JOHOSYORI No.3

情報処理 II 第 3 回 グラフの描き方 (2)

桂田 祐史

1992 年 5 月 13 日

1　注意事項

情報科学センターのワークステーションの OpenWindows は今年度に入ってからの機能拡張があだになって少し反応が鈍くなりました。そのため、つい待ちかねて二重三重に kterm 等を起動する人がいるようです。ウィンドウをたくさん開くと、メモリーが不足してワークステーションの調子が悪くなり易くなります。むやみにウィンドウを開かないように注意してください。

この講義で用意したコマンドにより表示される図のウィンドウは、マウス・カーソルをウィンドウの中に移動して左ボタンをクリックすれば閉じることが出来ます。

講義が終了して部屋を退出する際は、きちんとログアウトすることを忘れないようにして下さい。ワークステーションをハングアップ[1]させてしまって普通の方法ではログアウト出来ない場合は桂田まで申し出てその指示に従って下さい。

2　本日の例題プログラムの入手法

覚えてくれていると思いますが、 `getsample` コマンドを実行すると、カレント・ディレクトリィに今回の例題 Fortran プログラムがコピーされます。

（今回のではない）古い例題プログラムを入手するには `getsample` コマンドに第何回目の例題か数字で指定します。例えば今日第 1 回目のものを入手するには

```
waltz11% getsample 1
```

とします。

3　Fortran 向け図形描画ライブラリィ fplot の解説

3.1　はじめに – f77x コマンド

前回まで 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[2]から利用できる、一連の図形描画用のサブルーチンを用意しました。それらを利用するには Fortran プログラムを `f77o` 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]。

---

[1] 計算機が何らかの異常な状態に陥って、キー入力やマウスを使っての指示にも反応しなくなることです。

[2] 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。

[3] **懲り性の人のための注意：** `f77x` コマンドがしていることは C で書かれたサブルーチン・ライブラリィをリンクするようにしているだけです。これは情報処理 II 専用のコマンドです。

ここで用意したライブラリィは便宜上 fplot と呼ぶことにします（最初にプログラムを書いた人の姓（Fujio[4]）、言語の名前（Fortran）、座標のデータ型（floating point numbers）という 3 つの 'f' にかけてあります）。

二つの方法（mgraph,fplot）を用意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機能に限っても、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。

3.2 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 1-1 ですが、今回は fplot を用いて解きます。 reidai3-1.f を見てみましょう。

```
* 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
      real Pi,a,b,yoyuu
      parameter (Pi = 3.1415926535)
      parameter (a = 0.0, b = 2.0 * Pi)
      parameter (yoyuu = (b - a) / 20)
*       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 fspace(a-yoyuu, -3.0, b+yoyuu, 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 100 i = 1, N
        x = a + i * h
        call fcont(x, f(x))
 100  continue
*       グラフィックスの後始末
      call closepl
      end
************************************************
************************************************
      real function f(x)
```

[4]桂田の後輩の藤尾秀洋君。

```
real x
f = sin(x) + sin(3.0 * x) + sin(5.0 * x)
return
end
```

全体が、主プログラム “program 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 reidai1-1.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,fspace,fmove, fcont,closepl がグラフィックス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。これによってウィンドウがオープンされることになります。

fspace はウィンドウに座標を割り振ります。

```
call fspace(x1,y1,x2,y2)
```

とするとウィンドウの左下隅が (x1,y1)、右上隅が (x2,y2) になります。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,fspace,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 fspace(x1,y1,x2,y2)
.........
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[5]。では実行してみましょう。

```
waltz11% f77x reidai3-1.f
waltz11% 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（reidai1-1）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。

[5]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。これは藤尾君の独創ではなく、 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。

4　fplot を用いた簡単なアニメーション (1)

解説の都合上、 fplot で例題 3-1 のような問題を解きましたが、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である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、ウィンドウに描く以外のこと、例えば紙にきれいに印刷したりすることなどが難しくなります）。

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。ここでは簡単なアニメーション（動画）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テーマになりますが、微分方程式の解を理解するには、いわゆるアニメーション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）を今後何回か利用します。

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。（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。）

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。

```
waltz11% f77x reidai3-2.f
waltz11% 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。

上の例では、毎回 call erase としてウィンドウ内の図を消去していますが、これが好ましくないこともあります。描いた図を部分的に消去することも出来ます。詳細は必要が生じた時に説明することにして、今回は体験してもらうだけにしましょう。 reidai3-3.f を実行してみてください。

```
waltz11% f77x reidai3-3.f
waltz11% reidai3-3
分割数 =
10000                                  ←入力
```

5　第一回目のレポート提出について

これまでに出された問題のうち一問以上を解いて、 5 月 22 日（金）までに桂田に提出して下さい（こちらが用意した問題以外でも、講義の内容に関係することならば自分で問題を作って解いても構いません）。桂田が自室（6706 号室）にいない時は、数学科資料室（6606 号室）の金丸さん or 野町さんに渡して下さい。

レポートには氏名、ユーザー名、解いた問題を明記して下さい。自分でプログラムを書いた場合はプログラムのリスト、実験した場合は実行結果が分かるようなものを添えて下さい。