情報処理 II No.3

グラフの描き方 (2)

桂田 祐史

1995 年 5 月 12 日

1 FORTRAN 向け図形描画ライブラリィ fplot の解説

1.1 はじめに – f77x コマンド、 fplot ライブラリィ

前回は mgraph コマンドによる図形描画の方法を紹介しましたが、今回はそれとは異なるもう一つの手段を紹介します。 FORTRAN プログラム[1]から利用できる、図形描画用のサブルーチン・ライブラリィ **fplot**[2]を用意しました。それを利用するにはプログラムを `f77o` や `f77` コマンドの代わりに **f77x** コマンドを用いてコンパイルします[3]。

例えば “`reidai3-1.f`” というプログラムをコンパイルするには

```
f77x reidai3-1.f
```

とします。エラーがなければ “`reidai3-1`” という名前の実行形式のプログラムが出来ます。

二つの方法 (mgraph,fplot) を紹介した一番大きな理由は、この講義で必要になるごく単純な図形描画に限っても、万能のものがないということです。両者の長所・短所を理解して使いこなせるようになって下さい。

1.2 例題による解説

まず例をあげて解説することにしましょう。

例題 3-1: $y = \sin x + \sin 3x + \sin 5x$ のグラフを描きなさい。

これは既に `mgraph` で解いた例題 2-1 ですが、次の “`reidai3-1.f`” では fplot で解きます。

```
* reidai3-1.f -- サブルーチン呼び出しによるグラフ描き
      program reidai31
*       定数の宣言
*         a: 区間の左端の座標, b: 区間の右端の座標, margin: 余白の大きさ
      real Pi,a,b,margin
      parameter (Pi = 3.14159265)
      parameter (a = 0.0, b = 2.0 * Pi)
      parameter (margin = (b - a) / 20)
```

[1]実は C プログラムからも利用できます。その場合は **ccx** コマンドでコンパイルして下さい。

[2]この名前の頭文字は、最初にプログラムを書いた藤尾秀洋君の姓、座標のデータ型 (floating point numbers) という 2 つの “f” にかけてあります。

[3]昨年度の「情報処理・演習」では **f77g** コマンドが紹介されたようですが、 `f77x` はその兄弟分で情報処理 II 専用のコマンドです。

```
*       変数の宣言
      integer N,i
      real h,x,f
*       x 軸の分割の仕方の決定
      write(*,*) ’ x 軸の区間の分割数を入力して下さい’
      read(*,*) N
      h = (b - a) / N
*       グラフィックスの初期化
      call openpl
      call fspace2(a - margin, -3.0, b + margin, 3.0)
*       始点のセット
      call fmove(a, f(a))
*       グラフ上の点を順に結ぶ
      do i = 1, N
        x = a + i * h
        call fcont(x, f(x))
      end do
*       図形を記録する
      call mkplot(’rei31.plot’)
*       グラフィックスの後始末
      call closepl
      end
************************************************
      real function f(x)
      real x
      f = sin(x) + sin(3.0 * x) + sin(5.0 * x)
      end
```

全体が、主プログラム “reidai31” と関数副プログラム “f” の 2 つの部分からなっていることが見て取れます。後者は “reidai2-1.f” のそれと全く同じものです。主プログラムの中で call 文で呼び出している **openpl**, **fspace2**, **fmove**, **fcont**, **closepl** が描画用のサブルーチンです。

openpl は他のすべてのグラフィックス命令に先立って call します。これによって図を描くためのウィンドウがオープンされます。

fspace2 はウィンドウに座標を割り振ります。

```
call fspace2(x0,y0,x1,y1)
```

とするとウィンドウの左下隅が (x0,y0)、右上隅が (x1,y1) になります（当然 x0<x1, y0<y1 です）。扱う問題に応じて適当な値を見い出して call しなければなりません。次の fmove,fcont は後回しにして、最後の closepl を先に解説します。これは openpl と対になる命令で、図形描画を終了させて後始末をするものです。実行がここに到達すると、プログラムは利用者がマウスでウィンドウをクリックするのを待つ状態になります。

ここまで解説した openpl,fspace2,closepl は実際には何も描きませんが、何時でも必ず必要になります。つまりプログラムは必ず

```
.........
call openpl
.........
call fspace2(x0,y0,x1,y1)
.........
実際の描画命令の列
```

```
.........
call closepl
.........
```

という形になります。

fmove は「ペンを移動する (=move)」, fcont は「線を引きながら (つなぎながら =continue) ペンを動かす」という命令です。つまり、これらの命令では仮想のペンを考えて、それを移動することにより図形を描画します[4]。では `f77x` でコンパイルしてから実行してみましょう。

```
reidai3-1
x 軸の区間の分割数を入力して下さい     ←これはプログラムからのメッセージ
100                                    ←入力
```

問題 3-1: mgraph を利用した際 (reidai2-1) は、軸や目盛が表示された。"`reidai3-1.f`" を修正して、それらを描くようにしなさい。軸を点線で描くなど工夫してみなさい。(余裕があったら、関数を変更した場合に、修正が少なくてすむような書き方を考えてみよう。)

1.3 fplot で利用できる描画用サブルーチン一覧

fplot には以下のサブルーチンが含まれています。以下の引数 (x0,y0,r 等) は文字列である s を除いて単精度実数型 (real) です。

openpl plot ライブラリィの初期化をする。

closepl plot ライブラリィの後始末をする。

erase 画面をクリアする。

fspace(x0, y0, x1, y1) 左下端が (x0,y0), 右上端が (x1,y1) である長方形が描けるように座標を割り当てる。この際、縦横の拡大比が等しくなるような調節が行なわれる。 x0<x1, y0<y1 でなければならない。

fspace2(x0, y0, x1, y1) スクリーンの左下端を (x0,y0), 右上端を (x0,y0) とするように座標を割り当てる (fspace のような調節は行なわない)。これを利用した場合は `fcircle` や `farc` は使えない。 x0<x1, y0<y1 でなければならない。

label(s) 現在点に文字列を表示する。 s は `'Hungry?'`, `"Cup noodle!"` などの文字列（残念ながら日本語は使えません）。

linemod(s) 線分のパターンを指定する。 s として指定できるのは `'dotted'`, `'solid'`, `'longdashed'`, `'shortdashed'` と`'dotdashed'` である[5]。

fline(x0, y0, x1, y1) 点 (x0,y0) から点 (x1,y1) までの線分を描く。現在点は (x1,y1) となる。

fcircle(x, y, r) 点 (x,y) を中心とする半径 r の円を描く。現在点は (x,y) となる。

---

[4]実は XY プロッターという、そのものズバリ、ペンを紙の上で動かして図形を描く装置があって、fplot ライブラリィはその動作原理をモデルにして作られたものです。これは UNIX に昔からある plot ライブラリィ関数がお手本になっています。

[5]今のところ solid(**実線**) 以外は全部ただの点線になってしまいます。そのうちきちんと描き分けるようにします。

farc(x, y, x0, y0, x1, y1) 中心が (x,y) の円弧（始点 (x0,y0), 終点 (x1,y1)）を描く。描画は反時計回りに行なわれる。

fmove(x, y) 現在点を (x,y) に変更する。

fcont(x, y) 現在点から点 (x,y) まで線分を描く。現在点は (x,y) になる。

fpoint(x, y) 点 (x,y) に点を描き、そこを現在点とする。

座標を指定するのに倍精度実数型（double precision あるいは real∗8 と宣言する）を使うことも出来るように、先頭の文字が 'd' のサブルーチン（dspace, dline, dcircle, darc, dmove, dcont, dpoint）も用意してあります。使い方は、先頭の文字が 'f' であるものに準じます。

以下にあげるサブルーチンは、少し特殊なものなので、とりあえず無視してしまっても構いません (最初の二つのうちのどちらかは、お世話になるかも知れません)。

xflush それまでたまっている描画命令をはき出す (X のリクエスト・バッファーをフラッシュする)。アニメーションで切りの良いところで実行すると動きが滑らかになる。

xsync それまでたまっていた描画命令をはき出し、完全に実行し終わる（実際にウィンドウに図形が表示される）まで待つ。アニメーションで切りの良いところで実行すると動きが滑らかになるが、少し遅くなる。

fmark(x,y) 点 (x,y) にマーカーを描き、そこを現在点とする。

xor 今後二重に描いたところは消すようにする。図形の部分消去を実現できる。元に戻すには call set とする。

1.4 fplot で描画した図の印刷法

fplot には描画した図形を記録するため、次のようなサブルーチンがあります。

mkplot(s) 現在までに描いた図を plot 形式でファイルに出力する。ファイルの名前は文字列 s で指定する。

つまり mgraph コマンドが出力するような形式でデータが出力されるので、 xplot や plot2ps を利用して、画面に図を最表示したり、プリンターで印刷したり出来ます。

例えば "call mkplot('rei31.plot')" として出力したファイル "rei31.plot" の内容を画面に表示するには

```
cat rei31.plot | xplot
```

プリンターに印刷するには

```
cat rei31.plot | plot2ps | lpr
```

とします。

複数の図を一枚の紙に印刷するには、直接印刷しないで

```
cat rei31.plot | plot2ps > rei31.ps
```

のようにして "rei31.ps" のようなデータを作っておいてから **multi** コマンドを利用します（次の例では一枚の紙に 4 つの図を収めるようにしています。 rei35.ps は 2 枚目の紙に出力されます。）:

```
multi -m 4 rei31.ps rei32.ps rei33.ps rei34.ps rei35.ps | lpr
```

2 fplot を用いた簡単な動画

解説の都合上、 fplot で例題 3-1 のようなグラフを描きましたが、実はこの種の目的には fplot はあまり向いていないかもしれません（問題 3-1 等を解くのが結構面倒なものになることが一つの理由です。また X Window System や OpenWindows に密着し過ぎていてプログラムの移植性を損ないがちなこともあげられます）。一方で (mgraph には向いていない)fplot 向きの問題もたくさんあります。例えば一つのプログラムで何本もの曲線を描いたりする場合がそうですが、ここでは時間発展する系を表示するための簡単な動画 (アニメーション) に利用してみます。

この情報処理 II では微分方程式の数値シミュレーションが一つの大きなテーマになりますが、微分方程式の解を理解するには、動画にして見るのが有効なことが多く、ここで紹介するテクニック (少し大げさな言い方ですが) を利用します。

例題 3-2: 時刻 t, 位置 $x \in [0, 2\pi]$ における変移が $f(x,t) = \sin x \cos t + \sin 3x \cos 3t + \sin 5x \cos 5t$ で与えられる弦の振動を描きなさい[6]。

現代では常識になっていることですが、実際に連続的に図を変化させなくても、十分素早く図を切り変えれば「連続的に動いている」ように見えるわけです。適当な時間間隔 Δt を定めて、時刻 $t_j = j\Delta t$ $(j = 0, 1, 2, \cdots)$ における x の関数 $f(x, t_j)$ のグラフを次々に描けばいいでしょう。古いグラフを消すために erase サブルーチンを用いています。

```
* reidai3-2.f -- 簡単な動画（紙芝居？）
      program rei32
*       定数の宣言
      real Pi,a,b,margin
      parameter (Pi = 3.14159265)
      parameter (a = 0.0, b = 2.0 * Pi)
      parameter (margin = (b - a) / 20)
*       変数の宣言
      real Time,h,dt,t,x,f
      integer i,N
*       x 軸の区間の分割数、追跡時間の入力
      write(*,*) 'x 軸上の区間の分割数 ='
      read(*,*) N
      h = (b - a) / N
      write(*,*) ' 追跡時間 ='
      read(*,*) Time
*       グラフィックスの初期化
      call openpl
      call fspace2(a - margin, -2.0, b + margin, 2.0)
*       時間間隔を決め、時刻を 0 にセット
      dt = 1.0 / 16.0
      t = 0.0
******** メイン・ループ ********
      do
        call erase
        call fmove(a, f(a,t))
        do i = 1, N
          x = a + i * h
          call fcont(x, f(x,t))
        end do
        call xflush
```

[6] これは両端を固定された弦の、ある 3 つの固有振動の和です。特に $t = 0$ の瞬間には例題 3-1 の関数になっています。

```
        t = t + dt
        if (t .gt. Time) exit
      end do
****** メイン・ループ終り ******
*       グラフィックスの後始末
      call closepl
      end
****************************************
*       両端を固定された弦のある 3 つの固有振動の和
*       時刻 t=0 では、例題 3-1 に現われる関数に一致する
      real function f(x,t)
      real x,t
      f = sin(x)*cos(t)+sin(3*x)*cos(3*t)+sin(5*x)*cos(5*t)
      end
```

これも f77x でコンパイルして実行してみて下さい。

```
reidai3-2
x 軸上の区間の分割数 =
100                                    ←入力
追跡時間 =
6.28                                   ←入力（一周期分）
```

どうですか？動いているように見えるでしょうか。長い追跡時間を指定した場合は、停止させたくなった時に C-C（コントロール C）を打って下さい。

問題 3-2: 自分で何か動画を作ってみなさい。例えば

(a) 振り子の運動

(b) はずむボール

3 第一回目のレポート提出について

次回 (5 月 19 日) の授業時間中までに出された問題のうち、一問以上を解いて、 5 月 25 日 (木) までに提出することになります。レポート作成上の注意など、詳細については次回に発表します。