最低限の数値計算入門

はしもとじょーじ　(岡山大学理学部地球科学科)

プログラムを書いて計算をしてみることと，計算した結果を書いてみることについて，最初のとっかかりの部分の解説である．ひとつめのプログラミングについては FORTRAN 77 を，ふたつめの図を描くについては gnuplot を使う．

１．計算する　(FORTRAN 77)
２．図を描く　(gnuplot)

計算するのも図を描くのも，FORTRAN 77 と gnuplot でないものを使うこともできる．FORTRAN 77 と gnuplot を取り上げているのは，教える側の能力などによる制約のためで，ここに選んだものでなければならないというわけではない．なので、各人の好みでここに挙げた以外のものを使ってもらってもよい．

また、教えるといっても手取り足取り教える時間はないので、教えるのは最初のとっかかりの部分だけである．そこから先は参考文献なりを参照して各自の努力でなんとかして欲しい．

１．計算する　(FORTRAN 77)

計算機は指示された通りに動く．だから計算機を使って計算がしたければ、計算機に指示をしなければならない．計算機に対する指示は決まった書式に従って書かなければならない．この計算機が理解可能なように書かれた指示のことをプログラムと呼ぶ．

プログラムを作成する前に、いくつかやることがある．まず何を知りたいのかを明らかにする．そして何を解けばよいのか、解くべき方程式などを導出する．それらを計算機でどのように解くか考える．プログラムを作成するのはこれらの作業が終わってからである．

数値計算と言うと最後のプログラムを書く作業のことを思い浮かべる人が多いかと思うが、その作業自体はそれほど難しくはない．数値計算で難しいのはその前段である計算機でどのように解くかを考える部分である．桁落ちとか丸め誤差といった計算機のいろいろな特徴を踏まえつつ、どのようにしたら計算機でうまく計算できるかを考えなければならない．そこらへん、知らなければいけないことはたくさ

んあるのだが、ここではそれらを教えることはしない．参考文献なりを参考にして各自で勉強してもらいたい．

１－１．参考文献

FORTRAN の教科書やら解説本はたくさんあるので、自分の好みで選べばよい．いちおう、はしもとの好みをいくつか挙げておく．

１． 原田賢一 (1986) FORTRAN 77 プログラミング、サイエンス社.
２． Press et al. (1992) Numerical Recipes in FORTRAN, 2nd ed., Cambridge Univ. Press.
３． 伊理正夫・藤野和建 (1985) 数値計算の常識、共立出版

１は FORTRAN 77 の文法などが書いてある教科書． ２は FORTRAN 77 のプログラムだけでなく、数値計算のいろいろが書いてる本．何か計算しようと思ったとき、とても役にたつ．Ｃ言語版もある． ３は FORTRAN 77 の本ではない．数値計算をする人が常識として知っておくべきこと(例えば桁落ちなど)について書かれた本．計算仕事する人は一読することを強くお勧めします．

１－２．最低限の FORTRAN 入門

FORTRAN のプログラムを作成して実行する手順は以下の通り．

(1) プログラムを書く
(2) コンパイルする
(3) 実行する

(1)プログラムの作成．vi などのエディターを用いてプログラムを作成してファイルに保存する．プログラムを保存したファイルはソースファイルと呼ばれる．ソースファイルの拡張子は .f でなければならない(処理系によっては `.for` とするものもある)．ここではソースファイルの名前を `example.f` としておくことにする．
(2)コンパイル．シェルのプロンプトに対して以下のように入力すれば、ソースファイル(`example.f`)が処理されて、`a.out` という名前の実行型ファイルが生成する．

```
$ f77 example.f
```

難しいことはわからなくてもよい．とにかくこうすると実行型ファイルというものが生成することだけ知っていれば、とりあえずはなんとかなる．

(3)実行．コンパイルして生成したファイル(a.out)を実行するには、シェルのプロンプトに対して以下のように入力する．

```
$ ./a.out
```

これで example.f に書いたプログラムが実行される．

ちなみに実行するとき、後ろに > xy.dat のように付け加えると

```
$ ./a.out > xy.dat
```

a.out の出力(Write 文で書き出されるものなど)が xy.dat という名前のファイルに書き込まれる．これはリダイレクションと呼ばれる技で、安直にデータをファイルに書き出したいときなどに便利である．

１－３．プログラミング入門

「習うより慣れろ」とにかく自分でプログラムを書いて学んでいくのが近道である．とりあえず以下の４つ

(1) 変数の宣言
(2) Do ループ
(3) If 文
(4) Write 文

の使い方がわかるだけで、かなりのことができるようになる(はず)．ということで、上の４つを使っている簡単なサンプルプログラム(級数 $\sum_{k=1}^{n} k = 1 + 2 + \cdots + n = \frac{1}{2} n(n+1)$ の計算)を示す．

```
1         Program CalcSum
2         Implicit NONE
3
4   c -----
5         Integer n, ns, i
6
7   c -----
8   c input 'n'
9    1    Continue
```

```
10        Write(*,*) 'input n'
11        Read(*,*) n
12        If (n.le.0) then
13            Go to 1
14        End If
15
16  c calculation
17        ns = 0
18        Do 100 i = 1, n
19          ns = ns + i
20   100  Continue
21
22  c output
23        Write(*,*) '1+2+...+n ='
24        Write(*,*) ns
25
26        Stop
27        End
```

解説

プログラムは基本的に前から順番に実行される．
左端の 6 文字分は行番号を書く場所となっている．プログラム文は左端 6 文字分には書かない．行番号を全ての行につける必要はない(というか，ふつうは必要なところ以外には行番号をつけない)．
FORTRAN では大文字と小文字は区別されないので、Continue も CONTINUE も continue も cOnTinUE も同じ意味になる．ただし最後の例は読みにくいのでやめること．

1行目は `Program` に続けてプログラムの名前を入れる．名前はだいたいはなんでもよい．
2行目はおまじない．コンパイラーによっては，このおまじないが使えない場合もある．使えない場合は仕方がないが，使える場合は使った方がよいと，はしもとは思う．

4行目．先頭に `c` と書くとプログラムとして解釈されないコメントの行になる．

5行目．整数型の変数 n, ns, i を宣言．ちなみに、実数型(単精度)なら Real、
倍精度の実数型なら Real*8 として変数を宣言する．
計算の前に使用する変数は全て定義する．

9-14行目．ここは一つの固まり．
いくつまでの和を計算するかの問い合わせをする．
9行目の Continue は何もしない，という命令．行番号は１．
10行目で'input n'と画面に表示する．
11行目で入力された文字を読んで、n に入れる．
12-14行目．If文．A.le.B は A≤B の意味．
従って、n.le.0 は、n が０か負の場合に真になる．
n が０か負の場合は、13行目が実行されて，行番号「１」の行にとぶ．
すなわち9行目に飛んで，再度 n を入力させる．

17-20行目でまた一つの固まり．
ここで級数の計算をしている．
17行目は，変数 ns に値 0 を代入している．変数は宣言されただけでは何の値が入っているかわからないので，使う前に必ず値を入れる必要がある．
18行目-20行目で、和を計算している(Doループ)．
Do の行と，Do の後に書かれた行番号の行までの間を繰り返し実行する．
行番号に続いて書かれている変数はループを回すための変数(この例では i)
この変数は必ず整数型を使わなければならない．
ループを回す変数は所定の値(この例では 1)から始まり，１回実行されるたびにループを回す変数の値は変更(+1)され，所定の値(この例では n)になるまで繰り返される．

23行目は '1+2+...+n =' を出力．
24行目は ns の値を出力．
' ' で括られた文字列はそのまま出力される．
括られていない場合には，変数に代入されている数値や文字列が出力される．

26行目はプログラムを終了する合図．

27行目の End は、1行目の Program と対になっていて、
Program と End の行に囲まれた行がプログラムであることを示す．
忘れたらプログラムをコンパイルできないので注意．

１－４．デバッグの方法

プログラムに含まれる誤りのことをバグ(bug)と呼ぶ．このバグを取り除く作業がデバッグ(debug)である．とりあえずプログラムを書いてみた(コンパイルして実行型ファイルも作れた)けれど、実行してみると期待したように動作しない．そういう場合はプログラムのどこかにバグがあるわけで、プログラムを正常に動作させるためにデバッグ(誤りを発見・修正)の作業が必要とされる．

デバッグの方法はいろいろあるが、最も基本なのは Write 文を使うもの(と、はしもとは思う)．プログラムのあちこちに Write 文を埋め込み、いろいろな変数を出力させてみる．プログラムが途中で停止する場合には、どの Write 文までが実行されたかを見ることで、どこに問題がありそうか見当をつけることができる．また変数に代入されている数値を見ることで、ちゃんと意図した通りの値が変数に代入されているかどうかも確認できる．そうなっていなければ、それより前で何か間違っているわけである．

デバッグの作業においては、デバッガと呼ばれる種類のソフトウェアを活用することもできる．デバッガにもいろいろあるようなので、興味を持った人は調べてみましょう．

課題１－１．級数

左辺の級数を数値的に計算して右辺と一致することを確認する．

$$\sum_{k=1}^{n} k^2 = 1^2 + 2^2 + \cdots + n^2 = \frac{1}{6} n(n+1)(2n+1)$$

$$\sum_{k=1}^{n-1} \sin\frac{k\pi}{n} = \sin\frac{\pi}{n} + \sin\frac{2\pi}{n} + \cdots + \sin\frac{(n-1)\pi}{n} = \cot\frac{\pi}{2n}$$

$$\sum_{n=0}^{\infty} \frac{1}{n!} = 1 + \frac{1}{1!} + \frac{1}{2!} + \cdots = e$$

２．図を描く　(gnuplot)

　いろいろと計算した結果が出てくると、それを図に描いてみたくなる．gnuplot はそういうとき比較的手軽に使うことができる．以下で述べるようにオンラインマニュアルもあるのだが、以下のサイトなども gnuplot を使う上で参考にすることができる．

入門からかなり細かい点まで網羅している優れたサイト

```
http://t16web.lanl.gov/Kawano/gnuplot/
```

マニュアル日本語訳

```
http://takeno.iee.niit.ac.jp/%7Efoo/gp-jman/gp-jman.html
```

２．１．起動と終了とヘルプ

　シェルのプロンプトが出ているところで

```
$ gnuplot
```

と入力すると、gnuplot が起動してオープニングのメッセージの後に gnuplot のプロンプト

```
gnuplot>
```

がでる．gnuplot を終了するには、このプロンプトが出ているところで

```
gnuplot> exit
```

とする．

　gnuplot のごく簡単な使い方を説明する前に、オンラインマニュアルの読み方を説明しておく．困ったときや、何かやりたいことがあってどうやるかわからないときには、このオンラインマニュアルを読んでみるとよい．オンラインマニュアルの読むには gnuplot のプロンプトが出ているところで

```
gnuplot> help
```

と入力すればよい．概要説明から始まって項目を選択しながらいろいろな機能についての説明を読むことができる．

２．２．離散データのプロット

ファイル(xy.dat)にあるデータを読み込んでプロットするには、

```
gnuplot> plot 'xy.dat'
```

とする．このファイル(xy.dat)には以下の例のように、各行にx座標とy座標の順に空白で区切ったデータを書いておく．

```
1  1
2  4
3  9
5  25
6  36
```

折れ線グラフにしたいときには、

```
gnuplot> plot 'xy.dat' with lines
```

などとすればよい．

２．３．ファイルへの保存

作成したグラフを保存するには

```
gnuplot> setenv terminal png
gnuplot> set output 'xy.png'
gnuplot> replot
gnuplot> set output
```

これでPNG形式のファイル(xy.png)に保存される．

２．４．お手軽アニメーション

gnuplot そのものにアニメーションの機能はないが、あらかじめ作成しておいたデータファイルを順次読み込んで表示することで、アニメーションさせることができる．例えば、gnuplot に実行させる命令を箇条書きにした次のようなファイル animate.plt を作成して、

```
plot 'time00.dat'
pause 1
plot 'time01.dat'
pause 1
plot 'time02.dat'
pause 1
plot 'time03.dat'
pause 1
plot 'time04.dat'
```

次のように実行する．

```
gnuplot>  load 'animate.plt'
```

こうすると、animate.plt に書き込んだ命令が逐一実行されていく．ちなみに pause は、その後に指定された時間(単位は秒)だけ実行を停止する、という命令．すなわち上のファイルは、描画・１秒停止・描画・１秒停止・・・・、という処理を繰り返しおこなう．

謝辞

１－３プログラミング入門にあるプログラム解説の第１稿は神戸大学の村上真也さんに書いていただきました．その他、村上氏からもらったいくつかのコメントに基づいて文書を改訂しました．ありがとうございます．