実習の手引き (差分法の実習、磁気流体基本課題)

実習テキスト：スカラー方程式の差分解法

1　スカラー方程式の差分解法パッケージの説明

スカラー方程式の差分解法を試してみましょう。以下のファイルが用意されています。

```
# ls scalar
Makefile    anime.pro   main.f      pldt.pro    pldtps.pro    rddt.pro
```

プログラムは Fortran 言語を用いて書かれています。Fortran は天体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よく用いられています。上記のリストで main.f が Fortran プログラムファイルです。

1.1　プログラムのコンパイルと実行 (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。ディレクトリ “scalar” に移動した後に make を実行します。するとプログラムがコンパイルされ、実行されます。正しくコンパイルされるとオブジェクトブファイル main.o と実行オブジェクトファイル a.out を作成します。正しく実行されるとデータファイル out.dat を出力します。

```
# cd scalar
# make
f77   -c -o main.o main.f
main.f:
 MAIN:
f77 -o a.out main.o
./a.out
  write    step=       0 time= 0.000E+00
  write    step=      50 time= 0.125E+02
  write    step=     100 time= 0.250E+02
   ### normal stop ###
# ls
Makefile    anime.pro   main.o      pldt.pro     rddt.pro
a.out*      main.f      out.dat     pldtps.pro
```

1.2　出力ファイルの説明 (out.dat)

出力ファイル out.dat はデータの確認を容易にするためにアスキー形式でかかれており、ファイルを直接エディタで開いて見ることができます。ファイルをみてみましょう。第 1 行目に配列の大きさ (jx) と時間データの数 (nx) がそれぞれ書かれています。第 2 行目に始め

の時間データの time step 数 (ns)、時刻 (time) が書かれています。第 3 行目から第 102 行目に渡って、始めの時間データの x 座標 (x)、そこでの変数の値 (u) が順にかかれています。第 103 行目移行に、次の時間データが書かれています。書式については、Fortran プログラム main.f の 53, 55, 59 行目に Format 文で指定されていますので、参考にしてください。

```
# head out.dat
  100,    3
    0,  0.00
  1.0, 1.0000000
  2.0, 1.0000000
  3.0, 1.0000000
  (後略)
```

1.3 結果の可視化表示

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。)

1.3.1 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>
```

1.3.2 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。ファイル"out.dat" からデータが読み込まれ、idl でデータを利用できるようになります。

```
IDL> .r rddt
```

.r は run を意味します。

1.3.3 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
```

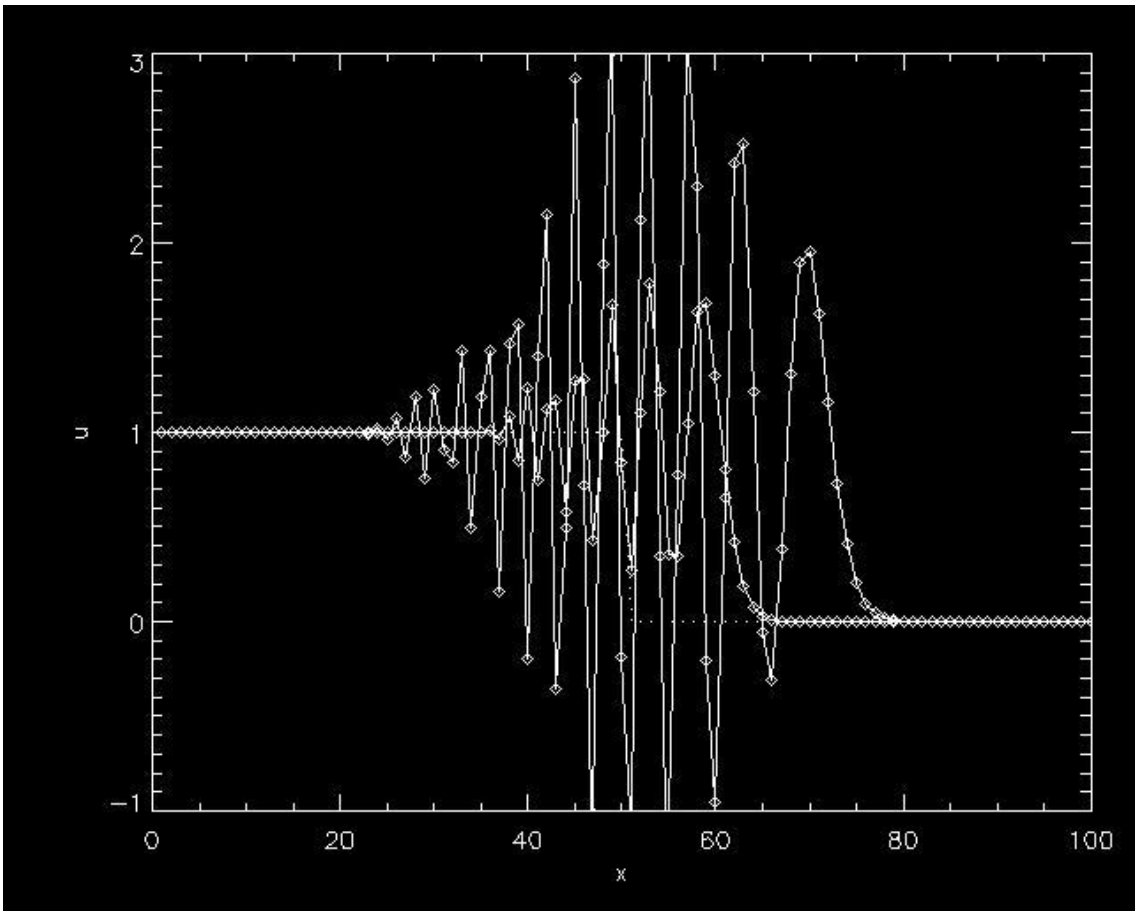

図 1: scalar パッケージの計算結果の例

1.3.4 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。

```
IDL> exit
```

2　プログラムの変更について

2.1　計算エンジンの変更

サンプルプログラムの計算エンジンは FTCS になっています。以下の 72 行目から 88 行目の間で、計算エンジンを変更してください。

```
c-----------------------------------------------------------------------|
c     solve equation
c
c                                                          ftcs - start
>>>
      do j=1,jx-1
         f(j)=0.5*cs*(u(j+1)+u(j))
      enddo

      f(jx)=f(jx-1)

      do j=2,jx-1
         u(j)=u(j)-dt/dx*(f(j)-f(j-1))
      enddo

      u(1)=u(2)
      u(jx)=u(jx-1)
c                                                          ftcs - end   >>>
```

2.2　メッシュ数の設定 (jx)

メッシュ数を変更するには、5 行目の parameter 文にある jx の値を変更します。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。

```
      parameter (jx=100)
```

2.3 最終ステップ数、出力の設定 (nstop, nskip)

最終ステップ数と出力ファイルの間隔は、14 行目の nstop と 15 行目の nskip の値をそれぞれ変更します。ファイルの出力間隔を短くすることで、アニメーションを滑らかに表示することができます。その一方、出力されるファイルのサイズは大きくなります。

```
c   time control parameters

      nstop=100
      nskip = 50
```

2.4 CFL 条件の変更 (safety)

CFL 条件は、68 行目の (safety) の値を変更します。出力されるデータファイルの時間間隔は、現在、nskip で制御しているため、safety の値を変更すると、出力される時間も変わることに注意してください。

```
c     obtain time spacing
      safety=0.25
```

3 データのアニメーション表示 (.r anime)

データのアニメーション表示には anime.pro というプログラムを用います。IDL でデータを読み込んだ後に、以下のように入力してみて下さい。

```
IDL> .r anime
```

デフォルトのままでは、データのステップ間隔が大きく、きれいにみれません。nskip を 1 にしてどのように進化するか見てみましょう。anime.pro の window が開いている状態で、もう一度、anime.pro を実行するとエラーが出ますので注意してください。

付録

サンプルプログラム、main.f

```
c=====================================================================|
c     array definitions
c=====================================================================|
      implicit real*8 (a-h,o-z)
      parameter (jx=100)

      dimension x(1:jx),u(1:jx),f(1:jx)

c=====================================================================|
c     prologue
c=====================================================================|
c   time control parameters

      nstop=100
      nskip = 50

c---------------------------------------------------------------------|
c  initialize counters

      time  = 0.0
      ns    = 0

      nx = nstop/nskip+1
c---------------------------------------------------------------------|
c     Set initial condition
c---------------------------------------------------------------------|
      pi=4.*atan(1.0)
c  grid
      dx=1.0
      x(1)=dx
      do j=1,jx-1
         x(j+1)=x(j)+dx
      enddo
```

```
c
c  variable
      do j=1,jx/2
         u(j)= 1.0
      enddo
      do j=jx/2+1,jx
         u(j)= 0.0
      enddo
c
c  velocity
      cs=1.0

c-----------------------------------------------------------------------|
c     Output initial condition
c
      write(6,103) ns,time
 103  format (1x,' write    ','step=',i8,' time=',e10.3)
      open(unit=10,file='out.dat',form='formatted')
      write(10,100) jx,nx
 100  format(i5,',',i5)
      write(10,101) ns,time
 101  format (i5,',',f6.2)
      do j=1,jx
         write(10,102) x(j),u(j)
      enddo
 102  format(f5.1,',',f10.7)

c=======================================================================|
c     time integration
c=======================================================================|
1000  continue
      ns = ns+1
c-----------------------------------------------------------------------|
c     obtain time spacing
      safety=0.25
      dt=safety*dx/cs
      time=time+dt
```

```
c-----------------------------------------------------------------------|
c     solve equation
c
c                                                          ftcs - start >>>
      do j=1,jx-1
         f(j)=0.5*cs*(u(j+1)+u(j))
      enddo

      f(jx)=f(jx-1)

      do j=2,jx-1
         u(j)=u(j)-dt/dx*(f(j)-f(j-1))
      enddo

      u(1)=u(2)
      u(jx)=u(jx-1)
c                                                          ftcs - end   >>>
c-----------------------------------------------------------------------|
c     data output
      if (mod(ns,nskip).eq.0) then
         write(6,103) ns,time
         write(10,101) ns,time
         do j=1,jx
            write(10,102) x(j),u(j)
         enddo
      endif

      if (ns .lt. nstop) goto 1000
      close(10)
*=======================================================================|
      write(6,*) '  ### normal stop ###'
      end
```

サンプルプログラム、rddt.pro

```
; rddt.pro
openr,1,'out.dat'
readf,1,jx,nx

; define array
ns=intarr(nx)
t=fltarr(nx)

x=fltarr(jx)
u=fltarr(jx,nx)

; temporary variables for read data
ns_and_t=fltarr(2,1)
x_and_u=fltarr(2,jx)

for n=0,nx-1 do begin
 readf,1,ns_and_t
 readf,1,x_and_u
 ns(n)=fix(ns_and_t(0,0))
 t(n)=ns_and_t(1,0)
 u(*,n)=x_and_u(1,*)
endfor

close,1
free_lun,1

x(*)=x_and_u(0,*)

delvar,ns_and_t,x_and_u

help
end
```

サンプルプログラム、pldt.pro

```
!x.style=1
!y.style=1
!p.charsize=1.4

plot,x,u(*,0),xtitle='x',ytitle='u',linest=1,yrange=[-1,3],xrange=[0,100]
for n=1,nx-1 do begin
oplot,x,u(*,n)
oplot,x,u(*,n),psym=4
endfor

end
```

サンプルプログラム、anime.pro

```
!x.style=1
!y.style=1
!p.charsize=1.4

window,xsize=480,ysize=480
xinteranimate,set=[480,480,nx]

for n=0,nx-1 do begin

plot,x,u(*,n),xtitle='x',ytitle='u',yrange=[-1,3],xrange=[0,100]
oplot,x,u(*,n),psym=4

xinteranimate,frame=n,window=0

endfor

xinteranimate

end
```

差分法の実習課題

1　1 次元波動方程式

スカラー方程式の差分解法のパッケージを実習テキスト (p273 から p282) に沿って動かしなさい。サンプルプログラムは、FTCS スキームを用いて、1 次元波動方程式を計算するものである。初期条件は、$j = 1, ...50$ に対して $u_j = 1$、$j = 51, ...100$ に対して $u_j = 0$、クーラン数 $\nu = c\Delta t/\Delta x = 0.25$ として計算をおこなう。このプログラムを

1. Lax-Wendroff 法によるもの、
2. 空間 1 次精度の風上差分によるもの、
3. 流束制限関数として minmod 関数を用いたもの

に書きかえ、計算結果をグラフ表示し、テキストの結果 (図 1.12, 図 1.10) と比較しなさい。

補足解説

1 次元波動方程式

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \tag{1}$$

を差分化すると

$$u_j^{n+1} = u_j^n - \frac{\Delta t}{\Delta x}(f_{j+1/2}^n - f_{j-1/2}^n) \tag{2}$$

となる。ここで、FTCS スキームの数値流束を用いると

$$f_{j+1/2}^n = \frac{1}{2}(f_{j+1} + f_j) = \frac{1}{2}c(u_{j+1} + u_j) \tag{3}$$

となる。数値流束に関しては p21 にまとめられており、これを用いるとよい。変更箇所に関しての解説は p276 に記されている。

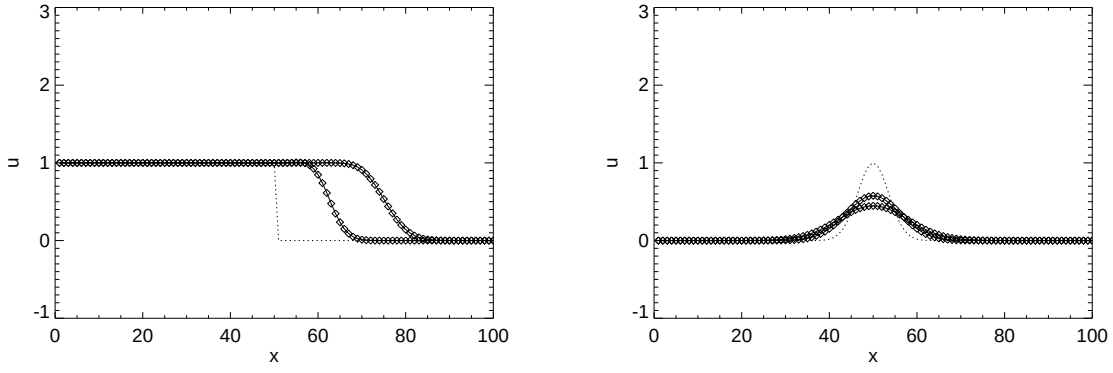

図 2: 計算結果。左：Lax-Wendroff 法+流束制限関数。右：1 次元拡散方程式。

2 Burgers 方程式

1 次元波動方程式のプログラムを参考にして Burgers 方程式

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) = 0 \tag{4}$$

を 1 次精度風上差分法を用いて解くプログラムを作成し、結果をグラフ表示し、テキストの結果 (図 1.15, 図 1.16) と比較しなさい。数値流束に関しては p23 にかかれている。

$$f^n_{j+1/2} = \frac{1}{2}\left\{\left(\frac{{u_{j+1}}^2}{2} + \frac{{u_j}^2}{2}\right) - \frac{1}{2}|u_{j+1} + u_j|(u_{j+1} - u_j)\right\} \tag{5}$$

と書くこともできる。

3 1 次元拡散方程式

1 次元波動方程式のプログラムを参考にして 1 次元拡散方程式

$$\frac{\partial u}{\partial t} = \kappa \frac{\partial^2 u}{\partial x^2} \tag{6}$$

を FTCS スキームを用いて解くプログラムを作成し、適当な初期条件を設定してシミュレーションを行い、結果をグラフ表示しなさい。

例えば、初期の分布にはガウス分布を仮定し、cs の代わりに kappa を定義しましょう。

```
c  variable
      do j=1,jx
         u(j)= exp(-(((x(j)-x(jx/2))/5.)**2))
      enddo
c
c  kappa
      kappa=1.0
```

実習テキスト：磁気流体基本課題

1　はじめに

千葉大学を始めとするグループでは、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。

2　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。

3　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。

4　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、CANS のパッケージの一覧を紹介した後、磁気流体一次元基本課題として、1 次元数値シミュレーション用の CANS1D パッケージの説明を行ないます。続いて 2 次元、3 次元数値シミュレーション用のパッケージ CANS2D, CANS3D の使い方を説明します。

5 CANSパッケージ一覧

```
Makefile パッケージをコンパイルするための Makefile
cans1d/    CANS 1D (1 次元計算) の Fortran プログラム
cans2d/    CANS 2D (2 次元計算) の Fortran プログラム
cans3d/    CANS 3D (3 次元計算) の Fortran プログラム
cansnc/    ファイルを入出力するためのモジュール
idl/       データ可視化に使用する IDL 用プログラム
```

注意事項：公開パッケージ cans-1.0 には，cans3d はついていません．

6 CANS関連 Webページ

宇宙シミュレーション・ネットラボラトリシステムホームページ

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

CANS デモページ

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

実習テキスト：磁気流体一次元基本課題

1 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、章末のリストや Web ページを参考にして下さい。

1.1 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。

1.2 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 params.txt や**.dac 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
        ./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 ###
```

1.3 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。)

1.3.1 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>
```

1.3.2 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
```

データを読み込んだあと help を実行すると、読み込んだ変数の情報などが表示されます。

```
IDL> help
(前略)
GM              FLOAT     =       1.40000
IX              LONG      =           208
NX              LONG      =             4
PR              FLOAT     = Array[208, 4]
PR1             FLOAT     =      0.100000
RO              FLOAT     = Array[208, 4]
RO1             FLOAT     =      0.125000
T               FLOAT     = Array[4]
TE              FLOAT     = Array[208, 4]
VX              FLOAT     = Array[208, 4]
X               FLOAT     = Array[208]
Compiled Procedures:
    $MAIN$  NCGETO1     NCGETOS     NCGETS1     NCGETSS
(後略)
```

PR(圧力)、RO(密度)、TE(温度)、VX(速度) が 2 次元配列 (空間方向と時間方向) となっており、X(空間) と T(時間) が 1 次元配列となっています。help を実行した場合、変数は大文字でかかれていますが、idl では小文字と大文字は区別されないので、PR と pr は同じ意味です。

1.3.3 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
```

1.3.4 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

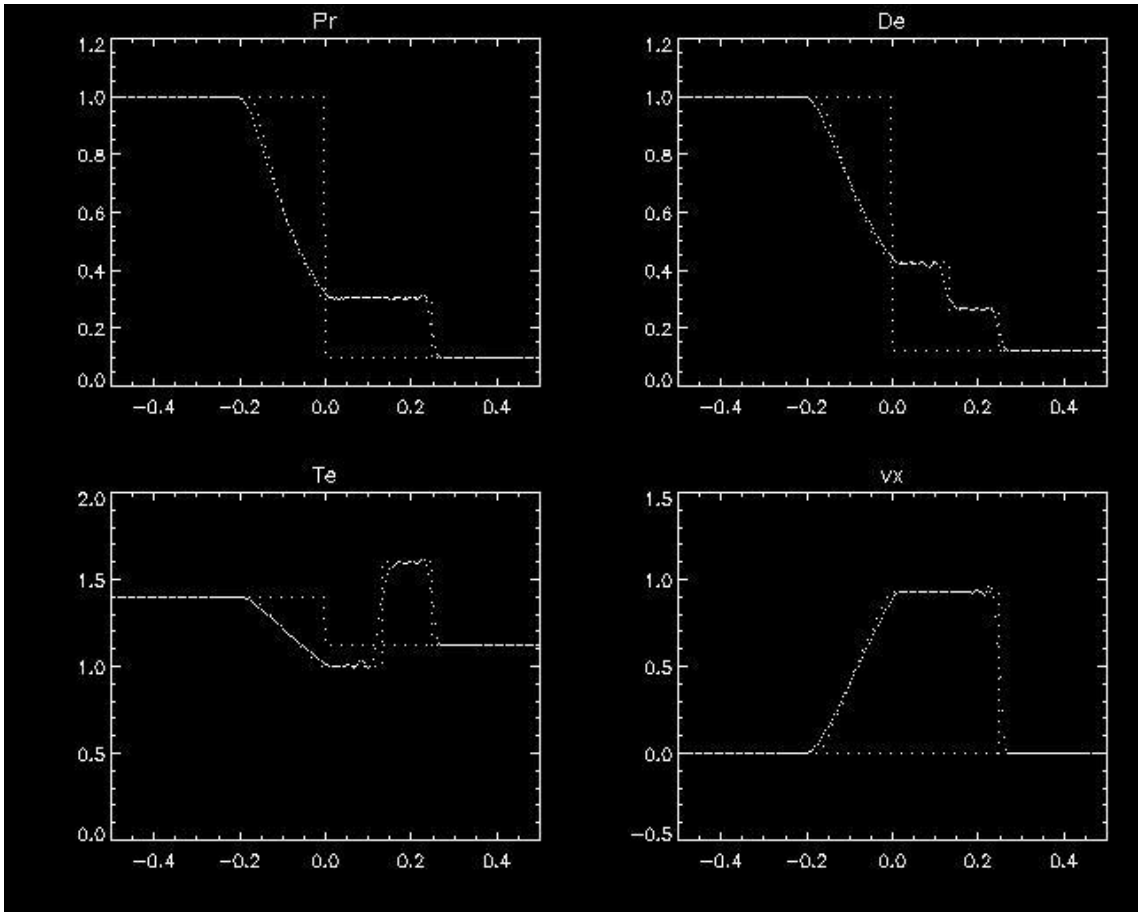

図 3: md_shktb の結果

て下さい。

```
IDL> .r anime
```

1.3.5 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> exit
```

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。

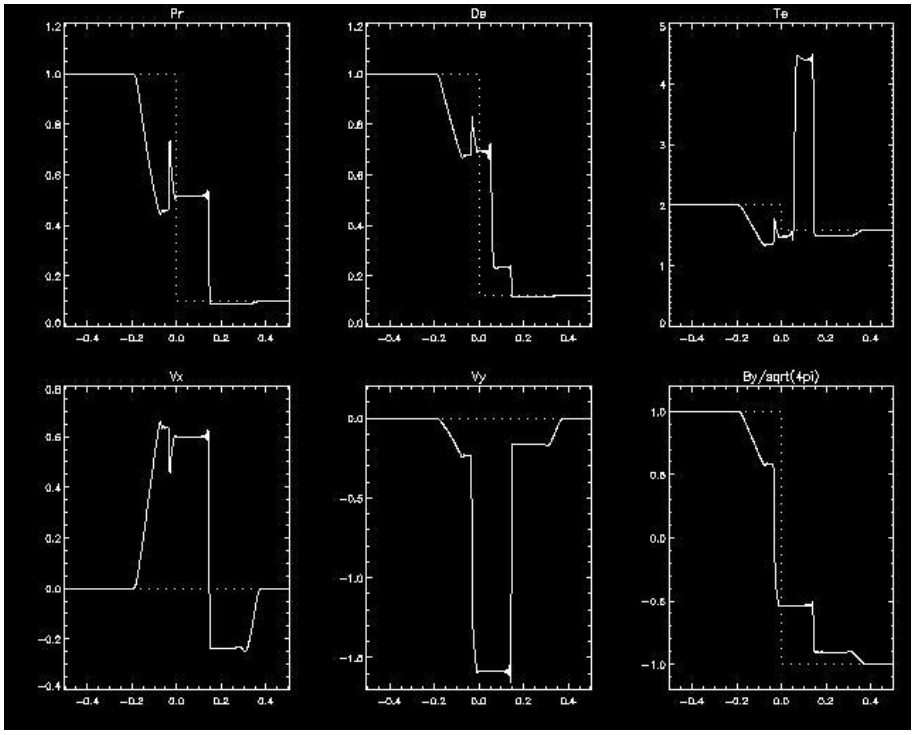

図 4: md_mhdshktb の結果

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 `params.txt` や`***.dac` を上書きします。必要な出力ファイルは、名前を変更したり、別ディレクトリに移してとっておきましょう。
- オブジェクトファイル、出力ファイルを消去したい場合は、`make clean` を実行してください。

2 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
```

2.1 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。

2.1.1 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、衝撃波菅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.f     main.o      pldt.pro
Makefile-nc    Readme.pdf  bnd.o     model.f     shktb_analytic.pro
Makefile-pg    Readme.ps   cipbnd.f  model.o     pr.dac
Makefile-pgnc  cipbnd.o  params.txt  rddt.pro  t.dac
README         a.out*      rdnc.pro  vx.dac
anime.pro   main.f    pldt.f      ro.dac    x.dac
```

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=1026)
```

初期のメッシュ数は 1026 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。

2.1.2 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。

2.2 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。

2.2.1 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。

2.3 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。

2.3.1 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 >>>
         qav=3.d0
         call mlw_m(ro,pr,vx,vy,by,bx,bxm,dt,qav,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        cvis=3.00d0
c        call  cip_m(ro,pr,vx,vy,by,te,vxm,rodx,tedx,vxdxm,vydx
c    &     ,bx,bxm,dt,cvis,gm,dx,dxm,ix)
c                                                       hdcip - end   <<<
```

ここで、“mlw_m” は改良 Lax-Wendroff で MHD 方程式を解くサブルーチン、“roe_m” は改良 Lax-Wendroff 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        qav=3.d0
c        call mlw_m(ro,pr,vx,vy,by,bx,bxm,dt,qav,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        cvis=3.00d0
c        call  cip_m(ro,pr,vx,vy,by,te,vxm,rodx,tedx,vxdxm,vydx
c    &     ,bx,bxm,dt,cvis,gm,dx,dxm,ix)
c                                                          hdcip - end   <<<
```

また、Roe 法と Lax-Wendroff 法とでは、「safety」の最適値が異なります。詳しくは main.f のコメント文を参照してください。

2.3.2 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     solve hydrodynamic equations

c                                                          hdmlw - start >>>
c        qav=3.d0
c        call mlw_m(ro,pr,vx,vy,by,bx,bxm,dt,qav,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 <<<
         cvis=3.00d0
         call  cip_m(ro,pr,vx,vy,by,te,vxm,rodx,tedx,vxdxm,vydx
     &    ,bx,bxm,dt,cvis,gm,dx,dxm,ix)
c                                                          hdcip - end   <<<
```

また、Roe 法と Lax-Wendroff 法とでは、「safety」の最適値が異なります。詳しくは main.f のコメント文を参照してください。

2.4 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。前半の 3 行は右側の境界条件、後半の 3 行は左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 bdperx 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。

CANS 1D モジュール一覧 (cans-1.0 版)

プログラムモジュール

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
hdcip/	CIP 法で流体計算をするためのモジュール
bc/	境界条件を定義するためのプロシジャ
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
cndsor/	熱伝導を陰解法（時間 1 次精度：行列反転は Red Black SOR 法）で解くためのモジュール
cndbicg/	熱伝導を陰解法（時間 1 次精度：行列反転は BICG 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
selfg/	自己重力を解くためのモジュール
md_***/	一次元基本課題モジュール（詳しくは後述）

ドキュメント

README	簡易解説ドキュメント
md_***/README	課題についてのドキュメント
md_***/Readme.pdf	課題についての PDF 形式のドキュメント
md_***/Readme.ps	課題についての PS 形式のドキュメント

基本課題モジュール

```
md_advect/ 単純移流　[移流]
md_shktb/ 衝撃波菅　[流体]
md_shkform/ 衝撃波生成　[流体]
md_strongshk/ 強い衝撃波管　[流体]
md_itshktb/ 等温衝撃波菅　[等温流体]
md_mhdshktb/ MHD 衝撃波菅　[MHD]
md_itmhdshktb/ 等温 MHD 衝撃波菅　[等温 MHD]
md_awdecay/ 大振幅 Alfven 波減衰不安定　[等温 3 成分 MHD]
md_sedov/ 超新星残骸：Sedov 解　[流体、非一様断面]
md_thinst/ 熱不安定　[放射冷却]
md_cndtb/ 単純熱伝導　[熱伝導]
md_cndsp/ 球対称単純熱伝導　[熱伝導、非一様断面]
md_spicule/ スピキュール　[MHD、非一様断面、重力]
md_wind/ 恒星風：Parker 解　[流体、非一様断面、重力]
md_mhdwind/ MHD 恒星風：Weber-Davis 解　[MHD、非一様断面、重力、回転]
md_diskjet/ 降着円盤からの MHD ジェット　[MHD、非一様断面、重力、回転]
md_flare/ フレア　[流体、非一様断面、重力、放射冷却、熱伝導]
md_cloud/ 等温自己重力収縮　[等温流体、自己重力]
md_clsp/ 球対称等温自己重力収縮　[等温流体、非一様断面、自己重力]
md_relshk/ 特殊相対論衝撃波管　[相対論流体]
```

境界条件モジュール

使用の際に変更をすることの多い境界条件について，個々のファイルの説明を述べます.

```
bdcnsx 一定値境界　例： qq(1)=q0
bdfrex 自由境界　例：qq(1)=qq(2)
bdfrdx 自由境界、微分一定。　例：qq(1)=qq(2)-dqq(2)*dx
bdperx 周期境界　例：qq(1)=qq(ix)
bdsppx 対称境界（グリッド点間に境があり、符号保存） 例： qq(1)= qq(2)
bdspnx 対称境界（グリッド点間に境があり、符号反転） 例： qq(1)=-qq(2)
bdsmpx 対称境界（グリッド点上に境があり、符号保存） 例： qq(1)= qq(3)
bdsmnx 対称境界（グリッド点上に境があり、符号反転） 例： qq(1)=-qq(3)
```

実習テキスト：磁気流体二次元基本課題

1　CANS 2D, 3Dパッケージの説明

CANS 2DやCANS 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。

1.1　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)。

```
# cd ../cansnc
# make
(後略)
```

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。

1.2 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
./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。

1.3 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。

1.3.1 IDL の起動 (idl)

idl を実行してみましょう

```
# idl
```

1.3.2 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
```

1.3.3 データ表示 (.r pldt)

データを表示してみましょう。

```
IDL> .r pldt
% Compiled module: $MAIN$.
 Plot columns \& rows ? : 2,2
 Variable for color-maps ? (ro,pr,te) : ro
  start step ? : 0
```

2 次元データを表示する場合、いくつかの変数を指定する必要があります。はじめに、画面に表示するデータの数を、行と列の数で指定します。上記の例では、2 行 2 列でデータを表示します。次に表示する変数を指定します。ro は密度、pr は圧力、te は温度です。変数としては、他に速度ベクトル (vx, vy, vz)、磁場ベクトル (bx, by, bz) があります。最後に、始めに表示するデータを指定します。0 とすると始めに出力したデータから表示されます。

1.3.4 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
```

1.3.5 色の変更 (loadct,39 や xloadct)

可視化表示の色を変更するには、loadct,39 などを実行した後に再び、pldt.pro を実行します。

```
IDL> loadct,39
```

loadct の後の番号は、色テーブルを表します。色テーブルは 0 から 40 番まであり、デフォルトは 0 番の B-W Linear(白黒) で、良く使われる 39 番は rainbow+white(虹+白) となっています。色テーブルは、xloadct を実行することで確認することができます。

```
IDL> xloadct
```

1.3.6 IDL の終了 (exit)

IDL を終了してみましょう。

```
IDL> exit
```

CANS 2D モジュール一覧 (cans-1.0 版)

プログラムモジュール

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

bc/　境界条件を定義するためのプロシジャ
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

cndsor/ 熱伝導を陰解法（時間 1 次精度：行列反転は Red Black SOR 法）で解くためのモジュール
cndbicg/ 熱伝導を陰解法（時間 1 次精度：行列反転は BICG 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
selfgmg/自己重力を Multigrid Iteration で解くためのモジュール（周期境界のみ）

commonmpi/ パラレル計算機で
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 (MPI)
cndsormpi/ パラレル計算機で
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 (MPI)

md_***/　二次元基本課題モジュール [スカラー計算機用]（詳しくは後述）
mdp_***/　二次元基本課題モジュール [パラレル計算機用]（詳しくは後述）

ほとんどの二次元基本課題に対しては，スカラー計算機用とパラレル計算機 (並列計算機) 用の二種類のパッケージが用意されています．それぞれは，ディレクトリの名前で区別されています．例えば，md_shktb と mdp_shktb とは，スカラー計算機とパラレル計算機でそれぞれ基本的に同じ計算を行ないます.

基本課題モジュール

md_advect/ 単純移流 [移流]
md_shktb/ 衝撃波菅 [流体]
md_itshktb/ 等温衝撃波菅 [等温流体]
md_mhdshktb/ MHD 衝撃波菅 [MHD]
md_itmhdshktb/ 等温 MHD 衝撃波菅 [等温 MHD]
md_mhd3shktb/ 3 成分 MHD 衝撃波菅 [3 成分 MHD]
md_awdecay/ 大振幅 Alfven 波減衰不安定 [等温 3 成分 MHD]
md_thinst/ 熱不安定 [放射冷却]
md_cndtb/ 単純熱伝導 [熱伝導]
md_mhdcndtb/ 単純 MHD 熱伝導 [MHD、熱伝導]
md_shkref/ 反射衝撃波 [流体]
md_kh/ Kelvin-Helmholtz 不安定性 [流体]
md_rt/ Rayleigh-Taylor 不安定性 [流体、重力]
md_sndwave/ 線形音波伝播 [流体]
md_mhdwave/ 線形 MHD 波動伝播 [MHD]
md_mhdkh/ MHD Kelvin-Helmholtz 不安定性 [MHD]
md_mhd3kh/ 3 成分 MHD Kelvin-Helmholtz 不安定性 [3 成分 MHD]
md_recon/ 磁気リコネクション [MHD、抵抗]
md_recon3/ 3 成分磁気リコネクション [3 成分 MHD、抵抗]
md_efr/ 太陽浮上磁場：Parker 不安定 [MHD、重力]
md_corjet/ 太陽コロナジェット [MHD、重力、抵抗]
md_mri/ 磁気回転不安定性 [MHD、潮汐 Coriolis 力]
md_mricyl/ 磁気回転不安定性 [MHD、潮汐 Coriolis 力、回転]
md_reccnd/ 熱伝導磁気リコネクション [MHD、抵抗、熱伝導]
md_cndsp/ 球対称単純熱伝導 [熱伝導、円柱・球座標]
md_sedov/ 超新星残骸：Sedov 解 [流体、円柱・球座標]
md_jetprop/ ジェット伝播 [流体、円柱座標]
md_mhdsn/ MHD 超新星残骸 [MHD、円柱・球座標]
md_diskjet/ 降着円盤ジェット [MHD、円柱座標]
md_cme/ 太陽コロナ質量放出：Low 解 [MHD、球座標]
md_cloud/ 等温自己重力収縮 [等温流体、自己重力]
md_mhdcloud/ 等温 MHD 自己重力収縮 [等温 MHD、自己重力]
md_mhdgwave/ 成層大気中の MHD 波動 [MHD]
md_parker/ 銀河 Parker 不安定 [MHD、重力]

Version 3.2 (2003/08/08) 福田尚也、松元亮治、監修：横山央明

応用課題

１．磁気リコネクション（Craig-Henton 解の安定性２）　柴田一成・田沼俊一
２．浮上磁場(パーカー不安定)とリコネクション　野澤　恵･宮腰剛広
３．降着円盤とジェット　松元亮治･桑原匠史
４．分子雲コアの重力収縮　福田尚也
５．太陽風磁気圏相互作用の３次元 MHD シミュレーション　荻野竜樹･中尾真季

応用課題の簡単な説明

１．磁気リコネクション（Craig-Henton 解の安定性２）　柴田一成・田沼俊一

磁気リコネクションは、太陽フレアや磁気嵐を引き起こす重要な現象である。しかし、その解析解はこれまでずっと知られていなかった。ところが、Craig& Henton (1995)によって、定常・非圧縮の仮定のもとで解析解（特解）が発見された。そこで、2002 年度サマースクールでは、この解の安定性を調べた。その結果、Craig-Henton 解は、パラメータの値によっては不安定であることが分かった（その成果を発展させたものが、Hirose et al. 2004, ApJ, 610,1107）。今回は、まだ調べられていないパラメータ領域での安定性を調べる。

２．浮上磁場(パーカー不安定)とリコネクション　野澤 恵・宮腰剛広

太陽の内部からパーカー不安定により光球に現れる浮上磁場とコロナ磁場のリコネクションによるエネルギー解放が、太陽活動の一部であると考えられている。そこで、初期に対流層から浮上する磁場を考え、上昇する間にリコネクションを起すモデルを用いる。特に今回は、コロナ中での激しいリコネクションでなく、ゆるやかであるがこれからの研究対象である光球リコネクションや彩層上部でのリコネクションについて二次元 MHD コードを用いて、MHD シミュレーションを行なう。可能であれば CIP 化したコードも使用するつもりである。

３．降着円盤とジェット　松元亮治・桑原匠史

降着円盤から磁気流体ジェットが形成される過程を円筒座標系 2 次元軸対称の MHD コードを用いてシミュレートする。最近、銀河系内のブラックホール候補天体では、降着円盤が光学的に薄い高温状態から光学的に厚い低温状態に遷移する際に、高速ジェットが噴出していることがわかってきた。そこで、高温トーラスを鉛直方向あるいは方位角方向の磁場が貫いているという初期状態から出発して輻射冷却を含めたシミュレーションを実施し、上記の状態遷移が円盤内部の磁場及びジェット噴出に及ぼす影響を調べる。

４．分子雲コアの重力収縮　　福田尚也

近年、近赤外の減光の観測から孤立した分子雲コアであるグロビュールの密度構造が測られてきた。そのような天体での星形成をシミュレートし、重力収縮の計算を行う。データの観測的な可視化についても勉強する。

５．太陽風磁気圏相互作用の３次元 MHD シミュレーション　　荻野竜樹、中尾真季

ベクトル化とベクトル並列化（MPI）された３次元 MHD コードを用いて、SUN ワークステーションとベクトル並列型のスーパーコンピュータ Fujitsu VPP5000 で太陽風と地球磁気圏相互作用のグローバル MHD シミュレーションを行い、惑星間磁場（IMF）が北向きと南向きの場合の磁気圏構造を調べる。図形処理として、断面図や３次元磁力線構造の描画、VRML を用いた３次元可視化を実行する。

3D-MHD, Modified Leap-Frog scheme, 並列化コード（MPI）