第 4 章　計算機言語入門

4.1　FORTRAN 入門

計算機の言語には色々なものがある。それぞれ特徴を持っており、分野や用途によって使い分けられている。科学計算においては、FORTRAN の重要度が高い。C や C++ も良く使われる。他に、数式処理のための言語として、REDUCE や LISP や Mathematica も重宝である。また、 awk, perl, ruby といったスクリプト言語と言われるものもある。どう分類するか難しいが Java も様々なところで使われている多くの計算機環境では、こういった言語を組み合わせて使用する事が出来るので、用途や目的に応じて、multilingual な特徴を生かした使い方をするのが良いであろう。

実際に天文学の研究をするという観点からは、多くの場合に様々な言語を使う必要に迫られる。というのは、観測的研究、理論的研究のどちらにしても、(本当に「紙と鉛筆」だけでできる理論的研究の一部を別にすれば) 計算機を使ってデータ解析やシミュレーションをすることが必須であり、そのためのプログラムを作る必要がある。ところが、こういったプログラムは大規模・複雑なものになってきており、一人でゼロから書けるものではなかなかない。何人もが共同で作ったり、また、過去に別の人が作ったものを変更・改良して使うことが普通である。このために、過去の資産が Fortran であれば Fortran が使えないといけないし、C であればそれでなんとかする必要がでてくる。

この講義では Fortran を扱い、計算天文学 II では C++ を扱うことにする。2 種類の非常に違う言語を短期間に扱うのは混乱を招く面もあることは否定しないが、逆に言語 1 つだけしか知らないと、2 番目を学ぶのは非常にハードルが高いからである。2 つの言語を一応理解していると、その間には共通点と相違点があることが分かり、さらに他の言語を理解する必要ができた時にも類推が効くようになる。

FORTRAN は、1950 年代に開発された言語で、計算機言語としては、歴史が古い。進歩の速い計算機に関しては、歴史が古いという事は、多くの場合、陳腐で劣る事を意味する事が多い。事実、50 年代或いはそれ以後開発された言語で、今日では使用されなくなった言語は非常に多い。そういった環境の中にあっても、FORTRAN は、色々な意味で、優れた点を多く持った言語なので、様々な改良を加えられながら、今日まで、科学計算用の標準的言語の一つとして、使われ続けている。その特徴としては、

- 各時代の最高速の計算機上で使われ続けている事
- 大きなプログラムでも容易に構築できる事
- 機種依存性が少ない事

- 数値計算ライブラリが数多く揃っている事

等が挙げられよう。

なお、Fortran の言語仕様には、 Fortran 66, 77, 90 とほぼ 10 年毎に大きな改訂が加えられてきた。この講義では、基本的には 77 に準拠するが、 90 での拡張をある程度取り入れた形でプログラムを書く。現在広く使われているのがその形であり、また計算機室のシステムで利用可能なのもそれだからである。

計算機の演算能力を最大限に発揮するには、計算機が直接理解できる機械語 (アセンブラ) でプログラムを書けば良いのだが、このためには、プログラムを書く人間の側に多大の努力を要し、効率悪く、とても一般の人向きではない。そこで、FORTRAN 等、人間が理解しやすい言語でプログラムを書き、それを機械語に翻訳させる作業 (コンパイルという) を計算機自身にさせる訳だが、FORTRAN には、高速にしかも最適化された機械語に翻訳する技法が進んだ形で取り入れられている。これらの理由から、FORTRAN は長い間多く利用され、それが更に、ライブラリを完備させる等、良い方向にフィードバックされて、より便利な道具として使われる理由となっている。

FORTRAN を使って計算をしたい時には、FORTRAN の言語で書かれた一つのファイルを先ず作らなければならない。こういったファイルをプログラムと呼ぶ。例として、球の半径が与えられた時に、球の表面積と体積を求める計算を考えよう。そのためのプログラムの例は、第??章で、エディタを使ってのファイル作成練習で書いた、`sphere.f` である。念のため、下に掲げておこう[1] 。

```
      Program sphere
      Implicit NONE
c constant:
      Real PI
      Parameter (PI=3.141593)
c local variable:
      Real radius
      Real area, volume
c begin:
      radius = 10.0
      area   = 4. * PI * radius**2
      volume = area * radius /3.
      Write(*,*) 'radius:',radius
      Write(*,*) ' surface area =',area,' volume =',volume
c end:
      Stop
      End
```

[1] http://www.astron.s.u-tokyo.ac.jp/~shibahas/sphere.f
尚、Web ページを見るには、ターミナルウィンドウで、
`% mozilla &`
と入力して見よ。

4.1.1 FORTRAN の約束事

FORTRAN は、カードでプログラムを入力していた時代 (と言っても、何の事かわからない世代の時代になってしまった。要するに、太古の時代からの意味) から使われていたせいで、プログラムを書く上で、様々な制約、約束事がある。

- FORTRAN のプログラムを構成する文は、1 行の 7 文字目から 72 文字目までに収める。上の例で、多くの行で、最初の 6 文字分下がっているのは、このためである。
- 1 行に収まりきらない時は、次の行の 6 文字目に & などの文字を入れる事で、継続行を意味させる事にして、2 行にわたらせる。もっと長い時も、同様。
- 1 文字目に c 或いは * の文字のある行はコメント行とする。
- ! の後ろもコメントと見做す。
- 1 文字目から 5 文字目まで数字列は行番号を表す。
- 大文字と小文字の区別はしない。
- プログラムは Program 文で、名前を宣言し、Stop 文と End 文で終わる。

4.1.2 式と代入

左辺=右辺 で右辺の値を左辺の変数の値とする。左辺は変数でなければならない。右辺は数値もしくは数式だが、数式の場合はその段階で数値計算可能である必要がある。
例 .

```
a = 1.0
b = a + 1.0
```

4.1.3 四則演算

右辺で行う四則演算は表 4.1 の様に記す。

4.1.4 変数と型

FORTRAN が扱う変数には、整数、実数、複素数、論理型、文字列がある。サンプルプログラムの中で、local variables の次の行で、Real radius とあるのは、radius という変数が実数型である事を宣言している。整数、実数、複素数を混合して演算しないようにプログラムを書くことを心掛けたい。混合演算をすると、右辺の代入結果は左辺の変数の型になる。表 4.2 に、変数と型に関する制限事項をまとめる。

表 4.1: FORTRAN での四則演算

演算	FORTRAN	例
加算	+	`x + y`
減算	−	`x - y`
掛算	*	`x * y`
割算	/	`x/y`
冪乗	**	`x**y`
括弧	( )	`(2.*(x + y))/(3.*(a + b))`

表 4.2: FORTRAN で扱われる変数の型

型	FORTRAN	大きさ
整数型	`Integer`	$-32768 \sim 32767$
	`Integer*4`	$-2147483648 \sim 2147487647$
実数型	`Real`	$\pm 0.29 \times 10^{-38} \sim \pm 1.7 \times 10^{38}$ (有効数字 7 桁)
	`Real*8`	$\pm 0.29 \times 10^{-38} \sim \pm 1.7 \times 10^{38}$ (有効数字 15 桁)
複素数型	`Complex`	(有効数字 7 桁)
	`Complex*16`	(有効数字 15 桁)
論理型	`Logical`	`.true.` (真) 或いは `.false.` (偽)
	`Logical*2`	
文字型	`Character*`サイズ	サイズ分の長さの文字列

4.1.5 宣言文

変数の型を宣言する例を見た。他にもこういった宣言文がある。先の例では、

```
Implicit NONE
Parameter (PI=3.141593)
```

がそうだ。最初の例は、そのプログラムで使用する変数は全て型宣言をしてから使う事を宣言している。FORTRAN では、最初の文字によって、計算機が型を暗黙のうちに解釈するという、便利なような恐ろしいような約束事がある。一見便利なようなルールだが、プログラムを書き直したり、長いプログラムを書く時等、使う変数をきちんと把握している方が、結局は便利である事の方が多い。`Implicit NONE` をプログラムの頭で宣言しておくと、型宣言していない変数をプログラム中で引用していたら、計算機が警告してくれる。

二番目の例の `Parameter` 文は、変数 `PI` の値を指定している。式として書いても同じ事だが、物理定数等この `Parameter` 文を用いて宣言すると、値を直す時等、場所が決まっているから、便利で

ある。

DIMENSION の宣言は、良く使うであろう。y_i $(i = 1, ..., 10)$ 等を変数として、使いたい時がある。この様な時、

```
DIMENSION y(10)
```

の様に宣言すると、y(1), ... , y(10) が使えるようになる。引数が二次元ならば、

```
DIMENSION y(i,j)
```

の様にすれば良い。i と j は、それより前の Parameter 文でサイズを宣言しておけば良い。

4.1.6 WRITE 文

先のプログラムで、

```
Write(*,*) 'radius:',radius
Write(*,*) ' surface area =',area,' volume =',volume
```

が、WRITE 文である。'radius:' は radius:と表示させる事を意味し、カンマに続く radius は、変数 radius の値を表示させる事を意味している。

4.1.7 IF 文

変数の値などの条件によって、次に行うべき演算が違う事がある。FORTRAN では、次の様にして場合分けを行う。

```
if (code.eq.1) then
   ...
else
   ...
end if
```

或いは次の様な更に細かい条件分岐も出来る。

```
if (code.gt.1) then
   ...
else if (code.eq.1) then
   ...
else
   ...
end if
```

表 4.3: FORTRAN で扱う大小符号や等号

関係式	FORTRAN
$a = b$	a.eq.b
$a \neq b$	a.ne.b
$a < b$	a.lt.b
$a \leq b$	a.le.b
$a > b$	a.gt.b
$a \geq b$	a.ge.b
$a \wedge b$	a.and.b
$a \vee b$	a.or.b
$\neg a$	.not.a

4.1.8　DO ループ

例えば

$$y = \sum_{i=1}^{10} a_i$$

を計算するために、DO ループと呼ばれる繰り返し演算がある。

```
y = 0.0
Do i=1,10
  y = y + a(i)
End do
```

i が 10 以下である限り繰り返すといった場合には、次の様な Do while が便利だ。

```
i = 1
Do while (i.le.10)
  y = y + a(i)
  i = i + 1
End do
```

4.1.9　条件判断

If 文や While 文の条件判断では、大小符号や真偽等を条件として使うが、それらは次のように表される。

4.1.10 サブルーチン

先に掲げた球の表面積と体積を求める計算で、半径を 1 から 2 刻みで 10 まで変えて計算する事を考えよう。同じ作業の繰り返しだから、その部分だけを別のパーツとしてプログラムを構築できたら、プログラムは短く、かつ見易いものとなるであろう。サブルーチンはこの様な目的に、使われるプログラムのパーツである[2] 。

```
      Program sphere2
      Implicit NONE
c constant:
      Real RADIUS_INITIAL,MAX_RADIUS,DELTA_R
      Parameter(RADIUS_INITIAL=1.0,MAX_RADIUS=10.0)
      Parameter(DELTA_R=2.0)
c local variable:
      Real radius
      Real area, volume
c begin:
      radius = RADIUS_INITIAL
      Do while (radius .le. MAX_RADIUS)
         Call calculation(radius, area, volume)
         Write(*,*) 'radius:', radius
         Write(*,*) ' area =',area,' volume =',volume
         Write(*,*)
         radius = radius + DELTA_R
      End do
      Stop
      End

      Subroutine calculation(radius, area, volume)
      Implicit NONE
c input:
      Real radius
c output:
      Real area, volume
c constant:
      Real PI
      Parameter (PI=3.141593)
c begin:
```

[2] ここに挙げた例は、http://www.astron.s.u-tokyo.ac.jp/~shibahas/sphere2.f

```
      area   = 4. * PI * radius**2
      volume = area * radius /3.
      Return
      End
```

メインプログラムでは、半径を与えるだけで、計算はサブルーチン calculation で行っている。そしてメインプログラムとサブルーチンとの間のやり取りは、メインプログラム中の call 文で行う。その際、引数 (radius,area,volume) は、radius がメインプログラムから与えられ、area と volume は、サブルーチンから計算の結果として、メインプログラムに返される。それぞれの引数の型は、メインプログラムとサブルーチンの間で必ず一致させなければならない。但し、引数の名前は両者の間で異なっていても構わない。この例に見るように、引数はメインプログラムからサブルーチンにも渡せるし、逆にサブルーチンからメインプログラムにも渡せる。どちらからどちらに渡されているかは、一見して明らかな訳ではないので、上のサブルーチン calculation の例のように、コメントで注釈を附けておくと、便利である。また、サブルーチンから、別のサブルーチンを呼ぶことも出来る。サブルーチンは Return と End で終わるのが約束である。

4.1.11 READ 文

先に掲げた球の表面積と体積を求めるプログラムでは、球の半径は、プログラムの中で与えた。任意の半径を与えられるようにならないだろうか。下に掲げるプログラムは、端末から、半径を与える様に、sphere2.f を直したものである[3] 。(サブルーチン calculation は再掲していない。)

```
      Program sphere3
      Implicit NONE
c constant:
      Real RADIUS_INITIAL,MAX_RADIUS,DELTA_R
c local variable:
      Real radius
      Real area, volume
c begin:
      Write(*,*) 'Input RADIUS_INITIAL:'
      Read(*,*) RADIUS_INITIAL
      Write(*,*) 'Input MAX_RADIUS:'
      Read(*,*) MAX_RADIUS
      Write(*,*) 'Input DELTA_R:'
      Read(*,*) DELTA_R

      radius = RADIUS_INITIAL
```

[3] http://www.astron.s.u-tokyo.ac.jp/~shibahas/sphere3.f

```
      Do while (radius .le. MAX_RADIUS)
         Call calculation(radius, area, volume)
         Write(*,*) 'radius:', radius
         Write(*,*) ' area =',area,' volume =',volume
         Write(*,*)
         radius = radius + DELTA_R
      End do
      Stop
      End
```

Read(*,*) が、端末から変数を読み込ませるときに用いる文である。

4.1.12 ファイルへの入出力

上の例で、Read 文や Write 文で、端末に入出力するのではなく、ファイルに入出力するには、次の様にすれば良い[4] 。

```
      Program sphere4
      Implicit NONE
c constant:
      Real RADIUS_INITIAL,MAX_RADIUS,DELTA_R
c local variable:
      Real radius
      Real area, volume
c file open:
      Open(Unit=2,File='Input.dat',Status='OLD')
      Open(Unit=3,File='Output.dat',Status='NEW')
c begin:
      Read(2,*) RADIUS_INITIAL,MAX_RADIUS,DELTA_R
      radius = RADIUS_INITIAL
      Do while (radius .le. MAX_RADIUS)
         Call calculation(radius, area, volume)
         Write(3,*) 'radius:', radius
         Write(3,*) ' area =',area,' volume =',volume
         Write(3,*)
         radius = radius + DELTA_R
      End do
      Stop
```

[4] http://www.astron.s.u-tokyo.ac.jp/~shibahas/sphere4.f

```
      End
```

(*,*) のところが (2,*) となったため、入力は 2 番の論理装置から行なわれる。その 2 番の論理装置が Input.dat という名前の既製のファイルである事を最初の Open 文が記述している。Write 文が、(3,*) であるから、出力は 3 番の論理装置に対してなされる。二つ目の Open 文は、同様に 3 番の論理装置が Input.dat という名前のの新規のファイルである事を記述している。

4.1.13 入力回数が予め決まっていない場合

入力データの数が予めわかっている場合は、先の方法や、それを DO ループでデータの読み込み回数分だけ繰り返せば良いが、入力データの数が予めわかっていない場合はどうすれば良いであろうか。FORTRAN の READ 分は、この様な場合に備えて、論理装置番号、読み込み形式を表す*のほかに、もう一つ割り込み文番号を指定できるようになっている。次のプログラム[5] で、Read 文が (2,*) ではなく、(2,*,end=900) となっている事に注意しよう。これは、N 回まで DO ループが回ろうとしているうちに、読み込みデータがなくなったら、900 番の文番号に飛ぶ事を意味している。900 番の文では、その時点での変数 i を書き出すように命令しているから、これにより、幾つデータを読んだかが判る。この様な場合、DO 文は、途中で割り込みが入ることになり、完結しないことに注意しておこう。

```
      Program sphere5
      Implicit NONE
c constant
      Integer N
      Parameter(N=2000)
c local variable:
      Real radius
      Real area, volume
      integer i
c file open:
      Open(Unit=2,File='Input.dat',Status='OLD')
      Open(Unit=3,File='Output.dat',Status='NEW')
c begin:
      Do i=1,N
         Read(2,*,end=900) radius
         Call calculation(radius, area, volume)
         Write(3,*) 'radius:',radius,' area =',area,' volume =',volume
      End do
  900 Write(*,*) 'number of data=', i-1
```

[5] http://www.astron.s.u-tokyo.ac.jp/~shibahas/sphere5.f

表 4.4: FORTRAN の組込み関数

FORTRAN	意味	FORTRAN	意味
`sqrt(x)`	$\sqrt{x}$	`exp(x)`	$\exp(x)$
`log(x)`	$\ln x$	`log10(x)`	$\log x$
`sin(x)`	$\sin x$	`asin(x)`	$\arcsin x$
`cos(x)`	$\cos x$	`acos(x)`	$\arccos x$
`tan(x)`	$\tan x$	`atan(x)`	$\arctan x$
`sinh(x)`	$\sinh x$	`cosh(x)`	$\cosh x$
`tanh(x)`	$\tanh x$	`abs(x)`	x の絶対値
`int(x)`	x の整数部	`float(n)`	整数 n を実数に変換
`max(x,y,z,...)`	$x, y, z, ...$ の最大値	`min(x,y,z,...)`	$x, y, z, ...$ の最小値

```
      Stop
      End
```

このプログラムで、N 以上読み込みデータがあると、number of data として正しくない答え N を表示する。

4.1.14 組み込み関数

計算の中で、三角関数などの関数を使いたい場合も多い。FORTRAN には様々な関数が標準で用意されており、それらを利用する事が出来る。例えば、プログラムの中で y=cos(x) とあれば、変数 y には、変数 x の余弦が代入される。良く使われる関数を表 4.4 に掲げる。

混合演算をしないように、と書いたが、実数 x を整数に型変換するには、int(x) を使い、整数 x を実数に型変換するには、float(x) を使う。

4.1.15 自分で定義して作る外部関数：FUNCTION 文

FORTRAN が用意してくれる標準的関数以外に、自分で外部関数を定義して使う事も出来る。例えば、球の表面積と体積を半径の関数として、それぞれ関数 S(r)、V(r) を

```
      real function S(r)
      implicit NONE
c constant:
      real PI
      parameter (PI=3.141593)
```

```
c local variable:
      real r
c begin:
      S=4.*pi*r**2
      return
      end


      real function V(r)
      implicit NONE
c constant:
      real PI
      parameter (PI=3.141593)
c local variable:
      real r
c begin:
      V=4./3.*pi*r**3
      return
      end
```

で定義して、メインプログラムからは、

```
      Program sphere6
      Implicit NONE
c constant:
      Real RADIUS_INITIAL,MAX_RADIUS,DELTA_R
c local variable:
      Real radius
      Real area, volume, S, V
c begin:
      Write(*,*) 'Input RADIUS_INITIAL:'
      Read(*,*) RADIUS_INITIAL
      Write(*,*) 'Input MAX_RADIUS:'
      Read(*,*) MAX_RADIUS
      Write(*,*) 'Input DELTA_R:'
      Read(*,*) DELTA_R

      radius = RADIUS_INITIAL
      Do while (radius .le. MAX_RADIUS)
```

```
      Write(*,*) 'radius:', radius
      area  =S(radius)
      volume=V(radius)
      Write(*,*) ' area =',area,' volume =',volume
      Write(*,*)
      radius = radius + DELTA_R
    End do
    Stop
    End
```

のように、外部関数を使う。FUNCTION 文は、メインプログラムの後ろにつけて、コンパイルするのが良いだろう[6] 。

4.1.16 参考書

FORTRAN の文法や利用法等、とてもこのテキストでは書ききれないので、以下に掲げるような参考書で自習して欲しい。

浦 昭二: FORTRAN 入門 (3 訂版) (培風館)

森口繁一：JIS FORTRAN 入門 (第 3 版) 上・下 (東大出版会)

4.2 FORTRAN プログラムのコンパイルと実行

プログラム sphere.f は無事ファイルとして作成されたとする。完成した FORTRAN プログラムをどうやって機械語に翻訳させるか、コンパイルと呼ばれる作業の手順を説明しよう。先ず、マウスを移動させて、ターミナルウィンドウをアクティブにしよう。そして、ターミナルウィンドウで、

```
% f77 -o sphere sphere.f
% sphere
```

と入力しよう。

```
% sphere
```

と入力してもエラーメッセージが表示されてしまう場合には、

```
% ./sphere
```

と入力して見よう。ターミナルウィンドウは、この様に、計算機に対するコマンドを一つ一つ入れていくためのウィンドウである。

ターミナルウィンドウは、必要に応じて、複数開いておくと便利である。ターミナルウィンドウを新しく開くには、マウスをウィンドウ以外の場所に移動させて、そこで、左ボタンを押す。すると、メニューが出現するが、その中の kterm の部分が反転するまでマウスを押したまま下に動かし、そ

[6] http://www.astron.s.u-tokyo.ac.jp/~shibahas/sphere6.f

こでマウスのキーを離す。暫くすると、画面には格子状のものが表示され、マウスと連動する。好みの場所に移動させて、マウスの左ボタンを押して離すと、その場所に新しいターミナルウィンドウが開かれる。

ターミナルウィンドウをいくつも開くと、計算機のメモリー等を浪費する事になるので、必要な数だけ開くように心掛けよう。

タイトルバーの部分にマウスのカーソルを入れて、中ボタンを押しながらマウスを動かすと、ウィンドウは格子状になり、マウスの動きと連動する。適当な場所にウィンドウを引きずって、そこでボタンを離すと、ウィンドウが再現する。こうして、ウィンドウの位置を変えることができる。

タイトルバーの右端の四角の部分にマウスカーソルを入れ、左ボタンを押しながら動かすと、マウスの動きに合わせて、格子状のものが伸び縮みをする。格子が適当な大きさになったところでボタンを離すと、ウィンドウの大きさが変化して再現する。こうして、ウィンドウのサイズを変更することができる。

f77 コマンドが sphere.f をコンパイルし、-o というオマジナイがついているので、更に sphere という実行ファイルを作成させる。計算の実行は sphere それ自身がコマンド名になる。

前の例題 sphere2 で、メインプログラムとサブルーチンの二つの別々のファイルとして作ったとし、メインプログラムのファイル名を sphere2.f とし、サブルーチンのファイル名を calculation.f としよう。そういった時、それぞれを別々にコンパイルして、オブジェクトファイルを融合させて、実行ファイルを作るには、次のようにする。

```
% f77 -c -o calculation.o calculation.f
% f77 -o sphere2 calculation.o sphere2.f
% sphere2
```

サブルーチンをコンパイルする時、-c -o のオマジナイを付けておく。結果として、.o の拡張子の付いたオブジェクトファイルが作られる。メインプログラムをコンパイルする時、後ろにこのオブジェクトファイル名を付けておく。この方法により、良く使うサブルーチンのオブジェクトファイルをいちいちコンパイルする手間を省いたり、プログラムのチェックの効率を上げたりする事が出来る。外部関数 FUNCTION 文もサブルーチンと同じように、メインプログラムとは別にコンパイルしておき、そのオブジェクトファイルとメインプログラムのオブジェクトファイルを結合して、実行ファイルを作る事が出来る。

4.3　今週 (と来週) の目標

1. 資料にあるプログラム sphere, sphere2, ... を入力 (画面からカットアンドペーストしたり、ファイルをコピーしてもいいが、書き写してみると何をしているか少し分かると思うので最初は書き写してみるのがいいと思う) して、コンパイル、実行してみる。

2. 多少複雑な計算をするプログラムを作る。例えば、2 次方程式

$$ax^2 + bx + c = 0$$

の係数 a, b, c を入力して、解の公式を使って答を計算、出力するプログラムを作成してみる。

第 5 章　計算機の更なる活用

5.1　計算グラフィックス入門

ここでは、FORTRAN で計算を行い、その計算結果を GNUPLOT を使ってグラフに表わすことをマスターしよう。レポートとして提出する際には、グラフも LaTeX に取り入れよう。

5.1.1　問題：黒体輻射

熱平衡にある完全黒体の物体から放射される光を黒体輻射と呼ぶ。単位面積から単位立体角に放射される黒体輻射のエネルギー $B(T)$ は、Planck の輻射式

$$B(T)d\lambda = \frac{2hc^2}{\lambda^5}\frac{1}{\exp(hc/\lambda kT)-1}d\lambda \qquad (\mathrm{J\cdot m^{-2}\cdot s^{-1}\cdot str^{-1}}) \tag{5.1}$$

に従う。但し、ここで、h は Planck 定数、k は Boltzman 定数、c は光速を表す。輻射エネルギー分布を $T = 3\mathrm{K}$、$300\mathrm{K}$、$3,000\mathrm{K}$、$30,000\mathrm{K}$ について、計算し、図示せよ[1] 。$hc/\lambda \gg kT$ の場合、輻射エネルギー分布は Wien の法則

$$B(T)d\lambda \simeq \frac{2hc^2}{\lambda^5}\exp(-hc/\lambda kT)d\lambda \tag{5.2}$$

で近似される事、逆に $hc/\lambda \ll kT$ の場合には、輻射エネルギー分布は Rayleigh-Jeans の法則

$$B(T)d\lambda \simeq \frac{2kTc}{\lambda^4}d\lambda \tag{5.3}$$

で近似される事を確かめよ。また、黒体輻射のエネルギー強度が最大になる波長 $\lambda_{\max}$ は、Wien の変位則

$$\lambda_{\max}T = 2.897\cdot 10^{-3} \quad \mathrm{m\cdot K} \tag{5.4}$$

を満たす事を確かめよ。ただし、$h = 6.6260755\times 10^{-34}\mathrm{J\cdot s}$, $k = 1.380658\times 10^{-23}\mathrm{J\cdot K^{-1}}$, $c = 2.99792458\times 10^{8}\mathrm{m\cdot s^{-1}}$ とせよ。

5.2　データベースの活用：HR 図を描く

天文学・宇宙科学は、装置や予算が大きいという意味で巨大科学である。世界有数の大望遠鏡の建設や衛星の打ち上げには巨額の費用がかかり、研究機関毎にそのような施設を持つことはできな

[1] GNUPLOT で図 5.1.1 の様に、λ や B_λ を書く方法は、章の初めに紹介したホームページを参照せよ。

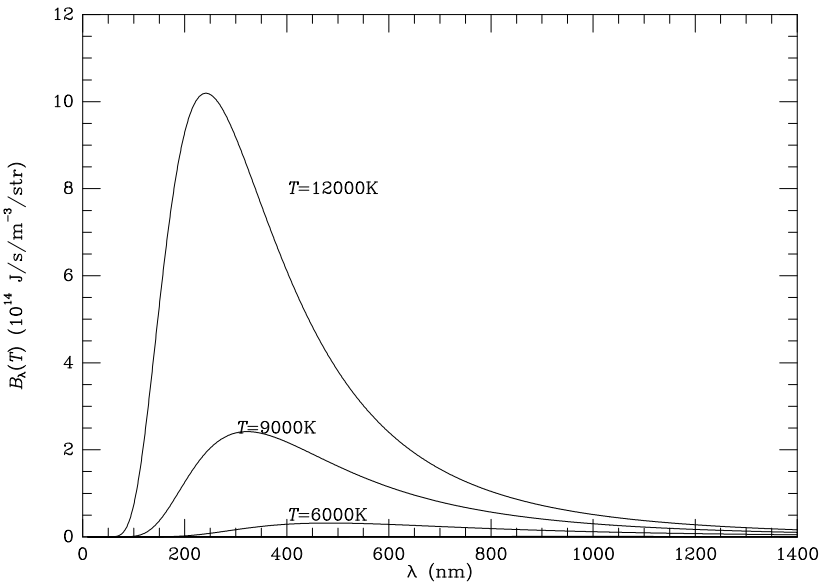

図 5.1: 黒体輻射のプランク関数。

い。そこで、これらの装置は、特定の研究者・研究機関によってだけ使われるというのでなく、多かれ少なかれ共同利用に供せられる。そのような場合、取得されたデ- タは、一定の期間は、取得した研究者に専有的に使われるが、しかる後は、運営している機関によりアーカイブとして保存され、一般の研究者に公開される、というのがよくあるルールである。これらは、オンラインで公開されているものが多い。こうした天文・宇宙科学の特質からか、天文学・宇宙科学の、ネットワ- クでの情報公開の量とサービスは、他の分野に比べても、充実したものが多いと思う。ここでは光学的に観測される約 9000 個の星の情報が格納された The Bright Star Catalogue (BSC4)[2] を用いて、HR 図を描いてみよう。

5.2.1 書式

この BSC4 カタログは、一つの恒星に対して、表 5.1 のような情報が各行の指定された位置に、指定された形式で与えられている。

但し、格納形式の An は、n 文字分の文字データを意味し、In は、n 桁の整数データである事を意味し、F$m.n$ は、小数点以下 n 桁で且つ小数点と符号も含めて m 桁の実数である事を意味している $(m \geq n+3)$。尚、書式としては、他に 1.25×10^{-8} を `1.25E-08` と表すような E$m.n$ という形式

[2] http://www.astron.s.u-tokyo.ac.jp/~shibahas/BSC4.dat

表 5.1: BSC4 カタログの表記内容

項目	格納形式	行中の位置
HR 番号	I4	1 – 4
名前	A7	8 – 14
赤経 (時)	I2	74 – 75
赤経 (分)	I2	77 – 78
赤経 (秒)	F4.1	80 – 83
赤緯 符号	A1	85
赤緯 (度)	I2	86 – 87
赤緯 (分)	I2	89 – 90
赤緯 (秒)	I2	92 – 93
実視等級	F5.2	109 – 113
スペクトル型	A2	137 – 138
視差	F5.3	170 – 174

がある。整数部は -38 から $+38$ までの整数、仮数部は小数点以下 n 桁で、且つ `E-08` という表記も含めて全部で m 文字である事を意味している $(m \geq n + 5)$。

5.2.2 FORMAT 文

このカタログを読んで、HR 図を描いてみよう。星の実視等級を m、星までの距離を d(パーセク) とすると、絶対等級 M は

$$M = m - 5.0 \log \frac{d}{10} \tag{5.5}$$

表 5.2: 入出力の表記

型	表記	意味
文字	An	n 文字
整数	In	n 桁整数 (含符号)
実数	F$m.n$	小数点以下 n 桁、少数と符号も含め全体で m 文字
	E$m.n$	小数点以下 n 桁、少数と符号も含め全体で m 文字 (単精度)
	D$m.n$	小数点以下 n 桁、少数と符号も含め全体で m 文字 (倍精度)
空白	X	
行替	/	

となる。星までの距離 d(パーセク) は、年周視差を π(秒)、地球の公転長半径を R(天文単位) とすると、

$$d = R/\pi \tag{5.6}$$

となる。そこで、BSC4 カタログの視差と実視等級から、絶対等級を求め、それをスペクトル型に対してプロットしてやればよい。実視等級は数値データだから、グラフ化での取り扱いも容易だが、スペクトル型は文字型データだから、取り扱いには工夫が必要だ。

格納形式が決まっているデータの読み込みや、決まった形式での書き込みのために、FORMAT 文なるものが用意されている。以下に掲げるのは、BSC4 カタログを BSC4.DAT として、読み込み、年周視差と実視等級から絶対等級を求め、また、HR 図にグラフ化する際に便利なようにスペクトル型を数値化した上で別のカタログに別の様式で格納するためのプログラム[3] である。

```
      program main

      implicit NONE
      integer N,m,maxindex
      parameter(N=50000,m=7,maxindex=10)
      integer j,number_of_stars,jj,jjj
      real Vmag,parallax
      real Abs_mag(N),Sp_index(N)
      character*1 Sp_type,Sp_sub,s(m),sub(maxindex)

      data s/'O','B','A','F','G','K','M'/
      data sub/'0','1','2','3','4','5','6','7','8','9'/
c-------------------------------------------------------------------------------
      number_of_stars=0

      open(unit=1,file='BSC4.DAT',status='OLD')
      open(unit=2,file='STARS.DAT',status='NEW')

! Read the BSC catalogue:
      do j=1,N
        read (1,100,end=20) Vmag,Sp_type,Sp_sub,parallax
        if(parallax.gt.0.)then
          do jj=1,m
            if(Sp_type.eq.s(jj))then
              do jjj=1,maxindex
                if(Sp_sub.eq.sub(jjj))then
```

[3] http://www.astron.s.u-tokyo.ac.jp/~shibahas/hr-diagram.f

```
                number_of_stars=number_of_stars+1
                Abs_mag(number_of_stars)=Vmag+5.0
     *                                  +5.0*log10(parallax)
                Sp_index(number_of_stars)=float(10*(jj-1) + (jjj-1))
                write(2,220) j,Vmag,
     *                       Sp_index(number_of_stars),
     *                       Abs_mag(number_of_stars)
              end if
            end do
          end if
        end do
      end if
     end do
  20 write(6,131) number_of_stars

     close(1)
     close(2)

 100 format(108x,f5.2,23x,2a1,31x,f5.3)
 131 format(////'   there are ',i6,' stars in the catalog'//)
 220 format(i7,3(3x,f5.2))

     stop
     end
```

READ 文が今までのように read(*,*) や read(1,*) ではなく、read(1,100) となっている事に注目しよう。最初の 1 は、ユニット番号 1 番から読み込む事を意味している事は、もう判ると思う。次の 100 は、100 番の文番号のついた FORMAT 文の形式で読む事を意味している。100 番の文番号の文は、確かに FORMAT 文になっており、表 5.1 の形式で読み込もうとしている事が判るであろう。READ 文で、end=20 とあるのは、データを読み終えたら、20 番の文番号に飛ぶ事を意味している。データを幾つ読むのか、予め定まっていない場合に、使われる手段であることは以前に説明した。この 100 番の文番号のついた FORMAT 文では、HR 番号や赤経赤緯等、今の目的に要しないデータを読み飛ばすために、nX で n 個分の空白データを読んでいる。WRITE 文も、220 番の FORMAT 文を従えており、実数形式で実視等級、スペクトル型に対応する数値、絶対等級を書き込もうとしている。220 番の FORMAT 文での 3x は、３個の空欄を空ける事を意味している。文番号は、１文字目から 5 文字目までに書く事が約束となっている。

5.2.3　DATA 文

上のプログラムでは、スペクトル型の文字部と細分類の数字部分をそれぞれ 1 文字データ Sp_type と Sp_sub として読み込み、型に応じて数値を当て嵌める事によって、スペクトル型を数値化して、グラフ化する際の x-座標に対応させようとしている。Sp_type が O であれば、1 の位、B であれば 10 代、A であれば 20 代をあてがう。そこで、7 元の配列 s(7) を s(1)='O'、s(2)='B'、···、s(7)='M' と定義して、Sp_type がどの s(n) に一致するかを調べれば良い。ところで、プログラム中で中身が変わる事のない 7 元の配列 s(7) を、s(1)='O'、s(2)='B'、···、s(7)='M' と定義式をいちいち書くのは煩わしいし、プログラムもみにくくなる。上のプログラムでは、代わりに、data s/'O','B',···'M'/ のような文で与えている。このような文を DATA 文という。これは、1 文字の文字型データ配列 s(7) を定義し、s(1)='O'、s(2)='B'、s(3)='A'、··· s(7)='M' と与える役割を果たしている。その次の行の data sub/'0','1',···'9'/ も同様の役割を果たす。

5.2.4　問題：HR 図を描く

BSC4 カタログを読んで、HR 図を描け。

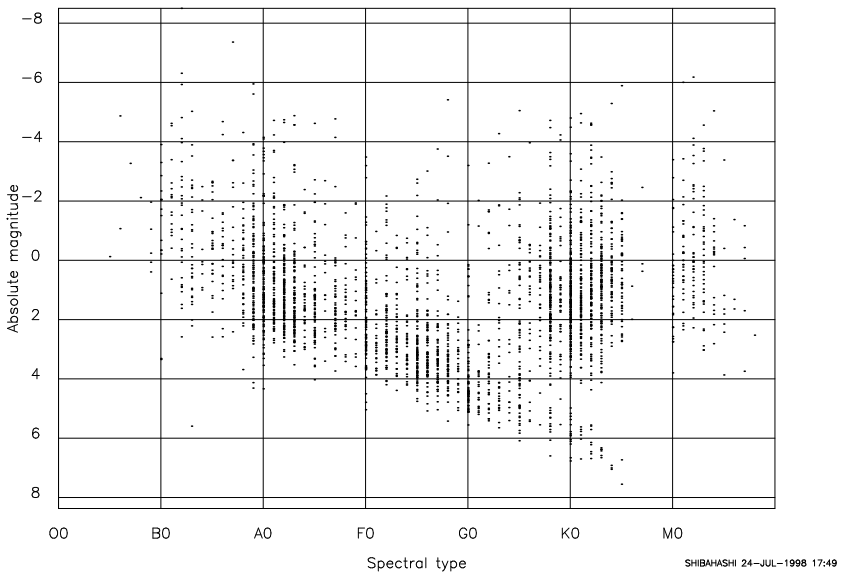

図 5.2: BSC4 カタログ上の星の HR 図。

5.3 データベースの活用：銀河の三次元分布図を描く

天体の天球上の二次元分布は、直接見ることができるが、距離をも考慮した三次元分布を見る事は容易ではない。そもそも天体までの距離を測る事は、天文学の歴史を顧みれば判るように、太陽系とか銀河系といった基本的概念を作り上げ、宇宙の中での我々の位置付けを確定する上で、天文学上の最も基本的且つ最も重要なテーマであった。この事情は、今も、また、今後も変わりない。銀河系の大きさが確定した現在、次に着目するのは、銀河がどのように分布しているかということである。銀河までの距離を測るのは、個々の星が分解して見えるほどの近くの銀河では、セファイドの周期光度関係を使う。更に遠方の銀河は、既に距離が決定された近くの銀河を参照に、距離と相関のある、見かけの物理量の様々な関係を使って、距離を決定する。一方、スペクトル線のドップラー遷移で測定して銀河の視線速度が測定されるが、両者の間には、遠方の銀河ほど後退速度が大きいという法則（ハッブルの法則）がある。

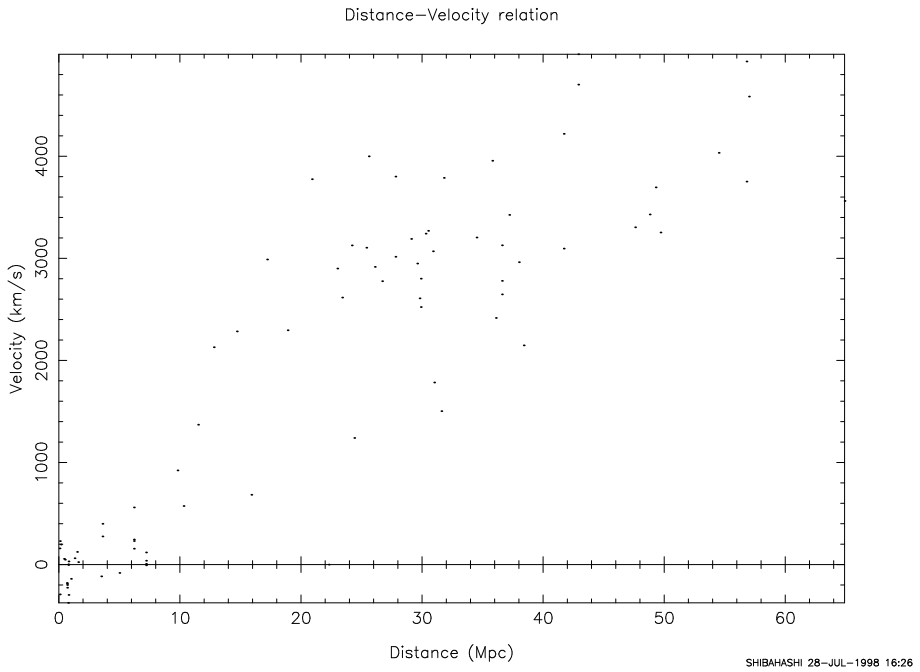

図 5.3: 銀河の距離速度関係。

更に遠方銀河の距離は、この関係を利用して、スペクトルの赤方偏移で測るしか手段がない。赤方偏移 ($z = \Delta\lambda/\lambda_0$) は、宇宙膨張によって生ずるものであるが、これを銀河の後退速度 ($v\mathrm{km}\cdot\mathrm{s}^{-1}$) と見做せば、$z$ と v の対応は

$$\frac{v}{c} = \{(1+z)^2 - 1\}/\{(1+z)^2 + 1\} \tag{5.7}$$

(但し、c は光速度で単位は $\mathrm{km}\cdot\mathrm{s}^{-1}$) で与えられる。$z < 0.05$ のときは、近似的に $v = cz$ を用いて

表 5.3: CfA カタログの表記内容

項目	格納形式	行中の位置
名前	A11	1 – 11
赤経 (時)	I2	12 – 13
赤経 (分)	I2	14 – 15
赤経 (秒)	F4.1	16 – 19
赤緯 符号	A1	20
赤緯 (度)	I2	21 – 22
赤緯 (分)	I2	23 – 24
赤緯 (秒)	I2	25 – 26
B (等級)	F5.2	27 – 31
後退速度	I5	32 – 36
後退速度誤差	I3	37 – 39
その他の情報		40 – 69
独立に求めた距離 (Mpc)	F4.2	70 – 73
その他の情報		74 – 120

も誤差は無視できる。ハッブル定数を $H_0 = 75\mathrm{km \cdot s^{-1}Mpc^{-1}}$ と仮定し、宇宙が平坦であるとして、何年前に銀河を発した光を我々が観測しているかに基づいて距離 d (光年) を以下のように計算することができる。

$$d = 3.26 \times 10^6 \frac{2}{3} \frac{c}{H_0} \{1 - (1+z)^{-3/2}\} \tag{5.8}$$

ハーバード大学のグループは、銀河の赤方偏移を隈無く測定するプロジェクトを行ない、宇宙地図を描く事を行った。銀河系から距離約 1 億光年以内にある銀河は、乙女座銀河団を中心とし、薄い円盤部とそれを取り巻くハロー成分とから成る局部超銀河団を形成している。この他に、かみのけ、ヘルクレス、うお-ペルセウス、うみへび-ケンタウルス等の超銀河団が同定されている。ハーバード大学のサーベイの結果、 1 億光年以上のスケールにわたって銀河の殆どない領域 (ボイド) のあること、超銀河団とボイドが互いに入り交じって大規模な構造を成している事が明らかになった。

このハーバード大学の赤方偏移カタログは、一つの銀河に対して、表 5.3 のような情報が各行の指定された位置に、指定された形式で与えられている。

このカタログを読んで、宇宙地図を描いてみよう。以下に掲げるのは、CfA カタログを `VELOCITY.DAT` [4] として、読み込み、赤経、赤緯を度の単位に直した上で別のカタログに別の様式で格納するためのプログラム[5] である。`READ` 文が `read(1,100)` となっている事に注目しよう。最初の `1` は、ユニット番号 1 番から読み込む事を意味している事、次の `100` は、`100` 番の文番号のついた `FORMAT` 文の形式で読む事を意味している事は、もう判ると思う。。`100` 番の文番号の文は、確かに `FORMAT` 文に

[4] http://www.astron.s.u-tokyo.ac.jp/~shibahas/VELOCITY.DAT
[5] http://www.astron.s.u-tokyo.ac.jp/~shibahas/CfA-read.f

なっており、表 5.3 の形式で読み込もうとしている事が判るであろう。同様に、WRITE 文も、220 番の FORMAT 文を従えており、実数形式で赤経、赤緯を書き込もうとしている。220 番の FORMAT 文での 3x は、３個の空欄を空ける事を意味している。文番号は、１文字目から 5 文字目までに書く事が約束となっている。

```
      program main

      implicit NONE
      integer N
      parameter(N=50000)
      integer j,RA_hr,RA_min,dec_deg,dec_min,dec_sec,
     *        vel,number_of_galaxy
      real RA_sec,RA,dec,distance,sign
      character*1 dec_sign,minus

      data minus/'-'/

      number_of_galaxy=0

      open(unit=1,file='VELOCITY.DAT',status='OLD')
      open(unit=2,file='GALAXY.DAT',status='NEW')

      do  j=1,N
        read (1,100,end=20) RA_hr,RA_min,RA_sec,
     *                      dec_sign,dec_deg,dec_min,dec_sec,
     *                       vel,distance
        RA=15.*(float(RA_hr)+float(RA_min)/60.+ RA_sec/3600.) !in units of deg

        if(dec_sign.eq.minus)then
          sign=-1.
        else
          sign=1.
        end if
        dec=(float(dec_deg)+float(dec_min)/60.+float(dec_sec)/3600.)
     *         *sign

        write(2,220) j,RA,dec,vel,distance
        number_of_galaxy=number_of_galaxy+1
      end do
```

```
 20 write(6,131) number_of_galaxy

    close(1)
    close(2)

100 format(11x,i2,i2,f4.1,a1,3i2,5x,i5,33x,f4.2/)
131 format(////'   there are ',i6,' galaxies in the catalog'//)
220 format(i7,3x,2f15.5,3x,i5,3x,f5.2)

    stop
    end
```

5.3.1　問題：ハッブルの法則

CfA カタログを使って、距離と後退速度が独立に測られている銀河に対して、両者がどのような関係になっているかを距離-後退速度図を描いて示せ。

5.3.2　問題：宇宙地図

CfA カタログを使って、ある赤緯帯の銀河を赤経・距離の二次元マップ上にを描け。

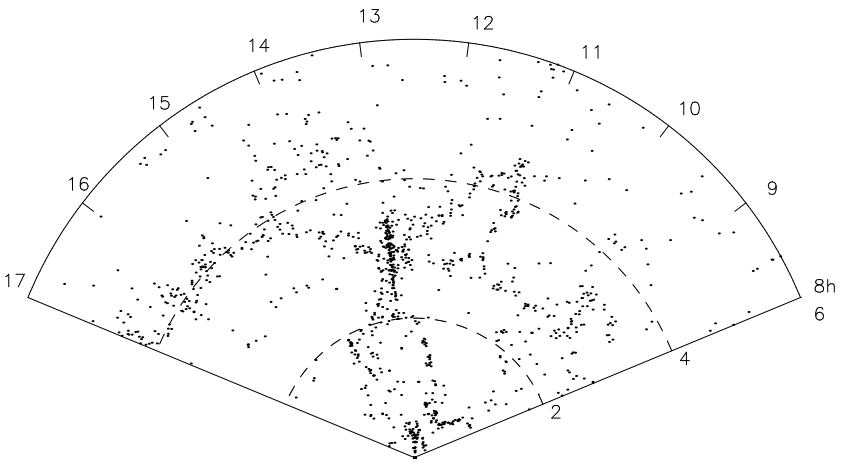

図 5.4: 赤経 8h-17h、赤緯 $+26.5°$-$+38.5°$ の帯状領域にある 15.5 等級より明るい銀河の空間分布。赤緯については、これ以上区別せず、同一平面上に描いてある。距離の単位は億光年。