# 第1章　CANSを使った磁気流体力学シミュレーション

横山央明 (東京大学)
（ver.1: 花山秀和 (国立天文台) ）

## 1.1　CANSとは

### 1.1.1　天文数値ソフトウェア

物理法則にしたがって時間発展する系の進化を計算機で追及するという数値実験が、従来の理論・観測（または室内実験）に加わる第3の研究手法として確立してきた。とくに天体現象は、地上での実験が困難なのでこの手法が有効である。そのような目的で開発される計算コードは、研究者個人が作成し、個人的に使用されるか、あるいは人づてなど狭い範囲での受け渡しが多い。またある特定の研究目的を達成するために作成されているため、可読性が高くない、柔軟性がない、などの難点をもつ。そこで（ある程度の）汎用性をもつコードがいくつか公開されている。ZEUSやFlashなどはその代表例である。これらのコードに対抗して、国産のパッケージとして開発されたのがCANS（Coordinated Astronomical Numerical Softwares）である。

### 1.1.2　CANSで何ができるか？

CANSを用いると、さまざまな流体現象の数値シミュレーションを実施し可視化することができる。とくに天体現象に適用することをめざしており、それに対応した問題設定（後述）を用意している。なかでも磁気流体力学方程式を解くことを重要視しており、プラズマ流体現象を扱うことができる。

CANSでは、圧縮性流体・磁気流体を基礎として、熱伝導・粘性・磁気拡散などの散逸現象や、光学的に薄い放射による冷却などの物理過程をモジュールとして追加することができる。1次元から3次元までの問題に対応しており、MPIライブラリによる並列化が施されている。

### 1.1.3　CANSの特徴

CANSの最大の特徴は、方程式を解く計算コードそのものに加えて、「問題設定」が多数用意されていることである。これは、初期条件・境界条件・推奨パラメータ・可視化プログラム・解説をひとまとめにしたもので、（環境さえあれば）すぐにでも結果を可視化するところまでたどりつける。「問題設定」にはたとえば、衝撃波管問題・Rayleigh-Taylor不安定・Sedov解・磁気リコネクションなどの基本的な

問題以外に、天体への応用で恒星風・ジェット伝播・磁気回転不安定・フレア・Parker 不安定・自己重力収縮なども用意されている。これらはシミュレーションに慣れるという目的以外に、それぞれの「問題設定」を改造して実際の研究に使うことができるよう配慮されている。

また、おなじ方程式に対し複数の数値解法が用意されており、比較しながら計算法の特徴を知ることができる。たとえば磁気流体方程式に対しては、改良 Lax-Wendroff 法・Roe 流束による高精度風上差分法・CIP-MOCCT 法など（HLLD 法をまもなく実装予定）がある。拡散方程式に対しては、中心差分陽解法・SOR 陰解法・biCG 陰解法などがある。

## 1.2　CANS を使ってみよう

CANS のインストールから計算実行・可視化までを概観する。

### 1.2.1　動作環境

計算コードは、Fortran77（プラス多少の拡張仕様）で記述されている。したがって Fortran コンパイラが必要である。（GCC に含まれている）GNU Fortran で開発されているが、機種依存機能は極力使われておらず、普及しているコンパイラではほぼ問題なく動作する。コマンドライン（つまり非 GUI 環境）で利用されることを前提に作られていて、make コマンドが使えることが必須である。Windows の場合、Cygwin などで使うことを勧める。可視化には、IDL を使うことが前提でプログラムが用意されている。

### 1.2.2　インストールと IDL 設定

CANS のホームページから配布アーカイブをダウンロードして解凍する。結果以下のようなディレクトリ（フォルダ）構成になる。

```
# ls
cans/
# cd cans/
# ls
Makefile          cans3d/            idl-nosupport/  cans1d/
cans3d-nosupport/  index.html      cans1d-nosupport/  cansio/
cans2d/            cansio-nosupport/  cans2d-nosupport/  idl/
```

ディレクトリ cans1d/、cans2d/、cans3d/、cansio/が、計算コードと問題設定とが含まれる CANS の主たる構成要素である。idl/には可視化にもちいるプログラムの一部が含まれる。*-nosupport/となっているディレクトリに含まれるコードは、動作保証がない開発途中のコードである。ここにある index.html をブラウザで表示すると、CANS 問題設定の計算結果例の画像や動画をみることができる。

以下、cans2d/を例に説明を続ける。

```
# cd cans2d/
# ls
Makefile*     hdcip/       md_corjet/      md_mhdgwave/    md_rt/
README.txt*   hdglr/       md_diskjet/     md_mhdkh/       md_sedov/
bc/           hdmlw/       md_efr/         md_mhdshktb/    md_shkref/
              hdroe/        md_flare/       md_mhdsn/       md_shktb/
cndbicg/      htcl/        md_itmhdshktb/  md_mhdwave/     md_sndwave/
cndbicgmpi/   md_advect/   md_itshktb/     md_mri/         md_sninteraction/
cndsor/       md_awdecay/  md_jetprop/     md_mricyl/      md_thinst/
cndsormpi/    md_cme/      md_kh/          md_parkerinst/
common/       md_cndsp/    md_mhd3shktb/   md_reccnd/
commonmpi/    md_cndtb/    md_mhdcndtb/    md_recon/
```

ディレクトリ名の先頭に「md_」と付いているのが「問題設定」（MoDel）で、それ以外が方程式を解くサブルーチン群である。たとえば hdmlw/は、改良 Lax-Wendroff 法で流体方程式を解くルーチン群が入っている。cndbicg/は、熱伝導方程式を biCG 法で解くルーチンなどといった具合である。詳しくは「README.txt」に記述されている。

md_*について一部紹介すると、md_shktb/は衝撃波管問題（Sod の問題）、md_rt/は Rayleigh-Taylor 不安定、md_mri/は磁気回転不安定、md_efr/は太陽表面の浮上磁場現象、md_recon/は磁気リコネクションなどなど。詳しくは各ディレクトリ下の Readme.pdf ファイルを参照してほしい。

可視化に IDL を使う場合、CANS 用 IDL プロシジャが使えるよう、IDL の探索パスを設定する必要がある。Linux の場合シェル設定ファイル（.cshrc や.bashrc などユーザー依存）で環境変数 IDL_PATH に「（CANS をインストールしたディレクトリ）/cans/idl/」を追加する。よくわからない場合は、IDL に詳しいひとに尋ねてほしい。

### 1.2.3 準備コンパイル

計算コードをコンパイルする。まずサブルーチン群の「アーカイブファイル」を作る。CANS のルートディレクトリ（cans/）に戻って、以下のように make する。

```
# pwd
/（インストールした元ディレクトリ）/cans/
# cd cans/
# make
...
# ls
....
libcans1d.a libcans2d.a libcans3d.a libcansio.a
```

無事に成功する（MPI コードは除く）と、上記のように libcans*.a というファイルが複数つくられる。

libcans1d.a は、1 次元計算用のルーチン群のオブジェクトファイル（コンパイル済みのサブルーチンプログラム）をまとめたアーカイブである。libcansio.a は、各次元で共通に使うファイル入出力用ルーチン群。

なお上記の方法では通常、MPI 用のルーチン群はまだコンパイルされていない。そこで、必要に応じて同じディレクトリで

```
# pwd
/（インストールした元ディレクトリ）/cans/
# cd cans/
# make mpi FC=mpif90
```

などとして、MPI 用のコンパイラを使って再度同じ作業を繰り返す。

以上の作業で、方程式を解くなどの CANS 基本部分のサブルーチン群（以後「計算エンジン」と呼ぶ）のコンパイルは終了しており、結果がアーカイブファイルにまとめられた。こうしておけば、計算エンジンのルーチンについて再度コンパイルする必要がなくなり、作業効率があがる。また異なった問題を解く際に、共通のエンジン部分のコードをコピーする必要がなくなる。（いっぽう短所もあり、「問題設定」ディレクトリ（次節で説明）に完全な形でのプログラムファイルが存在せず、全体を見通すことが難しくなる。これについては近々対策する予定である。）

### 1.2.4　プログラムの実行

2 次元問題設定のひとつである md_mhdshktb/ 「磁気流体衝撃波管」（Brio-Wu 問題）で説明する。まず該当のディレクトリに入る。

```
# cd cans2d/md_efr/
# ls
Readme.pdf  Readme.tex  hdcip/  hdmlw/  hdroe/  index.html  mpi-hdmlw/
```

複数ディレクトリがあるのは、それぞれ使っている計算エンジンが異なるためである。ここでは hdmlw/ すなわち改良 Lax-Wendroff 法エンジンを使った場合で説明する。

```
# cd hdmlw/
# ls
Makefile   bnd.f   model.f  moviedt.pro  pldt.pro   pldt1d.pro
batch.pro  main.f  moviedt/  pldt.png      pldt1d.png  rddt.pro
```

各ファイルの説明は次のとおり。重要なものだけ示す。

- Makefile make コマンド用設定ファイル
- main.f メインルーチン
- model.f 初期条件設定ルーチン

- bnd.f 境界条件設定ルーチン
- *.pro 可視化用 IDL プログラム

コンパイル・実行するには make コマンドを使う。

```
# make
# ls
a.out ...
# make run
./a.out
  write    step=       0 time= 0.000E+00 nd =  1
...
  write    step=      49 time= 0.816E-01 nd =  5
  write    step=      61 time= 0.102E+00 nd =  6
  stop     step=      61 time= 0.102E+00
   ### normal stop ###
# ls
Makefile   bnd.f*  main.f*   moviedt/      pldt.png     pr.dac     vx.dac
a.out*     bnd.o   main.o    moviedt.pro*  pldt.pro*    rddt.pro*  vy.dac
az.dac     bx.dac  model.f*  out.txt       pldt1d.png   ro.dac     x.dac
batch.pro  by.dac  model.o   params.txt    pldt1d.pro*  t.dac      y.dac
```

上記のように「normal stop」メッセージで終了すれば、無事計算できたことになる。新しく次のファイル群が生成されるはずである。*.dac が計算結果データ（バイナリ形式）、out.txt が画面出力と同じメッセージ（テキスト形式）、params.txt が計算パラメータ（テキスト形式）。

### 1.2.5 IDL による可視化

IDL を立ち上げ、計算結果データを読み込み、可視化する。

```
# idl
IDL> .r rddt ; データ読み込み
IDL> .r pldt
 Plot columns & rows ? : 3 2  ; パネル数が聞かれる。この例では「3 行 2 列」を選択
 Variable for color-maps ? (ro,pr,te) : ro ; 表示物理量。この例では「密度」を選択
  start step ? : 0 ; 開始ステップ。この例では「0 番目」すなわち「初期条件から」を選択
```

うまくいけば新しくウィンドウが立ち上がり、上の例では「3 行 2 列に、密度の分布が、初期条件から」表示されるはずである。成功例として同ディレクトリに pldt.png というファイルが用意されているので、一致を確かめてほしい。なお pldt.pro では、指定した物理量以外に、速度が矢印で、磁力線が実線（実際はベクトルポテンシャル $A_z$ の等高線）で示される。

また moviedt.pro を使えば、動画の材料となる画像ファイルを作れる。上と同じように実行すると、あらたに moviedt/というディレクトリが作成され（デフォルトでは作成済み）、その下に連番で画像ファイルが作られる。これをつなぎ合わせれば動画となる。注意 : プレビューの動画は moviedt/以下のファイルを表示しているので、moviedt.pro を実行すると上書きされてしまう。

## 1.3　計算プログラムの解説

この節では、問題設定 cans1d/md_mhdshktb/hdmlw、磁気流体衝撃波管問題を改良 Lax-Wendroff 法で解く場合を例にとって実際のプログラムについて解説する。ここでは計算法の詳細には踏み込まない。改良 Lax-Wendroff 法についての詳しい説明は、「数値天文学テクニカルマニュアル 2004 年版」の第 1 章に掲載されているのでそちらを参照にしてほしい。

### 1.3.1　メインルーチン main.f

メインルーチン（ファイル main.f）は、大きく分けて「開始処理部（エピローグ）」「時間発展部」「終了処理部（エピローグ）」の 3 部構成からなる。さらにその前に、Fortran プログラムで必要な「配列定義部」がある。開始処理部では、サブルーチン model を呼び、メッシュ座標や初期条件を設定するのが主な動作である。時間発展部が主たる計算エンジンで、磁気流体方程式を解いて物理変数の時間発展を求める。与えられた終了条件をみたすまでこの部分が繰り返される。そして終了処理部では、終了メッセージを出力してプログラム実行を止める。以下、実際のプログラムにコメントを入れながら解説する。

```
c=======================================================================|
c     array definitions
c=======================================================================|
      implicit double precision (a-h,o-z)
! 倍精度浮動小数点データを基本データ型とする。
      parameter (margin=1)
      parameter (ix=1024+2*margin)
      dimension x(ix),xm(ix),dx(ix),dxm(ix)
! x(*)がメッシュ座標、xm(*)はメッシュ境界の座標、dx(*)がメッシュ幅、dxm(*)はメッシュ
中心間の距離。
! ixはx軸のメッシュ数
! marginは、興味ある領域の外に準備する、計算上必要な袖メッシュ数。計算エンジンに依存して
数が異なる。ここで用いるmlw_mでは、1個。

      dimension ro(ix),pr(ix)
      dimension vx(ix),vy(ix),by(ix)
! 時間変化する基本変数の配列。roは密度、prは圧力、vx、vyは速度、byは磁場
      dimension bx(ix),bxm(ix)
! x軸に沿った1次元MHDでは、磁場のx成分は時間変動しない（divB=0なので）。
! サブルーチンmodel.fで固定値として与えられる。

913   format (1x,' write    ','step=',i8,' time=',e10.3,' nd =',i3)
915   format (1x,' stop     ','step=',i8,' time=',e10.3)
c=======================================================================|
c     prologue
c=======================================================================|
      mcont=0
! mcont=1とすると、過去の計算結果データ（ディレクトリin/以下に置く）を読み込んで継続計
算を実行する。継続計算では、下のtendの値設定に注意すること。前回の計算の最終時刻がtend
より大きいと、継続計算がただちに終了してしまう。
c-----------------------------------------------------------------------|
c     set parameters controling finalization and data-output
      tend=0.1d0
! tendは終了させたい時刻。
      nstop=100000
! nstopは終了させたいステージ数。
! t > tend または ns>nstopが満たされたら計算終了。
      dtout=0.01d0
! dtoutは、データ出力間隔。tの値がdtoutの（ほぼ）整数倍になったら出力。
c     dtout=1.d-10
c     nstop=1
c-----------------------------------------------------------------------|
c  initialize counters
```

```
      merr  = 0
      ns    = 0
      t  = 0.0d0
      tp = 0.0d0
      nd=1
      mwflag=0
! 以上、カウンタの初期化。
! merr はエラーが起きたときゼロ以外の値が代入される。
! ns は計算ステージ数。
! t は時刻。tp は、ひとつ前のステージでの時刻。データ出力タイミングの判断に使う。
! nd は、次に出すデータが何番目か。
! mwflag は、「いま保持している計算結果を出力しました」というフラグ。重複して同じデータを
出力するのを防ぐための仕掛け。
c-------------------------------------------------------------------------|
c  file open for "standart output"
      mf_out=7
      open(mf_out,file='out.txt',status='replace',iostat=merr)
      if (merr.ne.0) then
        merr=10001
        goto 9999
      endif
      close(mf_out)
! 標準出力（コンソール出力）と同じ内容を残すファイル（out.txt）のオープン
! out.txt は出力のたびに開け閉めする。こうすると長時間の計算途中でも中身を確認できる。
c-------------------------------------------------------------------------|
c  file open
      mf_params=9
      call dacdefparam(mf_params,'params.txt')
! 計算パラメータを出力するファイル（params.txt）をオープン
      mf_t =10
      call dacdef0s(mf_t,'t.dac',6)
      mf_ro=20
      call dacdef1s(mf_ro,'ro.dac',6,ix)
      mf_pr=21
      call dacdef1s(mf_pr,'pr.dac',6,ix)
      mf_vx=22
      call dacdef1s(mf_vx,'vx.dac',6,ix)
      mf_vy=23
      call dacdef1s(mf_vy,'vy.dac',6,ix)
      mf_by=24
      call dacdef1s(mf_by,'by.dac',6,ix)
! 計算結果を出力するファイル（t.dac、ro.dac、pr.dac など）をオープン
! 引数「6」は、データ型が倍精度であることを示す。
```

```
      call dacputparami(mf_params,'ix',ix)
      call dacputparami(mf_params,'margin',margin)
! 計算パラメータを出力
c-----------------------------------------------------------------------|
c   setup numerical model (grid, initial conditions, etc.)
      call model(ro,pr,vx,vy,by,bx,bxm,gm,margin,x,ix,mf_params)
! メッシュ座標・初期条件を設定（後で詳述）
      call grdrdy(dx,xm,dxm,x,ix)
! メッシュ座標（x）から、メッシュ幅（dx）やメッシュ境界での座標など（xm、dxm）を計算
      call bnd(margin,ro,pr,vx,vy,by,ix)
! 境界条件の適用
c     floor=1.d-9
c     call chkdav(n_floor,ro,vx,floor,ix)
c     call chkdav(n_floor,pr,vx,floor,ix)
c-----------------------------------------------------------------------|
c  read-data
      ndi=1000
      if (mcont.eq.1) then
      mfi_t=60
      call dacopnr0s(mfi_t,'in/t.dac',mtype,nx0)
      mfi_ro=70
      call dacopnr1s(mfi_ro,'in/ro.dac',mtype,ix0,nx0)
      mfi_pr=71
      call dacopnr1s(mfi_pr,'in/pr.dac',mtype,ix0,nx0)
      mfi_vx=72
      call dacopnr1s(mfi_vx,'in/vx.dac',mtype,ix0,nx0)
      mfi_vy=73
      call dacopnr1s(mfi_vy,'in/vy.dac',mtype,ix0,nx0)
      mfi_by=74
      call dacopnr1s(mfi_by,'in/by.dac',mtype,ix0,nx0)
      do n=1,ndi
        read(mfi_t,end=9900) t
        read(mfi_ro) ro
        read(mfi_pr) pr
        read(mfi_vx) vx
        read(mfi_vy) vy
        read(mfi_by) by
      enddo
9900  continue
! 以上、継続計算（mcont=1）のとき、過去の計算結果（ディレクトリ in/の下に置く）を読み込
む。
      endif
```

```
c-------------------------------------------------------------------------|
c     data output
      mf_x=11
      call dacdef1d(mf_x,'x.dac',6,ix)
      write(mf_x) x
      close(mf_x)
      mf_bx=12
      call dacdef1d(mf_bx,'bx.dac',6,ix)
      write(mf_bx) bx
      close(mf_bx)
      call dacputparamd(mf_params,'gm',gm)
      write(mf_t) t
      write(mf_ro) ro
      write(mf_pr) pr
      write(mf_vx) vx
      write(mf_vy) vy
      write(mf_by) by
      write(6,913) ns,t,nd
      open(mf_out,file='out.txt',status='old',form='formatted'
     &    ,position='append')
            write(mf_out,913) ns,t,nd
      close(mf_out)
      nd=nd+1
! 初期条件を出力する
c=========================================================================|
c     time integration
c=========================================================================|
1000  continue
         ns = ns+1
         mwflag=0
c-------------------------------------------------------------------------|
c     obtain time spacing

         safety=1.0d0
         dtmin=1.d-10
         call cfl_m(dt,safety,dtmin,merr,gm,bx,ro,pr,vx,vy,by,dx,ix)
         if (merr.ne.0) goto 9999
! CFL 条件にもとづいて、時間刻み幅 dt を各ステージで決定する。
! safety は、CFL 条件の臨界値から実際の dt を求めるときの安全係数。通常 1 以下の値。
! dtmin は、dt の値に対する下限値。dt<dtmin になるとエラーとみなして計算を止める。
         tp = t
         t = t+dt
! 時刻の更新
```

```
c-------------------------------------------------------------------------|
c     solve hydrodynamic equations
         qav=1.d0
         call mlw_m(ro,pr,vx,vy,by,bx,bxm,dt,qav,gm,dx,dxm,ix)
! 主要エンジン部。ここでは改良 Lax-Wendroff（MLW）（と Lapidus 人工粘性）で MHD 方程式を解
く。
! qav は人工粘性の強さ。
         call bnd(margin,ro,pr,vx,vy,by,ix)
! 境界条件の適用
c        floor=1.d-9
c        call chkdav(n_floor,ro,vx,floor,ix)
! 計算結果をチェックして、floor の値より ro が小さいメッシュでは値を floor と置き換え、vx
をゼロとする。この問題設定では不要。
c        call chkdav(n_floor,pr,vx,floor,ix)
! 同様。ただし pr をチェック。
c-------------------------------------------------------------------------|
c     data output
         mw=0
         nt1=int(tp/dtout)
         nt2=int(t/dtout)
         if (nt1.lt.nt2) mw=1
! 「dtout の整数倍」の判断をする部分
         if (mw.ne.0) then
           write(mf_t) t
           write(mf_ro) ro
           write(mf_pr) pr
           write(mf_vx) vx
           write(mf_vy) vy
           write(mf_by) by
           write(6,913) ns,t,nd
           open(mf_out,file='out.txt',status='old',form='formatted'
     &          ,position='append')
                 write(mf_out,913) ns,t,nd
           close(mf_out)
            nd=nd+1
            mwflag=1
         endif
! データ出力
c-------------------------------------------------------------------------|
c     loop test
      if (ns .lt. nstop .and. t .lt. tend) goto 1000
! 主要計算部を繰り返すかどうかの判断文
! 「t<tend かつ ns<nstop」であれば繰り返す
```

```
c=======================================================================|
c     epilogue
c=======================================================================|
9999  continue
c-----------------------------------------------------------------------|
c  data output
      if (mwflag.eq.0) then
           write(mf_t) t
           write(mf_ro) ro
           write(mf_pr) pr
           write(mf_vx) vx
           write(mf_vy) vy
           write(mf_by) by
           write(6,913) ns,t,nd
           open(mf_out,file='out.txt',status='old',form='formatted'
     &          ,position='append')
             write(mf_out,913) ns,t,nd
           close(mf_out)
      endif
! データ出力
c-----------------------------------------------------------------------|
c  ending message
      write(6,915) ns,t
      if (merr.eq.0) then
        write(6,*) '  ### normal stop ###'
      else
        write(6,*) '  ### abnormal stop ###'
        write(6,*) '  merr = ',merr
      endif
      open(mf_out,file='out.txt',status='old',form='formatted'
     &    ,position='append')
      write(mf_out,915) ns,t
      if (merr.eq.0) then
        write(mf_out,*) '  ### normal stop ###'
      else
        write(mf_out,*) '  ### abnormal stop ###'
        write(mf_out,*) '  merr = ',merr
      endif
      close(mf_out)
! 終了メッセージ
      stop
! プログラム終了
      end
```

### 1.3.2 サブルーチン model.f：メッシュ座標・初期条件設定

サブルーチン model.f では、計算に必要なパラメータ（比熱比$\gamma$など）や、メッシュ座標（配列 x(ix)）、初期条件を設定する。

```
c=======================================================================|
      subroutine model(ro,pr,vx,vy,by,bx,bxm,gm,margin,x,ix
     &     ,mf_params)
c=======================================================================|
      implicit double precision (a-h,o-z)
c-----------------------------------------------------------------------|
      dimension dxm(ix),x(ix)
      dimension ro(ix),pr(ix),vx(ix)
      dimension vy(ix),by(ix)
      dimension bx(ix),bxm(ix)

c-----------------------------------------------------------------------|
c   parameters
c-----------------------------------------------------------------------|
      pi = acos(-1.0d0)
! pi は、円周率。逆三角関数を使って値を出している。
      gm=2.d0
! gm は、比熱比

c-----------------------------------------------------------------------|
c   grid
c-----------------------------------------------------------------------|
      dx0=1.d0/real(ix-margin*2)
! この問題設定では、一様幅のメッシュを配置するので、計算領域サイズ（=1 としている）とメッシュ数（ix）とからメッシュ幅を求める。袖メッシュの分をだけ引くのを忘れずに。

      do i=1,ix
         dxm(i)=dx0
      enddo

      izero=ix/2
        x(izero)=-dxm(izero)/2
! x=0 となる座標を、ix/2 番目と（ix/2+1）番目とのメッシュ境界に置いた。こうすれば メッシュ数が偶数になり、dx0 がキリのよい数値になる。
```

```
        do i=izero+1,ix
           x(i) = x(i-1)+dxm(i-1)
        enddo
        do i=izero-1,1,-1
           x(i) = x(i+1)-dxm(i)
        enddo
! メッシュ座標値の設定
c-----------------------------------------------------------------------|
c     initial condition
c-----------------------------------------------------------------------|
      b0=sqrt(4.d0*pi)

      do i=1,ix
         if (x(i).le.0) then
           ro(i)  = 1.d0
           pr(i)  = 1.d0
           by(i)  = b0
         else
           ro(i)  = 0.125d0
           pr(i)  = 0.1d0
           by(i)  = -b0
         endif
         vx(i) = 0.0d0
         vy(i) = 0.0d0
         bx(i) = b0*0.75d0
         bxm(i) = b0*0.75d0
      enddo
! 初期条件を設定。基本 Brio & Wu (1988) にならった。
! x 軸に沿った 1 次元 MHD では、磁場の x 成分（bx(*)、bxm(*)）は時間変動しない（divB=0 なので）。
! サブルーチン model.f で固定値として与えられる。

c-----------------------------------------------------------------------|
c     write parameters to file
c-----------------------------------------------------------------------|



      return
      end
```

### 1.3.3 サブルーチン bnd.f：境界条件

サブルーチン bnd.f では、境界条件を適用する。この問題設定 md_mhdshktb では、対称境界条件を与えている。

```
c=======================================================================|
      subroutine bnd(margin,ro,pr,vx,vy,by,ix)
c=======================================================================|
c     apply boundary condition
c-----------------------------------------------------------------------|
      implicit double precision (a-h,o-z)
      dimension ro(ix),pr(ix),vx(ix),vy(ix),by(ix)
c-----------------------------------------------------------------------|
      call bdsppx(0,margin,ro,ix)
      call bdsppx(0,margin,pr,ix)
      call bdspnx(0,margin,vx,ix)
      call bdsppx(0,margin,vy,ix)
      call bdsppx(0,margin,by,ix)
! 「bd（境界）s（対称）p(メッシュ境界）p（正対称）x（x 座標)」あるいは
! 「bd（境界）s（対称）p(メッシュ境界）m（反対称）x（x 座標)」の意味。
! 最初の引数「0（1)」は、x 座標の負（正）サイドの境界という意味。

      call bdsppx(1,margin,ro,ix)
      call bdsppx(1,margin,pr,ix)
      call bdspnx(1,margin,vx,ix)
      call bdsppx(1,margin,vy,ix)
      call bdsppx(1,margin,by,ix)


      return
      end
```