1　実習テキスト 基礎編

1.1　1次元パッケージの説明

パッケージは プログラムモジュール と 基本課題モジュール でできています。例えばプログラムモジュールの一つである改良 Lax-Wendroff法で流体方程式をとくためのモジュールはディレクトリ (“hdmlw”) にあり、次のファイルが含まれます。

```
# ls hdmlw/
Makefile     mlw_h_c.f    mlw_m.f      mlwartv.f    mlwhalf.f
README       mlw_h_g.f    mlw_m_g.f    mlwfull.f
mlw_h.f      mlw_h_gc.f   mlw_m_gc.f   mlwgrd.f
```

基本課題モジュールは数値シミュレーションを始める人に適している課題 (例えば、衝撃波管問題、点源爆発など) を集めたものです。一つ一つの課題が別パッケージになっており、“md” で始まるディレクトリに入っています。例えば衝撃波管問題 (流体) のパッケージは以下のディレクトリにあります。

```
# ls md_shktb
Makefile             bnd.f                rddt.pro
README               main.f               shktb_analytic.pro
Readme.pdf           model.f
Readme.tex           pldt.pro
```

各モジュールは Fortran 言語を用いて書かれています。Fortran は数値シミュレーションの分野のプログラミングでもっとも用いられており、Fortran プログラムファイルの名前の最後は .f です。基本課題モジュールの説明は README と Readme.pdf とにかかれています。どんなプログラムモジュールや基本課題モジュールが準備されているかは章末を参考にして下さい。

1.2 プログラムのコンパイル (make)

次にプログラムをコンパイルする方法について説明します。パッケージの一番上のディレクトリで make を実行します。するとプログラムモジュールのコンパイルができます。

```
# make
```

すると以下のようになり、コンパイルを開始し、終了します。

```
f77 -I../  -c -o cflhd.o cflhd.f
cflhd.f:
        cflhd:
(後略)
```

1.3 プログラムの実行 (make)

プログラムの実行は基本課題ディレクトリに移動して、make することで実行します。ここでは衝撃波管問題 (流体) を動かしてみましょう。

```
# cd md_shktb
# make
```

すると以下のようになり、計算結果のファイル (out.cdf) を出力して終了します。

```
f77 -I..  -c -o main.o main.f
main.f:
 MAIN:
(中略)
a.out
  write    step=       0 time= 0.000E+00 nd =  1
  write    step=     210 time= 0.500E-01 nd =  2
  write    step=     409 time= 0.100E+00 nd =  3
  write    step=     573 time= 0.142E+00 nd =  4
  stop     step=     573 time= 0.142E+00
   ### normal stop ###
 Note: IEEE floating-point exception flags raised:
    Inexact;  Underflow;
 See the Numerical Computation Guide, ieee_flags(3M)
```

他のパッケージを動かす場合も同じようにおこないます。make の実行について詳しくは応用編で説明します。

1.4 結果の表示

結果の表示にはIDLといった可視化プログラムを利用します。IDLは数値シミュレーション結果を可視化をするのによく用いられます。(※idlは高価なソフトウェアなので、今回の実習で全員が起動できるだけのライセンスがありません。ライセンスが足りない場合、idlはデモモードで起動します。デモモードでは7分間は同様に動きます。).

1.4.1 IDLの起動 (idl)

まずはidlを実行してみましょう

```
# idl
```

すると以下のようになり、IDLが起動します。

```
IDL Version 5.3 (sunos sparc). (c) 1999, Research Systems, Inc.
Installation number: 70271.
Licensed for use by: CHIBA-UNIV

IDL>
```

1.4.2 データ読み込み (.r rddt)

データの読み込みには rddt.pro というプログラムを用います。以下のように入力してみて下さい。.rはrunを意味します。

```
IDL> .r rddt
```

1.4.3 データ読み込み (.r pldt)

データの表示には pldt.pro というプログラムを用います。以下のように入力してみて下さい。

```
IDL> .r pldt
```

1.4.4 IDLの終了 (end)

endを入力するとIDLを終了することができます。デモモードでは7分後に自動的に終了します。

```
IDL> end
```

1.5 実習課題

1次元パッケージを使ってみなさい。

1. 基本課題「流体衝撃波管問題」「球対称点源爆発問題」「MHD衝撃波管問題」を実行し、IDLで可視化せよ (`.r rddt`, `.r pldt`)。
2. 基本課題「磁力管 Alfvén 波管伝播 (スピキュール生成) 問題」を実行し、IDLで可視化せよ (計算に時間がかかるので注意、昼食前に動かすのが吉)。

モジュール一覧

プログラムモジュール

```
nc/      netCDF フォーマットでデータ出力するためのモジュール
common/  計算で使うさまざまな共通ルーチンを集めたモジュール
bc/      境界条件を定義するためのプロシジャ
hdmlw/   流体力学方程式・MHD 方程式を改良 Lax-Wendroff ＋人工粘性法で解くためのモジュール
hdroe/   Roe 法で流体計算をするためのモジュール
cndsor/  熱伝導を陰解法（時間精度 1 次：行列反転は Red Black SOR 法）で解くためのモジュール
htcl/    放射冷却・静的加熱を陽解法で解くためのモジュール
```

基本課題モジュール

```
md_shktb/    流体衝撃波菅問題     [流体]
md_sedov/    球対称点源爆発（Sedov 解）問題   [非一様断面積、流体]

md_mhdshktb/ MHD 衝撃波菅問題   [MHD]
md_spicule/  磁力管 Alfven 波伝播（スピキュール生成）問題 [非一様断面積、MHD]

md_cndtb/    単純熱伝導問題   [熱伝導]
md_cndsp/    球対称単純熱伝導問題   [非一様断面積、熱伝導]
md_thinst/   熱不安定問題     [冷却加熱]

md_flare/    フレアループ問題   [非一様断面積、流体、熱伝導、冷却加熱]
```

2　実習テキスト 応用編

2.1　基本課題パッケージの変更

基本課題のモデルパラメータを変更するには、基本課題パッケージの中にある model.f や main.f を変更します。モデルの簡単な変更は model.f でおこないます。model.f を開いてみるとわかるようにこのプログラムでは初期の密度 (ro)、速度 (vx, vy)、圧力 (pr)、磁場 (bx, by) の初期の分布を定めています。

2.1.1　メッシュ数の設定 (ix, main.f)

ここでは簡単なモデルパラメータの変更をおこなってみましょう。まずは基本課題の衝撃波管でメッシュ数 (ix) をかえてみましょう。メッシュ数は main.f を変更します。メッシュ数をかえることで数値計算の分解能をあげることができます。一度、衝撃波菅を問題をおこなったあとは

```
# cd md_shktb
# ls
Makefile            bnd.f               model.o
README              bnd.o               out.cdf
Readme.pdf          main.f              pldt.pro
Readme.tex          main.o              rddt.pro
a.out*              model.f             shktb_analytic.pro
```

になっています。このディレクトリにある main.f をエディタで開き、以下 (1–5 行目) を見てください。

```
c=======================================================================|
c     array definitions
c=======================================================================|
      include ‘‘precision.h’’
      parameter (ix=1001)
```

初期のメッシュ数は 1001 です。これを 2 倍の 2001 にしてみましょう。一般にメッシュ数を 2 倍にすると同じ時刻まで計算をすすめるためには計算時間は約 2 倍になります。

実習課題

上記手順にそって、メッシュ数を変更してみなさい。

2.1.2 断熱比 γ の変更 (gm, model.f)

次に基本課題の点源爆発で断熱比 γ (gm) をいじってみましょう。断熱比は model.f を変更します。一度、点源爆発をおこなった人は、

```
# cd md_sedov/
# ls
Makefile   a.out*     bnd.o      main.o     model.o    pldt.pro
README     bnd.F      main.f     model.f    out.cdf    rddt.pro
```

となっていると思います。このディレクトリにある model.f をエディタで開いて見て下さい。そして次の文を探しましょう (13–16 行目)。

```
c-----------------------------------------------------------------------|
c   parameters
c-----------------------------------------------------------------------|
      gm=5./3.
```

初期に断熱比は $5./3.$ になっています。これを別の数字に例えば $7./5.$ に変更してみて下さい。

```
      gm=7./5.
```

make を実行し、idl を立ち上げ、何が変わったか比較してみましょう。

2.1.3 最終ステップ数、出力の設定 (tend, dtout, main.f)

続けて、最終時刻 (dend) と出力ファイルの時間間隔 (dtout) を変更してみましょう。main.f をエディタで開いて見て下さい。そして次の文を探しましょう (43–49 行目)。

```
c-----------------------------------------------------------------------|
c   time control parameters
c     nstop : number of total time steps for the run

      tend=3.0
      dtout=0.5
      nstop=1000000
```

ここでは最終時刻を 5.0 に変更し、出力の時間間隔を 1.0 にしてみましょう。

実習課題

上記手順にそって、断熱比、出力の変更をしてみなさい。次節を参考にしてアニメーション表示もしてみなさい。

2.2 結果のアニメーション表示 (.r anime)

idlではアニメーションの表示をすることもできます。anime.proというプログラムを用います。idlを立ち上げ、データを読み込み、anime.proを実行してみましょう。

```
# idl
(中略)
> .r rddt
> .r anime
```

3 参考：実行結果編

実行例

メッシュ数を2001として、衝撃波管問題（流体）を動かした結果は以下です。

```
# make
f77 -I..  -c -o main.o main.f
main.f:
 MAIN:
(中略)
a.out
  write    step=       0 time= 0.000E+00 nd =  1
  write    step=     409 time= 0.500E-01 nd =  2
  write    step=     804 time= 0.100E+00 nd =  3
  write    step=    1132 time= 0.142E+00 nd =  4
  stop     step=    1132 time= 0.142E+00
   ### normal stop ###
 Note: IEEE floating-point exception flags raised:
    Inexact;  Underflow;
 See the Numerical Computation Guide, ieee_flags(3M)
```

実行例

球対称点源爆発（Sedov解）問題を動かした結果は以下です。

```
# cd md_sedov
# make
f77 -I..  -c -o main.o main.f
main.f:
 MAIN:
(中略)
a.out
  write    step=       0 time= 0.000E+00 nd =  1
  write    step=     612 time= 0.501E+00 nd =  2
  write    step=    1034 time= 0.100E+01 nd =  3
  write    step=    1407 time= 0.150E+01 nd =  4
  write    step=    1751 time= 0.200E+01 nd =  5
  write    step=    2075 time= 0.250E+01 nd =  6
  write    step=    2384 time= 0.300E+01 nd =  7
  stop     step=    2384 time= 0.300E+01
   ### normal stop ###
 Note: IEEE floating-point exception flags raised:
    Inexact;  Underflow;
 See the Numerical Computation Guide, ieee_flags(3M)
```

実行例

比熱比 $\gamma = 7./5.$ として、球対称点源爆発（Sedov 解）問題を動かした結果は以下です。

```
# make
f77 -I..  -c -o main.o main.f
main.f:
 MAIN:
(中略)
a.out
  write    step=       0 time= 0.000E+00 nd =  1
  write    step=     750 time= 0.501E+00 nd =  2
  write    step=    1330 time= 0.100E+01 nd =  3
  write    step=    1861 time= 0.150E+01 nd =  4
  write    step=    2361 time= 0.200E+01 nd =  5
  write    step=    2840 time= 0.250E+01 nd =  6
  write    step=    3303 time= 0.300E+01 nd =  7
  stop     step=    3303 time= 0.300E+01
   ### normal stop ###
 Note: IEEE floating-point exception flags raised:
    Inexact;  Underflow;
 See the Numerical Computation Guide, ieee_flags(3M)
```

実行例

最終時刻を 5.0、出力の時間間隔を 1.0 とし、比熱比 $\gamma = 7./5.$ として、球対称点源爆発（Sedov 解）問題を動かした結果は以下です。

```
# make
f77 -I..  -c -o main.o main.f
main.f:
 MAIN:
(中略)
a.out
  write    step=       0 time= 0.000E+00 nd =  1
  write    step=    1330 time= 0.100E+01 nd =  2
  write    step=    2361 time= 0.200E+01 nd =  3
  write    step=    3303 time= 0.300E+01 nd =  4
  write    step=    4191 time= 0.400E+01 nd =  5
  write    step=    5042 time= 0.500E+01 nd =  6
  stop     step=    5042 time= 0.500E+01
   ### normal stop ###
 Note: IEEE floating-point exception flags raised:
    Inexact;  Underflow;
 See the Numerical Computation Guide, ieee_flags(3M)
```