天体統合シミュレーション ソフトウェア CANS

2003年4月30日

第1章　はじめに

計算機と計算科学技術の発達に伴って、物理法則にしたがって時間発展する系の進化を計算機を用いて追跡するという数値実験が従来の理論と実験・観測に加わる第 3 の科学研究手法として確立してきました。天体現象は地上での実験が困難であることが多いため、計算機の中に仮想的な宇宙をつくって実験を行い、その結果を観測と比較することを通してモデルの妥当性を検証し、現象の物理機構を明らかにしていくという計算科学的手法が特に有効です。

我々のグループでは、科学技術振興事業団計算科学技術活用型のプロジェクトとして「宇宙シミュレーション・ネットラボラトリーシステムの開発」を実施し、高速ネットワークを活用した宇宙シミュレーションのバーチャルラボラトリーの構築を目標として研究開発を進めてきました。このラボラトリーの構成要素として開発されたのが、宇宙磁気流体現象のシミュレーション研究を支援するソフトウェアライブラリー CANS（Coordinated Astronomical Numerical Software) です。

1.1　CANS で何ができるか？

CANS を用いると、さまざまな宇宙磁気流体現象の数値シミュレーションを行い、シミュレーション結果を可視化することができます。宇宙は様々な活動性と多様性に満ちていますが、フレアやジェットをはじめとして、磁場と強く相互作用するプラズマのマクロな運動がエネルギー解放や質量輸送の起源になっている現象が数多くあります。CANS が扱うことができるのは、このようにマクロな流体・磁気流体方程式で記述できる現象です。

CANS では自己重力、磁力線方向に依存する非等方熱伝導、磁気拡散、輻射冷却などの物理過程を取り入れることができ、1 次元から 3 次元に至る種々の天体磁気流体現象の数値実験に適用できます。

1.2　CANS の特徴

CANS の最大の特徴は、シミュレーションコードに加えて、宇宙シミュレーションの典型的問題（基本課題）のシミュレーションモデル（初期条件、境界条件等)、推奨パラメータ、各モデルの解説、シミュレーション結果の動画、可視化・解析ツール等をセットにして公開している点です。基本課題のテーマは各種流体・磁気流体不安定性をはじめとして、衝撃波伝播、ジェット、フレア、星形成など多岐にわたっています。

シミュレーションコードだけを公開しても、それを使いこなすことは一般には困難で

す。CANS では、各種初期モデルのソースコードとその解説、入力パラメータのサンプル等が公開されていますので、これらを手がかりにして、利用者は容易にシミュレーションモデルを再構成し、新たなシミュレーションを実施することができます。

CANS ではシミュレーション結果データの入出力フォーマットとして netCDF 形式を採用しています。これにより、netCDF 形式をサポートしている各種可視化ソフトウェアを用いてシミュレーション結果を可視化することが可能です。CANS は、Web ブラウザを用いて遠隔地からシミュレーションを実行し、データを可視化することができる Web アプリケーション NetCANS にも拡張されています。NetCANS について詳しくは別のマニュアルをご覧ください。

1.3 CANS の構成

CANS は流体・磁気流体シミュレーションコードの共通プラットフォームと、各基本課題のモジュール及び解説ページから構成されます。磁気流体方程式に従う時間発展を解くシミュレーションエンジンとして、改良 Lax-Wendroff 法、近似的リーマン解法にもとづく Roe 法、CIP-MOCCT 法の 3 種類のエンジンがあり、エンジン部分だけを取替えてシミュレーションを実施することも可能です。CANS には、1 次元パッケージ、2 次元、3 次元パッケージに加えて並列計算機用に MPI を用いて並列化した CANS/MPI もあります。

以下では、まず 1 次元パッケージの使い方、1 次元基本課題について解説したのち、1 次元コードの各モジュールの相互関係、簡単なパラメータの変更方法等を説明します。続いて 2 次元、3 次元基本課題をとりあげます。

1.4 CANS Web ページ

以下のページで CANS で行なえる天体シミュレーションの概要を紹介しています。

```
http://www.astro.phys.s.chiba-u.ac.jp/netlab/astro
```

第2章　CANS 1Dパッケージの説明

CANS 1D 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 法で流体方程式を解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_ht.f       mlw_m3_g.f     mlw_m_g.f      mlwfull.f
README         mlw_ht_c.f     mlw_m3t.f      mlw_mt.f       mlwhalf.f
Readme.tex     mlw_ht_cg.f    mlw_m3t_c.f    mlw_mt_c.f     mlwsrcf.f
mlw_a.f        mlw_ht_g.f     mlw_m3t_cg.f   mlw_mt_cg.f    mlwsrch.f
mlw_h.f        mlw_m.f        mlw_m3t_g.f    mlw_mt_cgr.f
mlw_h_c.f      mlw_m3.f       mlw_m_c.f      mlw_mt_g.f
mlw_h_cg.f     mlw_m3_c.f     mlw_m_cg.f     mlw_rh.f
mlw_h_g.f      mlw_m3_cg.f    mlw_m_cgr.f    mlwartv.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のパッケージは” 以下のようになっています。

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

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います。上記のリストでファイルの拡張子は.f になっているものが、Fortran プログラムファイルです。

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。これらのドキュメントへのアクセスは HTML 形式によるドキュメント “htdocs” を web browser で開くと便利です。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、別紙のリストや Web ページを参考にして下さい。

2.1 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。パッケージのディレクトリ “cans1d” と “cansnc” の 2 箇所で make を実行します。“cans1d” で make するとプログラムモジュールから実行アーカイブファイル libcans1d.a、“cansnc” で make すると実行アーカイブファイル libcansnc.a をそれぞれ作成し終了します。

```
# cd cans1d
# make
        cd hdmlw; make
        f77 -O -c mlwfull.f
(中略)
        touch .update
```

```
# cd ../cansnc
# make
(後略)
```

cans1d や cansnc の上のディレクトリで make をおこなうと時間はかかりますが、cans1d、cans2d、cans3d、cansnc の全てのパッケージを一度にコンパイルします。

2.2 プログラムの実行 (make)

基本課題プログラムは基本課題モジュールのディレクトリに移動して、make することでコンパイルし、実行します。ここでは、衝撃波管問題 (md_shktb) を動かしてみましょう。

以下のように表示され、計算結果のファイル out.cdf を出力して終了します。

```
# cd md_shktb
# make
        f77 -O -c main.f
        f77 -O -c model.f
        f77 -O -c bnd.f
        f77 -O -c cipbnd.f
        f77 -o a.out main.o model.o bnd.o cipbnd.o -L..
            -lcans1d -L/usr/local/netcdf/lib -lnetcdf
        ./a.out
  write    step=       0 time= 0.000E+00 nd =  1
  write    step=      75 time= 0.505E-01 nd =  2
  write    step=     154 time= 0.100E+00 nd =  3
  write    step=     221 time= 0.142E+00 nd =  4
  stop     step=     221 time= 0.142E+00
   ### normal stop ###
```

2.3 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です。)

2.3.1 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 5.5 (sunos sparc). (c) 2001, Research Systems, Inc.
Installation number: XXXXX.
Licensed for use by: XXXXX

IDL>
```

2.3.2 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
```

2.3.3 データ表示 (.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
```

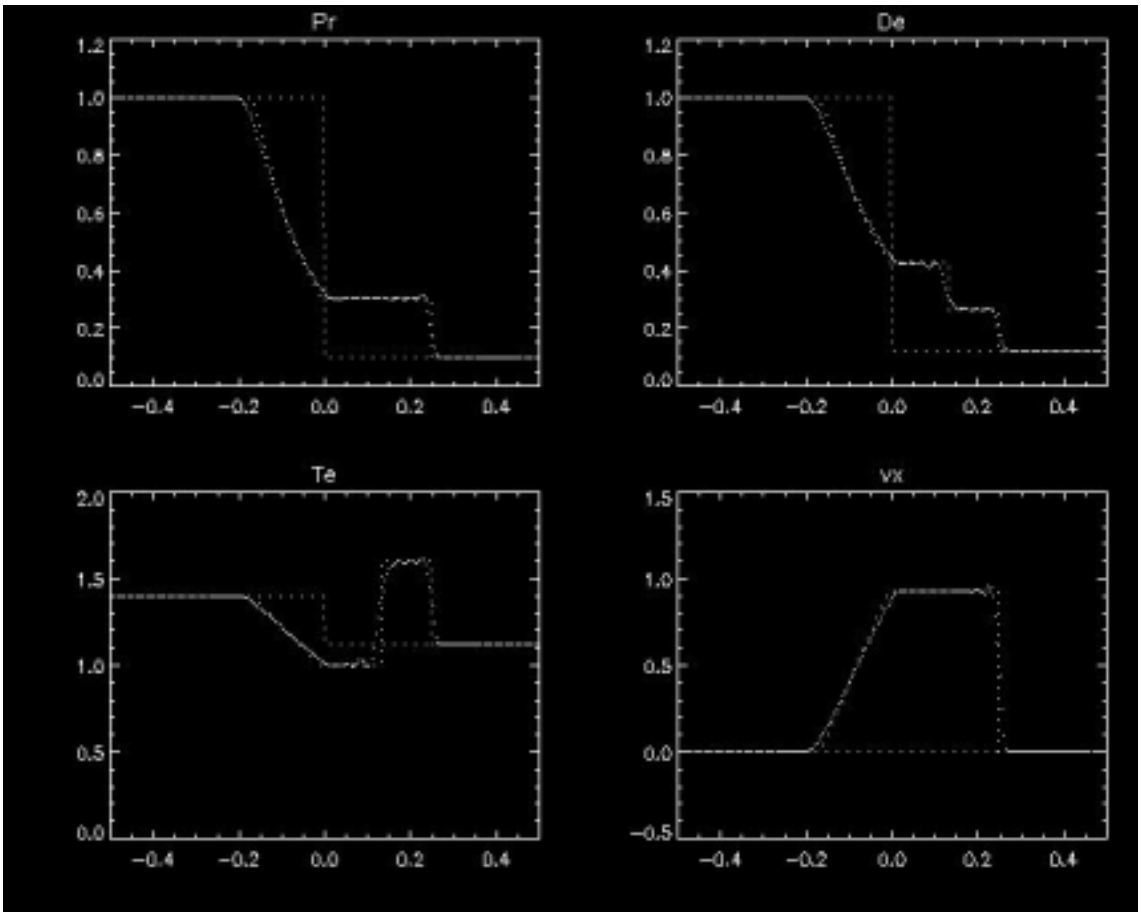

図 2.1: md_shktb の結果

2.3.4 データ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。データを読み込み、anime.pro を実行してみましょう。以下のように入力してみて下さい。

```
IDL> .r anime
```

2.3.5 IDL の終了 (exit)

exit 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
```

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「等温衝撃波管 (md_itshktb)」を実行し、IDL で rddt.pro と pldt.pro を用いて可視化せよ。
2. 基本課題「流体衝撃波管 (md_shktb)」を実行し、可視化せよ。
3. 基本課題「衝撃波生成 (md_shkform)」を実行し、anime.pro を用いて可視化せよ。
4. 基本課題「MHD 衝撃波管 (md_mhdshktb)」を実行し、可視化せよ。

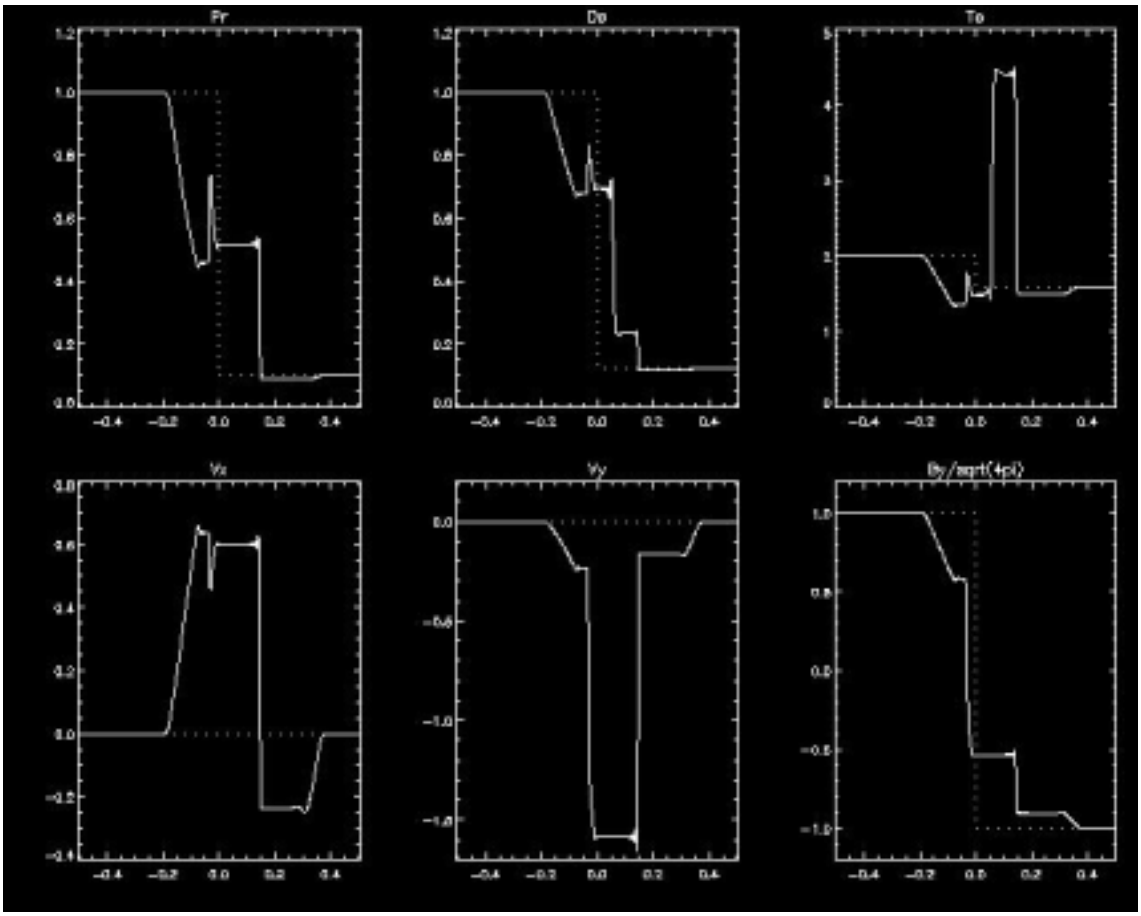

図 2.2: md_mhdshktb の結果

補足

- 基本課題を動かした後に Fortran プログラムを変更し、make すると、再コンパイルし、計算を実行します。この際、出力ファイル out.cdf を上書きします。必要な出力ファイルは、名前を out1.cdf など変更してとっておきましょう。
- オブジェクトファイル、出力ファイル out.cdf を消去したい場合は、make clean を実行してください。

2.4 CANS 1D パッケージの変更法

基本的なパッケージの変更は、基本課題モジュールの中にある 3 つのファイル main.f, model.f, bnd.f を変更します。モデルを大幅に変更する場合は、変更したいモデルパッケージ md_*** を以下のようにディレクトリごとコピーしてからおこなうと良いでしょう。

```
# cp -r md_shktb md_shktb1
```

2.4.1 計算パラメータの変更 (main.f)

基本課題の計算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 main.f を変更します。

メッシュ数の設定 (ix)

ここでは簡単な計算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 (md_shktb) 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、衝撃波管を問題をおこなったあとは以下のようになっています。

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

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=======================================================================|
      implicit real*8 (a-h,o-z)
      parameter (ix=207)
```

初期のメッシュ数は 207 です。これを約 2 倍の 407 に変更し、make してみましょう。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 1 次元計算では計算時間は約 4 倍になり、2 次元計算では計算時間は約 8 倍になります。

最終ステップ数、出力の設定 (tend, dtout)

続けて、最終時刻 (tend) 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 (26–32 行目)。

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

      dtout=0.05
      tend=0.14154
      nstop=1000000
```

ここでは、出力の時間間隔を 0.01 にしてみましょう。出力の時間間隔を小さくすることによって、短い時間間隔での変化を見ることができ、アニメーションのコマ数を増やすことができます。出力されるファイルの大きさは、それに応じて大きくなります。

実習課題

上記手順にそって、基本課題「流体衝撃波管問題 (md_shktb)」のメッシュ数、出力の時間間隔を変更して、計算を実行してみなさい。

2.4.2 モデルの変更 (model.f)

モデルの変更は model.f でおこないます。model.f では初期の密度 (ro)、速度 (vx, vy)、圧力 (pr)、磁場 (bx, by) などの分布を定めています。

断熱比 γ の変更 (gm)

ここでは、基本課題の超新星残骸：Sedov 解で断熱比 γ (gm) をいじってみましょう。断熱比は model.f を変更します。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。

実習課題

上記手順にそって、基本課題「超新星残骸:Sedov 解 (md_sedov)」の断熱比、出力の変更をしてみなさい。

2.4.3 計算のメインエンジンの変更 (main.f)

この節では計算のメインエンジンの変更をおこないます。

Roe 法への変更

基本課題 MHD 衝撃波管問題 (md_mhdshktb) を Roe 法で解いてみましょう。計算エンジンは main.f を変更します。以下 (120 行目) を探して見て下さい。

```
c     solve hydrodynamic equations

c                                                       hdmlw - start >>>
         call mlw_m(ro,pr,vx,vy,by,bx,bxm,dt,gm,dx,dxm,ix)
c                                                       hdmlw - end   <<<
c                                                       hdroe - start >>>
c        call roe_m(ro,pr,vx,vy,by,bx,bxm,dt,gm,dx,ix)
c                                                       hdroe - end   <<<
c                                                       hdcip - start <<<
c        call  cip_m(ro,pr,vx,vy,by,te,vxm,rodx,tedx,vxdxm,vydx
c     &    ,bx,bxm,dt,gm,dx,dxm,ix)
c                                                       hdcip - end   <<<
```

ここで、“mlw_m” は改良 Lax-Wendroff 法で MHD 方程式を解くサブルーチン、“roe_m” は Roe 法で MHD 方程式を解くサブルーチンです。以下のように “call mlw_m” にコメントをつけ、“call roe_m” のコメントをはずすことでメインエンジンを変更できます。

```
c     solve hydrodynamic equations

c                                                       hdmlw - start >>>
c         call mlw_m(ro,pr,vx,vy,by,bx,bxm,dt,gm,dx,dxm,ix)
c                                                       hdmlw - end   <<<
c                                                       hdroe - start >>>
        call roe_m(ro,pr,vx,vy,by,bx,bxm,dt,gm,dx,ix)
c                                                       hdroe - end   <<<
c                                                       hdcip - start <<<
c        call  cip_m(ro,pr,vx,vy,by,te,vxm,rodx,tedx,vxdxm,vydx
c     &    ,bx,bxm,dt,gm,dx,dxm,ix)
c                                                       hdcip - end   <<<
```

CIP 法への変更

基本課題 MHD 衝撃波管問題 (md_mhdshktb) を CIP 法で解いてみましょう。Roe 法への変更同様に、計算エンジンは main.f を変更します。CIP 法では、微分量を計算に必要とし、それらの変数を定義する記述が必要になります。そのため、計算エンジンの入れ替えの箇所以外に 3 箇所、CIP 独自の部分が存在します。hdcip を検索し、コメントをはずしてください。

```
c                                                          hdcip - start >>>
      dimension te(ix),vxm(ix),rodx(ix),tedx(ix),vxdxm(ix),vydx(ix)
c                                                          hdcip - end   <<<
```

```
c                                                          hdcip - start >>>
      call ciprdy_m(te,vxm,rodx,tedx,vxdxm,vydx,ro,pr,vx,vy
     &     ,gm,dx,dxm,ix)
      call cipbnd(margin,ro,te,vxm,vy,by,rodx,tedx,vxdxm,vydx,ix)
c                                                          hdcip - end   <<<
```

```
c-----------------------------------------------------------------------|
c     solve hydrodynamic equations

c                                                          hdmlw - start >>>
c         call mlw_m(ro,pr,vx,vy,by,bx,bxm,dt,gm,dx,dxm,ix)
c                                                          hdmlw - end   <<<
c                                                          hdroe - start >>>
c        call roe_m(ro,pr,vx,vy,by,bx,bxm,dt,gm,dx,ix)
c                                                          hdroe - end   <<<
c                                                          hdcip - start <<<
        call  cip_m(ro,pr,vx,vy,by,te,vxm,rodx,tedx,vxdxm,vydx
     &     ,bx,bxm,dt,gm,dx,dxm,ix)
c                                                          hdcip - end   <<<



      call bnd(margin,ro,pr,vx,vy,by,ix)
c     floor=1.d-9
c     call chkdav(n_floor,ro,vx,floor,ix)
c     call chkdav(n_floor,pr,vx,floor,ix)
c                                                          hdcip - start <<<
      call cipbnd(margin,ro,te,vxm,vy,by,rodx,tedx,vxdxm,vydx,ix)
c                                                          hdcip - end   <<<
```

2.4.4 境界条件の変更 (bnd.f)

次に境界条件について説明します。境界条件は bnd.f で決められています。基本課題の md_shktb の bnd.f を見てみましょう。

```
call bdsppx(0,margin,ro,ix)
call bdsppx(0,margin,pr,ix)
call bdspnx(0,margin,vx,ix)
call bdsppx(1,margin,ro,ix)
call bdsppx(1,margin,pr,ix)
call bdspnx(1,margin,vx,ix)
```

ここでは、境界条件モジュールの一つである bdsppx と bdspnx によって、境界が定められています。bdsppx は境界での符号を保存する対称境界で、bdspnx は符号反転を反転する対称境界です。

対称境界では、bdsppx(境界の位置, 境界の袖, 変数, メッシュ数) のように 4 つの変数を引きます。

- 始めの変数で、境界条件を与える袖の位置を指定しています。始めの変数が 0 の場合は左側の境界条件を定め、1 の場合は右側の境界条件を定めます。すなわち、上記の例では前半の 3 行は左側の境界条件、後半の 3 行は右側の境界条件を定めています。
- 境界の袖の大きさは margin というパラメータで定められています。境界での計算の精度を維持するために、margin は計算法によってその大きさを変える必要があります。改良 Lax-Wendroff 法では 1 以上、Roe 法では 2 以上、CIP 法では 4 以上が必要です。
- 3 番目の変数を、密度 (ro)、圧力 (pr)、速度 (vx) として、それぞれの境界条件を与えています。

他の境界条件に変える場合は、境界モジュール一覧を参照して、必要なものに変えます。例えば周期境界にする場合は、bdspnx を bdsppx に変更します。自由境界や一定値境界を指定する場合は、上記の 4 つの変数に加えて、他の変数も引く必要があります。詳しくは bc/README を参照して下さい。

第3章　CANS 2D, 3Dパッケージの説明

CANS 2Dや3Dのパッケージは、CANS 1Dパッケージ同様に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。基本課題モジュールには、並列計算機に対応したMPI Fortranで書かれたモデル“mdp_”やプログラムモジュールがあります。プログラムのコンパイル、実行、データ可視化の手順については、CANS 1Dを動かす方法とほぼ同じです。簡易的に手順を記します。

3.1　プログラムのコンパイル、実行 (make)

プログラムをコンパイル・実行する方法について説明します。パッケージのディレクトリ“cans2d”と“cansnc”の2箇所でmakeを実行します。“cans2d”でmakeし、実行アーカイブファイルlibcans2d.aを作成します。

```
# cd cans2d
# make
cd hdmlw; make
f77   -c  mlwfull.f
mlwfull.f:
        mlwfull:
f77   -c  mlwhalf.f
mlwhalf.f:
        mlwhalf:
(略)
```

CANS 1Dを実行していない場合は、“cansnc”でmakeし、実行アーカイブファイルlibcansnc.aを作成してください。(※ CANS 1Dを実行済みの場合はこの手順は必要ありません)。

CANS 3Dを実行したい場合も同様です。まずは、パッケージのディレクトリ“cans3d”でmakeし、実行アーカイブファイルlibcans3d.aを作成します。

```
# cd cans3d
# make
cd hdmlw; make
f77   -c  mlwfull.f
mlwfull.f:
        mlwfull:
f77   -c  mlwhalf.f
mlwhalf.f:
        mlwhalf:
(略)
```

CANS 1D の説明でも触れましたが、cans1d、cans2d、cans3d、cansnc が見えるディレクトリで make をおこなうと、全てのパッケージをコンパイルします。

3.2 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。例えば、Kelvin-Helmholtz 不安定性の基本課題 (md_kh) を動かしてみましょう。

```
# cd md_kh
# make
f77   -c  main.f
main.f:
 MAIN:
f77   -c  model.f
model.f:
        model:
f77   -c  bnd.f
bnd.f:
        bnd:
f77 -o a.out main.o model.o bnd.o \
-L../.. -lcans2d -lcansnc -L/usr/local/netcdf/lib -lnetcdf
./a.out
  write    step=       0 time= 0.000E+00 nd =  1
  write    step=      83 time= 0.100E+01 nd =  2
  write    step=     167 time= 0.201E+01 nd =  3
  write    step=     251 time= 0.301E+01 nd =  4
  write    step=     336 time= 0.401E+01 nd =  5
  write    step=     421 time= 0.500E+01 nd =  6
  write    step=     510 time= 0.600E+01 nd =  7
  write    step=     605 time= 0.701E+01 nd =  8
  write    step=     707 time= 0.801E+01 nd =  9
  write    step=     817 time= 0.900E+01 nd = 10
  write    step=     940 time= 0.100E+02 nd = 11
  stop     step=     940 time= 0.100E+02
   ### normal stop ###
```

CANS 2D の基本課題は、CANS 1D よりも計算に時間がかかります。課題にもよりますが、通常のワークステーションでは、数分から数時間にもなる場合があります。

3.3　結果の表示

結果の表示には可視化プログラム IDL を利用します。

3.3.1　IDL の起動 (idl)

idl を実行してみましょう

```
# idl
```

3.3.2　データ読み込み (.r rddt)

データを読み込んでみましょう。

```
IDL> .r rddt
```

3.3.3　データ表示 (.r pldt)

データを表示してみましょう。

```
IDL> .r pldt
```

3.3.4　データのアニメーション表示 (.r anime)

アニメーション表示をしてみましょう。

```
IDL> .r anime
```

3.3.5　IDL の終了 (exit)

IDL を終了してみましょう。

```
IDL> exit
```

このマニュアルについて

Version 3.0 (2003/04/30)
執筆：第 1 章：松元亮治、第 2 & 3 章：福田尚也、監修：横山央明