JOHOSYORI No.9

# 第２回レポートの解説 (1)

桂田 祐史

1992年6月24日

先週提出されたレポートの範囲を解説します。

## 1 問題 3-1 解説

この問題には多くの人が挑戦してくれました。f77x すなわち fplot ライブラリィに慣れるということで結構なことですが、ここでは目標を少し高く持って、どういうプログラムが良いかを考えてみましょう。

これはもちろん何を目的とするかにもよるのですが、ここでは $f$, $[a,b]$ の変更が容易であること、つまり変更の際にあまり手間が増えないようにすることを目指してみましょう。実際に計算機で仕事する時は、ただ一つの場合について計算をすれば済むということは稀で、大抵は多くの類似した場合についての計算を実行する必要がありますが、そういう際に、少し問題の条件が変わっただけで大きな変更が必要となるプログラムは望ましくないでしょう。

reidai3-1.f の段階で注意していたことは、たとえ定数であっても、名前をつけて、後で変更しやすいようにしていたことです。

```
real Pi,a,b,yoyuu
parameter (Pi = 3.1415926535)
parameter (a = 0.0, b = 2.0 * Pi)
parameter (yoyuu = (b - a) / 20)
...
h = (b - a) / N
...
```

```
call fspace(a-yoyuu, -3.0,b+yoyuu, 3.0)
...
x=a+i*h
...
```

例えばグラフの範囲を $[0, 2\pi]$ から $[0, 10]$ に変えようとする場合、一行だけ書き変えれば済むようになっていることを確かめてください（また倍精度計算をするために Pi の桁数を増やす場合も簡単に出来ることが分かるでしょう)。これが

```
yoyuu=2.0*3.1415926535/20
call fspace(-yoyuu,-3.0,2.0*3.1415926535+yoyuu,3.0)
...
x=i*(2*3.1415926535)/N
```

となっていたとすると（行数は減りますが)、変更が面倒になることが分かるでしょう。

この問題では、目盛や、軸上の点の座標を描くわけですが、以下では二つの点について考察します。

(1) グラフを描く際の範囲の設定（fspace に与えるデータ）をどうするか？

$x$ の範囲については、プログラムを使う人が指示するとしても、$y$ の範囲については事前に分からないこともあります。この問題の一つの解決法は、最初に関数値をすべて計算して配列 y(0:MAXN) に記憶しておいてから、最大値、最小値を求めて、それを元に描く範囲を決めるというものです。サンプルプログラム mondai3-1.f では

```
      do 100 i=0,N
        x(i) = a + i * h
        y(i) = f(x(i))
  100 continue
      call maxmin(y,N,maxy,miny)
      ....
      yoyuuy=(maxy-miny)/10
      ....
      call fspace2(a-yoyuux,miny-yoyuuy,b+yoyuux,maxy+yoyuuy)
```

のようにしています。サブルーチン maxmin では y(0), y(1), ..., y(N) の最大値 maxy と最小値 miny を求めています（このサブルーチンの内容は

特に難しくはありませんので、読んでもらえれば分かると思います)。ウィンドウぴったりにグラフを描くと窮屈な感じがするので、上下に $1/10 = 10\%$ の余裕を持たせるようにしました。

(2) 目盛り、ラベル

提出されたレポートで多かったのは、目盛りやラベル（ここでは軸上に記す座標などのこと）を描く命令を、その数だけプログラムの中に並べるというものでした。プログラムの再利用性の観点からすれば、これはループを作って扱うのが良いでしょう。サンプル・プログラムでは目盛り、ラベルを最初に描く位置、間隔を使う人に入力してもらって、後は自動的に処理しています。(mgraph などでは、目盛りやラベルの間隔決定まで自動的に処理しまが、そこまでするのは面倒なので、サンプル・プログラムはそういう点は使う人任せにしています。)

**注意**. ここでは、なるべく自動的にやろうとしているわけですが、本当に常にそれがいい結果を生むかどうかは別問題です。特にこの問題では、関数が周期 $2\pi$ ですから、ラベルの間隔を例えば $\pi$ にして、pi とか、2pi などのラベルを書く方が良いと思われますが、こういうことをプログラムで自動的に処理するのは難しいです。

ラベルについては、xclick のような real 型の変数に収められた数値データ（ここでは real 型）を label サブルーチンで表示するのに工夫が必要です。label サブルーチンは引数として文字列しか受け付けてくれませんから、xclick の内容を文字列に変換しないといけません。サンプル・プログラムでは、そのために「内部ファイルへの書き込み[1]という Fortran の文字列処理機能を利用しています。

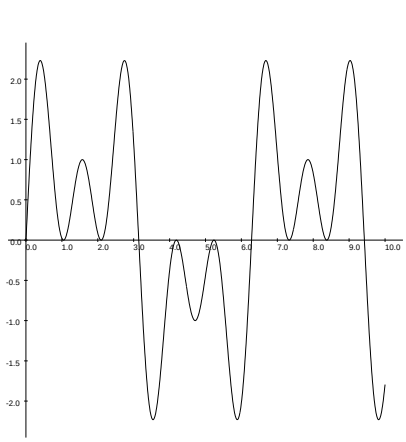
```
real xclick
character s*20                    ← 長さ 20 の文字列変数 s の宣言
...
write(s,*) xclick                 ← s に xclick の10進表示を納める
call delsp(s,length)              ← 余分な空白(space0)を削除する
```

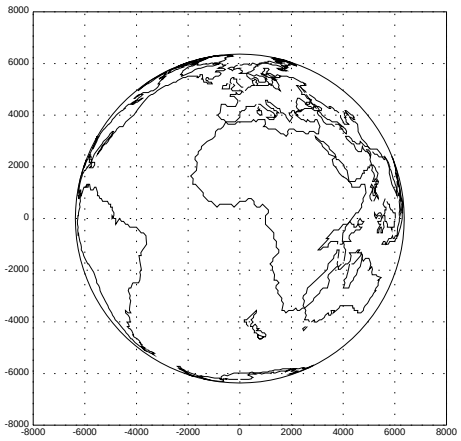
[1]内部ファイルとは要するに文字変数のことです。write 文は「普通」は標準出力（おおざっぱに言ってディスプレイ画面のことだと思ってください）やディスク・ファイルなどの「外部にある」ファイルに出力するが、実は実行中のプログラム「内部」の文字変数にも出力できるのだ、というニュアンスです。これは例えば C 言語だったら sprintf 関数等で実現できる機能のことです。

```
...
call label(s)                    ← 文字列 s をウィンドウに表示する
```

`write(s,*) xclick` としているのが「内部ファイルへの書き込み」です。この時 `s` の先頭には空白が入っているので、それを削除するために、サブルーチン `delsp` を呼んでいます。`delsp` の中でも色々な文字列処理機能を使っていますので、興味ある人は読んでみて下さい。

以下に示す図（左側）は、区間 $[0, 2\pi]$ で、$x$ 軸については 1 ずつ、$y$ 軸については 0.5 ずつの間隔で目盛とラベルをつけたものです。




## 2　透明な地球 – 文字列変数の扱いの参考に

第一回レポートの範囲になりますが、問題 1-5 のヒントとなるプログラムを `hint1-5.f` を公開します。このプログラムでも文字列変数を使っているので、興味のある人は参考にして下さい。コンパイルして実行すると “`mapdata`” を読み込んで “`mapdata.new`” というファイルを作成します。これは経度・緯度のデータを直角座標系（北極を $(0, 0, R)$, 南極を $(0, 0, -R)$, 緯度と経度が共に 0 の点を $(R, 0, 0)$ としたもの。ここで $R$ は地球の半径を km 単位で表わしたもの値です。）のデータに変換して、$y$,$z$ 座標を出力したものです。

```
waltz11% f77o hint1-5.f          ← コンパイルする
waltz11% hint1-5                 ← 実行する（画面には何も出ない）
```

```
waltz11% ls mapdata.new                ← mapdata.new が
出来たか？
mapdata.new                            → 出来た。
```

このデータを mgraph にかければ、「透明な地球」が表示されます。

```
waltz11% cat mapdata.new | mgraph -b | xplot
```

この問題のポイントの一つは、mgraph を用いるには、座標データの変換だけをすればいいが、線を区切るために入っている “" "” という空ラベルをうまく扱うために工夫が必要だということです。率直に言って Fortran はこの種のデータの扱いが苦手です（Fortran で何か問題を解くに当たっては、形式がきれいに揃った行が整然と並んでいるようなデータ・フォーマットを設計するのが普通です)。サンプル・プログラムでは、一度行を文字列変数 “line” に読み込んで、引用符があるかどうかを判定しています。

```
      ....
      character line*30                ← 長さ 30 の文字列
変数の宣言
      ....
      read(1,1000) line                ← 文番号 1000 の形
式で入力
      if (line(1:1) .eq. '"') then     ← line の 1 文字目
は " か？
      ....
      else
        read(line,*) keido,ido         ← line から二つの
数を得る
      ....
      endif
      ....
 1000 format(A20)
      ....
```

このプログラムのもう一つの要点は、経度、緯度のデータをデカルト座標系に直すところでしょう。これは極座標の式とほぼ同様の変換式です（興味があればプログラムを読んで下さい)。

3次元の対象物である球面を描くのに、単純に、一つの座標平面（ここでは $yz$ 平面）への投影を行いましたが、こういう 3次元グラフィックスの話は始めると長くなるのでカットします。

（プログラムの表題にした「透明な地球」は、裏側のものまで描いてしまったために出来た、いわば手抜きの結果です。「不透明な地球」や「回る地球」等に挑戦するのも面白いと思います。そういうプログラムはレポートとして受け付けます。）

# 3　第4回のサンプル・プログラムの訂正

第4回目に配った、例題 4-1 等の Euler 法のプログラムで

```
*       Euler 法による計算
        do 200 i=0,N-1
          t=t+h
          x=x+h*f(t,x)
  200   continue
```

のように書いていたのですが、これは私の不注意で生じたミスで、正しくは（t と x の更新の順番を逆にした）

```
*       Euler 法による計算
        do 200 i=0,N-1
          x=x+h*f(t,x)
          t=t+h
  200   continue
```

とすべきでした。この回に扱った問題では関数 $f$ が $t$ に依存していないため、計算結果がおかしくならずに済んでいたのですが、それだけに気付くのが遅れてしまいました。