3400[2]:vi /usr/freeware/doc/GKS/gks.doc      ← GKS ルーチンの説明を見る
hmogi@onyx3400[3]:f77 sample.f -old_rl -lgks -o sample ← f77 でコンパイルする場合（長い）
hmogi@onyx3400[4]:lf77 sample.f -o sample   ←lf77 でコンパイルする（前回 .cshrc 設定の参照）
hmogi@onyx3400[5]:ls                        ←実行ファイル sample が作成された
Gnuplot/    sample*     sample.f    sankaku*    sankaku.f   wave.out
hmogi@onyx3400[6]:sample                    ←作成した図化プログラムを実行する
 Key in IDSP (0 or 1 or -1).
   0 ... display and save
   1 ... display only
  -1 ... save only

0                                           ← 0 or -1 で図形データをセーブ（保存）する
 (PDFILE) NPLT =    405
 Do you save plot data ?  Key in Y or N. [N]
y                                           ← y=yes で GKS 出力をセーブ（保存）する.
 Key in output file name.
sample.out                                  ← GKS 出力を保存するときのファイル名.
 Key in 0 or 1 for plot data mode.
  0 ... numeral
  1 ... compact
0                                     ← 0（圧縮形式）or 1（圧縮形式），どちらでも良い.
 (PDSAVE) number of original plot data =   405
hmogi@onyx3400[7]:ls                  ← GKS 出力ファイル (sample.out) が作成されている
Gnuplot/     sample.f     sankaku*     wave.out
sample*      sample.out   sankaku.f
hmogi@onyx3400[8]:ntop sample.out sample.eps  ← GKS 出力ファイル (sample.out) を PostScript
                                                ファイルに変換する
 (PDLOAD) number of plot data =   405  (Data is in numeral mode)
hmogi@onyx3400[9]:gs sample.eps  ← gs(shostscript) で画像 (PostScript ファイル) を表示する.
```

```
Aladdin Ghostscript 3.33 (4/10/1995)
Copyright (C) 1995 Aladdin Enterprises, Menlo Park, CA.  All rights reserved.
This software comes with NO WARRANTY: see the file COPYING for details.
>>showpage, press <return> to continue<< ← 図のように画像 (PostScript ファイル) が表示される
```

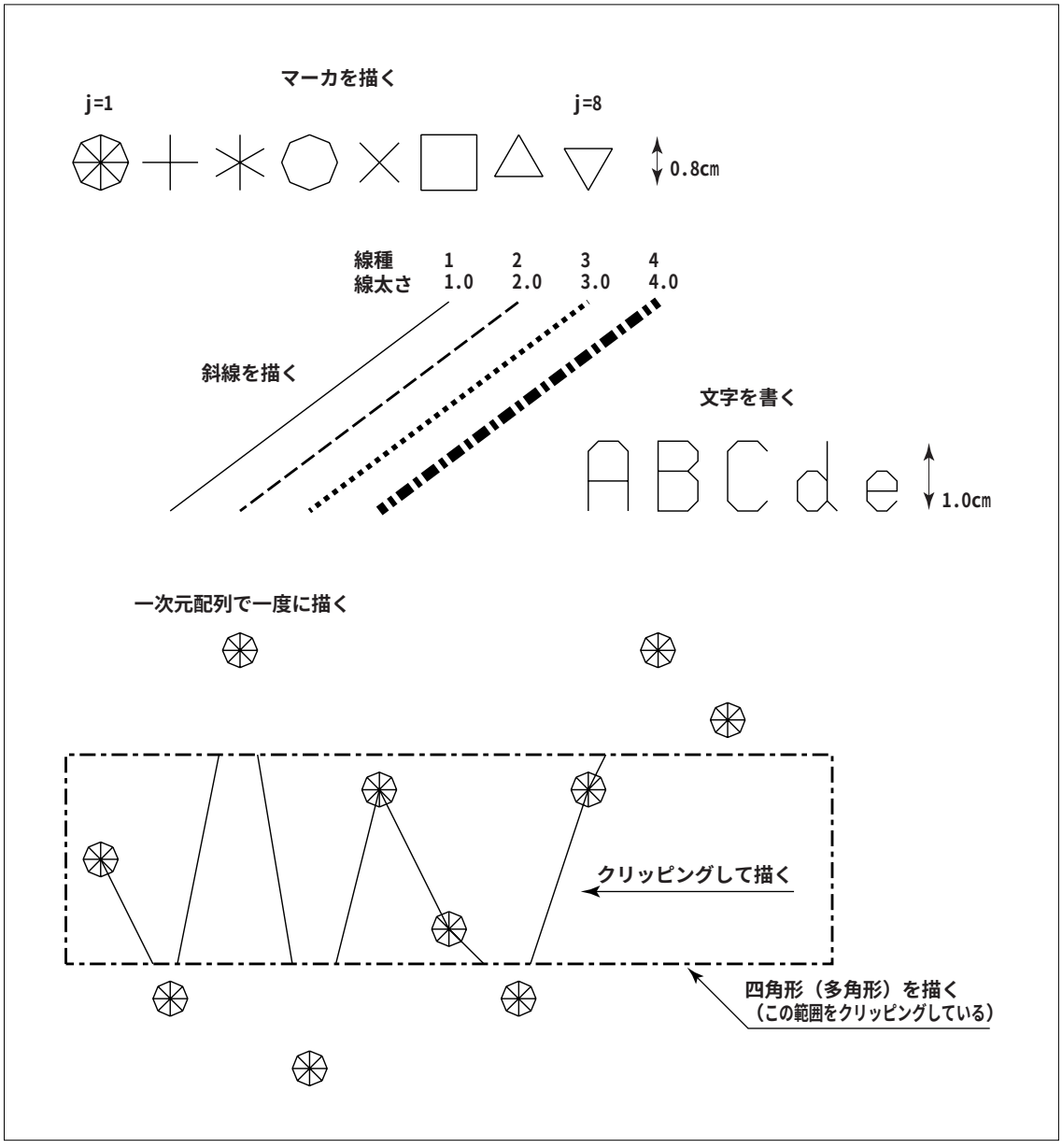

Fig. `sample.f` の出力結果

3 グラフを描く

3.1 座標・座標変換の考え方

グラフィックライブラリの座標系が紙の上の cm 単位の値に固定されているのに対して、実際に図化する観測値や計算値はさまざまな値や単位をとるため、描く対象に応じて紙の上の座標に割り当てることになる。

ここでは便宜的に、紙の上の座標を紙座標 (x_p, y_p)、図化する値の座標をユーザー座標 (x, y) と呼ぶ。

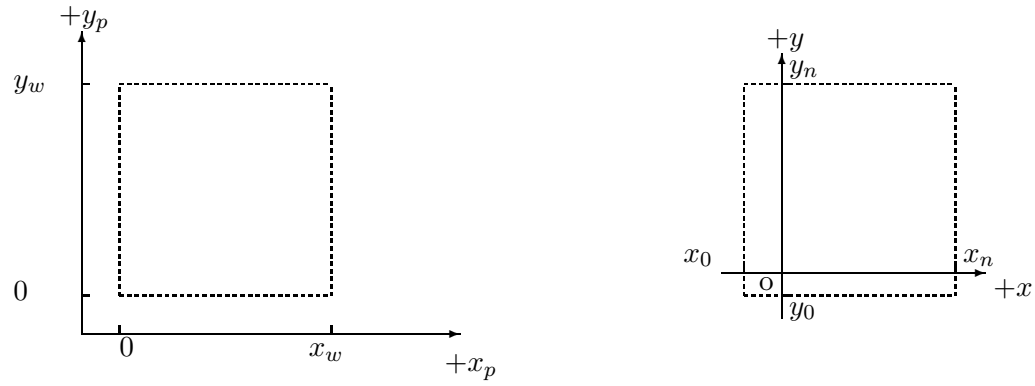

Fig. 紙座標 (左) とユーザー座標 (右)

図化プログラムの中では、観測値を紙座標値に変換しながら描画して行く。普通のグラフの場合、ユーザー座標から紙座標への変換は単純に比例配分すれば良いから、

$$x_p = x_w \frac{x - x_0}{x_n - x_0}, \quad y_p = y_w \frac{y - y_0}{y_n - y_0} \tag{1}$$

を変換関数として定義しておく。

同様に対数グラフの場合、

$$x_p = x_w \frac{\log x - \log x_0}{\log x_n - \log x_0}, \quad y_p = y_w \frac{\log y - \log y_0}{\log y_n - \log y_0} \tag{2}$$

を変換関数とすればよい。次節のサンプルプログラム `graph.f` では、これらの変換関数を関数副プログラム `wtox(x,x0,xn,xw)`, `wtoy(y,y0,yn,yw)` を定義している。

3.2 座標軸を描くプログラム `graph.f`

```
      program graph
      real wtox, wtoy
      real x0, y0, xn, yn, xw, yw, hgt
      integer mx, my

      data x0/-10.0/, y0/-1.0/, xn/10.0/, yn/1.0/   ! グラフにする観測値の範囲
      data xw/10.0/,  yw/10.0/                      ! 紙の上の長さ
      data mx/5/,     my/4/                         ! 座標軸の目盛りの本数

* 描画開始と原点移動
      call plots                ! 描画開始
      call plot (3.0, 3.0, -3) ! 紙座標原点を紙の左下隅から (3cm,3cm) のところに移動する

* 座標軸の線を引く
      call axisx(xw,yw,mx*2,0,0) ! x 座標軸を描き, mx*2 個の目盛りで等分する
      call axisy(yw,xw,my*2,0,0) ! y 座標軸を描き, mx*2 個の目盛りで等分する

* 座標軸に数字をふる
      hgt = 0.25                                    ! 座標軸の文字の高さ
      call axinx(x0, xn, xw, mx, hgt, '(F5.1)') ! 書式 F5.1 で x 座標軸に
                                                    数字をふる       !! F5.1 は大文字
      call axiny(y0, yn, yw, my, hgt, '(F4.1)') ! y 座標軸に数字をふる
```

```
c
c               ここに観測値を描画させるプログラムを加える
* おわり
      call plotv

      stop
      end

      real function wtox(x,x0,xn,xw) ! 紙座標への変換関数 (x)
      real x, x0, xn, xw
      wtox=xw/(xn-x0)*(x-x0)
      return
      end

      real function wtoy(y,y0,yn,yw) ! 紙座標への変換関数 (y)
      real y, y0, yn, yw
      wtoy=yw/(yn-y0)*(y-y0)
      return
      end
```

4 PostScript ファイルの加筆・修正

ベクトル形式の画像ファイルの編集ツールの一つである sketch（フリーソフト，UNIX 上でのみ利用可能）を用いて加筆，編集を行う．sketch では PostScript 形式の出力が可能であるが，読み込みはできない．このため，sketch で読み込む前に pstoedit を用いて，PostScript 形式から sketch の専用の画像形式に変換しておく必要がある（以下の実行例を参照）．また，sketch で PostScript 形式で保存した場合でも，再度編集することを考えて，sketch 形式でも保存しておいた方がよい．作成した画像を印刷するには，セーブした PostScript 形式のファイルをプリンタに送ればよい．なお，現在 sketch では日本語が使えない．

```
hmogi@onyx3400[9]:ls
Gnuplot/     graph.eps    graph.out    sample.eps   sample.out   sankaku.f
graph*       graph.f      sample*      sample.f     sankaku*     wave.out
hmogi@onyx3400[10]:pstoedit -f sk graph.eps graph.sk  ← PostScript ファイル (graph.eps)
                                                        から sketch 形式のファイル
                                                        (graph.sk) へ変換
pstoedit: version 3.15 / DLL interface 105 (build Oct 13 2002) :
 (続き) Copyright (C) 1993 - 1999 Wolfgang Glunz
Interpreter finished. Return status 0
Warning: no %d found in name of output file and page feed found in input and
 (続き) either the selected format does not support multiple pages
 (続き) or the -split option was specified.
Please insert a %d in the name of the input file if you want to split pages
 (続き) into different files
hmogi@onyx3400[11]:sketch graph.sk                  ← sketch を起動する
shared memory images not supported
```

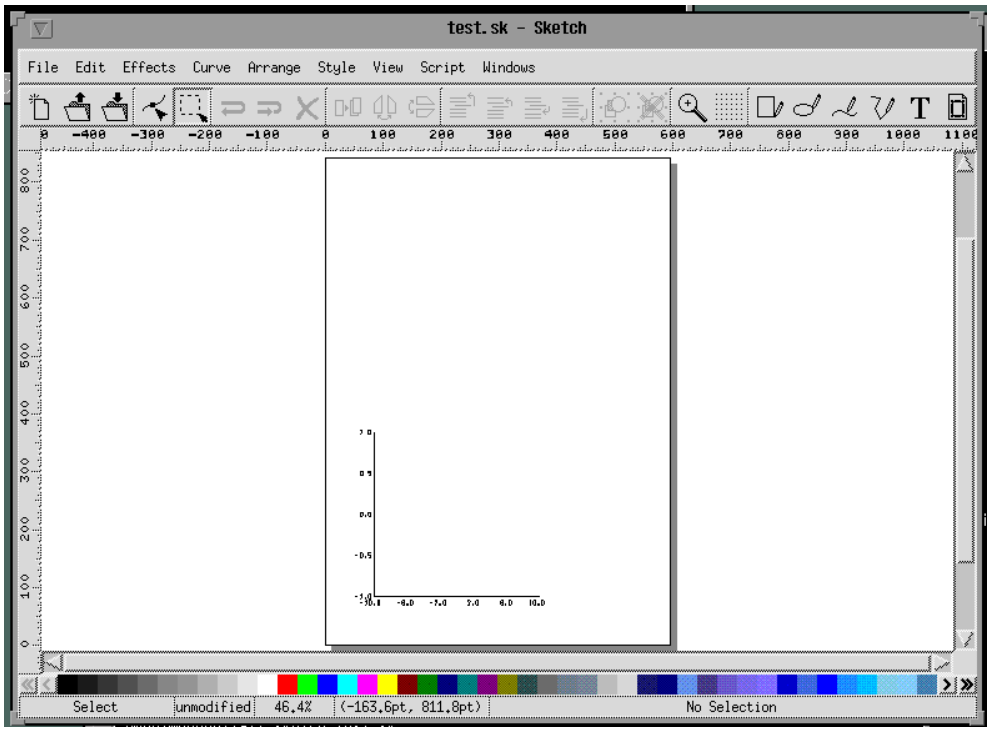

Fig. `sketch` で `graph.f` の出力結果を読み込んだところ

宿題　次式を x-y 座標上に図化し，図化プログラムと出力結果を提出せよ．

$$\left\{ \begin{array}{l} x = r\cos\theta \\ y = r\sin\theta \\ r = \sin n\theta \end{array} \right. \quad , \qquad \text{ただし，} 0 \leq \theta \leq 2\pi, \quad n \text{ は } 1 \leq n \leq 8 \text{ から一つ決める}$$

提出　出力結果はメールボックスに提出，作成したプログラムはメールでおくる。例えば、作成したプログラムが `gr.f` なら　`mail q3@kiban.civil.saitama-u.ac.jp < gr.f`　とする（宛先注意）．電子メールの着信確認は前回と同様．