補足

1. 名古屋大学情報メディア教育センターの利用について

受講者アカウント利用可能期間：９月２日（月）～９月２０日（金）
受講者アカウント：　02act***　（*** は３桁の数字）
パスワード変更後使用：後の説明参照

受講者アカウントのディスク使用量の上限：３０ＭＢ
シミュレーション結果などの大量データ：大容量のディスクに保管
（一人当たり約５ＧＢ使用できる）

1.1. 名古屋大学情報メディア教育センターシステムのインターネットからの利用について

情報メディア教育システムは、NICE（名古屋大学キャンパスネットワーク）と民間プロバイダによるインターネットの 2 個所に接続されています。これらのインターネットとの接続点にはファイアーウォール装置とルータが設置されています。これらの機器は、インターネットで使用されるグローバルアドレスをメディアセンター内部のプライベートアドレス(172.16.zzz.www)に変換するとともに、いろいろなプロトコルに対してフィルターをかけ、外部からの招かれざるアクセスや、外部への無駄なアクセスを防止します。

1.1.1. メディアセンター内部から外部への通信

原則としてメディアセンター内部から、NICE へはすべてのプロトコルで個別通信が可能、民間プロバイダ向けには限定されたポートで通信が可能です。

1.1.2. 外部からメディアセンター内部への通信

NICE やインターネットからメディアセンターに対して利用可能なのは以下のプロトコル（サービス）です。

1.1.3. プロトコル（NICE からとインターネットから）

telnet （NICE からは）telnet.media.nagoya-u.ac.jp に telnet してください。
(media.nagoya-u.ac.jp には telnet しないでください。ログインできません。)
telnet.media.nagoya-u.ac.jp が反応しない場合は、
telnet1.media.nagoya-u.ac.jp あるいは telnet2.media.nagoya-u.ac.jp
を試してください。
（インターネットからは）利用できません。ssh をお使い下さい。

rlogin　利用できません。

ssh (Secure Shell) telnet サーバ(telnet.media.nagoya-u.ac.jp または
telnet.rt2.media.nagoya-u.ac.jp)のみに接続可能です。 NICE から利用する場合は、必ず、telnet.media.nagoya-u.ac.jp を使ってください。
telnet.media.nagoya-u.ac.jp が接続できないときは、
telnet1.media.nagoya-u.ac.jp または telnet2.media.nagoya-u.ac.jp を 試してください。

ftp （NICE からは）ftp サーバ(ftp.media.nagoya-u.ac.jp)のみに接続可能です。
ftp.media.nagoya-u.ac.jp がつながらない場合は、

ftp1.media.nagoya-u.ac.jp または ftp2.media.nagoya-u.ac.jp を試してください。
（インターネットからは）利用できません。ssh(scp)をお使い下さい。

1.2. 名古屋大学情報メディア教育センターシステムの Solaris で利用できる Workshop （コンパイラ）について

1.2.1. センター端末からの Workshop の起動は 起動メニューから行うことができます。

コマンドラインから起動できるコマンドは /opt/SUNWspro/bin にあります。（ホストによってはこのディレクトリが見えない場合があります。その場合は sv010 に rlogin してください。）その際、コンパイラ契約の関係上以下の制限があります。

シェルで C コンパイラ、FORTRAN コンパイラを利用するには以下のような環境設定をします。

コマンドサーチパス に /opt/SUNWspro を加える

(/usr/ucb より前に加える)
環境変数 LD_LIBRARY_PATH に /opt/SUNWspro/lib を加える
環境変数 LANG を無効にする

例えばＣシェルを使用している場合、以下のような設定を ~/.cshrc.local に追加します。

```
set path=(/opt/SUNWspro/bin $path)
setenv LD_LIBRARY_PATH /opt/SUNWspro/lib:$LD_LIBRARY_PATH
unsetenv LANG
```

ただし、環境変数 LANG を無効にすると日本語オンラインマニュアルが読めません。 日本語オンラインマニュアルを利用する場合は、.cshrc.local に設定を書かずに、コンパイラを使用するシェルのみで LANG を無効にしてください。この場合、コマンドラインで
’unsetenv LANG’を実行します。

1.3. 名古屋大学情報メディア教育センターシステムの利用期間について

講師と受講者が、情報メディア教育センターシステムをサマースクールで利用できる期間は次のように予定しています。

講師： 平成１４年８月２０日～９月２０日
受講者：平成１４年９月 ２日～９月２０日

1.4. 名古屋大学情報メディア教育センターシステムのコンパイラについて

情報メディア教育センターシステムでは次のフォートランコンパイラとＣコンパイラが使用できます。サマースクールで計算機実習使用する部屋は C,E 教室です。

(a) A・D(717 号)・E・W 教室(日本語 Solaris2.6 H/W 5/98 SPARC)
(1) Fujitsu 版

・Fujitsu Fortran Compiler4.0
・Fujitsu C Compiler4.0
・Fujitsu C++ Compiler4.0

(2) Sun 版
・Sun WorkShop Compiler FORTRAN 4.2
・Sun WorkShop Compiler C 4.2
・Sun WorkShop Compiler C++ 4.2

(b) C 教室(日本語 Solaris2.6 for x86 INTEL)

(1) Sun 版
・Sun WorkShop Compiler FORTRAN 4.2
・Sun WorkShop Compiler C 4.2
・Sun WorkShop Compiler C++ 4.2

使用例： frt -O program.f
frt -o execute -O program.f
f90 -O program.f
f90 -o execute -O program.f
f77 -O program.f
f77 -o execute -O program.f

1.5. コマンドラインからのパスワードの設定方法

パスワードの変更にあたっては以下の点に留意してください。

1.新しいパスワードは旧パスワードと 3 文字以上変っている必要があります。
2.パスワードの変更は頻繁におこなわないでください。変更後、パスワードが変更されるまでに最大 10 分かかります。 また、システム内の処理が確定するまで、1 時間程度は変更しないようにしてください。

注意: パスワードの変更については以下のルールに従ってください。現状では、以下のルールに従っていなくてもパスワードが変更されたように見えることがあります。
（が、実際は変更されていない）

6 文字以上のパスワードをつけてください。
２つ以上の英文字および一つ以上の数字もしくは特殊文字を含めてください。
ログイン名そのもの、あるいはその文字順を入れ替えたパスワードは不可

センター端末から変更する場合(UNIX と Windows/NT の構成)

1.Unix(Solaris)の動作しているマシンにログインします。このとき、昨年度までのアカウントとパスワードが使えます。
2.「端末エミュレータ」を開いて以下のコマンドを実行します。

/opt/thrpassop/bin/tpassset

ここで、を自分の ID でおきかえてください。
例えば、アカウントが t901234h の人は、

/opt/thrpassop/bin/tpassset t901234h

のように入力してください。このあと、古いパスワードと新しいパスワードの入力(2 回)を求められますので、入力してください。

外部から接続する場合

telnet サーバ(telnet.media.nagoya-u.ac.jp)にログインしてください。
telnet サーバでは SSH がインストールされていますので、可能な場合はパスワード をのぞかれないように SSH を利用してください。
telnet サーバで上と同様のコマンド(/opt/thrpassop/bin/tpassset)を入力してください。

この方法でくりかえしパスワードを変更することができます。
以下のページからも変更できます。
http://pass.media.nagoya-u.ac.jp/tpass/default.htm
このページはメディアセンター端末およびリモートアクセスシステムからアクセスできます。

1.6. 実際の利用例

(1) sv010 に rlogin する。
rlogin sv010
(2) 講師のアカウント 02act132 のディレクトリ earthb から、プログラムファイルを受講者のディレクトリにコピーする。
cp /mdhome/home2/02act132/earthb/earthb1.f .
(3) Fortran プログラム earthb1.f をコンパイルする。
frt -O earthb1.f ：実行ファイル a.out を生成
frt-o earthb1 -O earthb1.f ：実行ファイル earthb1 を生成
(4) a.out (earthb1) ：実行ファイル a.out (earthb1)を実行
(5) a.out & (earthb1 &) ：実行ファイル a.out (earthb1)をバックグラウンドジョブとして実行
(6) gs earthb1.ps ：PostScript file, earthb1.ps を GhostScript でプレビューする。
(7) lpr earthb1.ps ：PostScript file, earthb1.ps をプリントする。

1.7. 更に詳細は次の URL を参照のこと

情報メディア教育センターの Homepage
http://www.media.nagoya-u.ac.jp/
情報メディア教育センターシステムの利用法
http://www.media.nagoya-u.ac.jp/riyou.html

---

2. 名古屋大学情報連携基盤センター計算機システムの利用について

受講者アカウント利用可能期間：９月９日（月）～９月２０日（金）
受講者アカウント： w49***a （*** は３桁の数字）
パスワード変更後使用：パスワードはスクールの初日（9/9）に変更します。
パスワード変更は、後の説明参照のこと。

2.1. 名古屋大学情報連携基盤センター計算機システムの利用期間について

講師と受講者が、情報メディア教育センターシステムをサマースクールで利用できる期間は次のように予定しています。利用するためには予め登録が必要です。

講師： 平成１４年８月２０日～９月２０日
受講者：平成１４年９月 ９日～９月２０日

2.2. 情報連携基盤センター計算機システムの利用について

名古屋大学情報連携基盤センター（旧大型計算機センター）のベクトル並列型スーパーコンピュータ、Fujitsu VPP5000/64（vpp）の 2PE の２セットとフロントエンドの SUN（gpcs）を利用することができます。

Fujitsu VPP5000/64（vpp） vpp.cc.nagoya-u.ac.jp (133.6.90.2)
front-end-processor （gpcs） gpcs.cc.nagoya-u.ac.jp (133.6.90.3)

これらは、MPI (Massage Passing Interface), VPP Fortran, HPF (High Performance Fortran) などの並列プログラミングや並列ジョブの実行に利用します。

以下の情報連携基盤センター計算機システムの利用方法の説明は、名古屋大学太陽地球研究所の計算機利用共同研究者のためのより一般的な説明で、まだサマースクール用に書き換えていません。サマースクールでは並列化のために 2PE のみを利用しますので、その点を考慮して参照・利用して下さい。

(I) How to Use Computer System in the Nagoya University Computer Center

(0) To use gpcs.cc.nagoya-u.ac.jp (133.6.90.3), front-end-processor (SUN workstation) of Fujutsu supercomputer VPP5000/64 (vector-parallel machine) or to use vpp.cc.nagoya-u.ac.jp (133.6.90.2) of Fujutsu supercomputer VPP5000/64 itself.

how to change password
gpcs% yppasswd
Old yp password: present password
New password: new password
Retry new password: new password

(1) How to connect initially
telnet gpcs.cc.nagoya-u.ac.jp (or 133.6.90.3)
: You will connect Front end processor, gpcs to VPP5000/64
You can use usual UNIX commands

cdvpp:　transfer to desk area for VPP5000

(2) How to use
The following is your directory which you first get in (not VPP desk area)
```
gpcs% pwd
/home/usr7/l46637a　: disk area of front end processor (gpcs)
```
Change to VPP desk area use "cdvpp"
```
gpcs% pwd
/vpp/home/usr7/l46637a : disk area of superprocessor (vpp)
```

2.3.　サマースクールでの利用方法

サマースクールでは、情報メディア教育センターのコンピュータから、情報連携基盤センターの gpcs や vpp に telnet などで接続して利用します。大量のシミュレーションデータは名古屋大学情報メディアセンターの大容量のディスク（サマースクール用に導入）にファイル転送・保管（約５ＧＢ／人）して、図形処理などを行うこと。

サマースクール用の gpcs や vpp での実行キューは　s1　クラスのみを使用する。

(1) gpcs や vpp に login する。（受講者アカウント：w49000a の場合）
cdvpp　：gpcs のディスクから vpp のディスクに移動する。
mkdir mearthb　：mearthb のディレクトリを作成
cd mearthb　：mearthb のディレクトリに移動

(2) 情報メディア教育センターの講師アカウントからプログラム progmpi.f を ftp で取得
/vpp/home/usr7/w49000a/mearthb の中に get (put)で転送

(3) TSS でコンパイルして実行（vpp で実行することが必要）
mpifrt progmpi.f　：MPI でコンパイル
jobexec -vp 2 ~/mearthb/a.out　：2PE を用いて TSS で実行

(4) Batch でコンパイルして実行（gpcs で実行可）
qsub -q c -eo -o pconpmpi2.out pcompmpi2.sh　：MPI でコンパイル、並列化情報在
qsub -q c -eo -o pconpmpi.out pcompmpi.sh　：MPI でコンパイル
qsub mpi_lim02e.sh　：2PE を用いて Batch で実行（s1 クラス）

<<pcompmpi2.sh>>
```
cd mearthb
mpifrt -Lt progmpi.f -Pdt -o progmpi -Z mpilist
```

<<pcompmpi.sh>>
```
cd mearthb
mpifrt -o progmpi progmpi.f
```

<<mpi_lim02e.sh>>
```
#　@$-q s1　-eo -o pexecmpi02.out
#　@$-lP 2
setenv　VPP_MBX_SIZE　256000000
./mearthb/progmpi -np 2
```

2.4. 更に詳細は次の URL を参照のこと

情報連携基盤センターの Homepage
http://www2.itc.nagoya-u.ac.jp/
スーパーコンピュータ VPP5000/64 利用案内
http://www2.itc.nagoya-u.ac.jp/sys_riyou/vpp/vpptebiki.htm

実習の手引き (差分法、MHD 実習)

福田尚也

2002.9

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
```

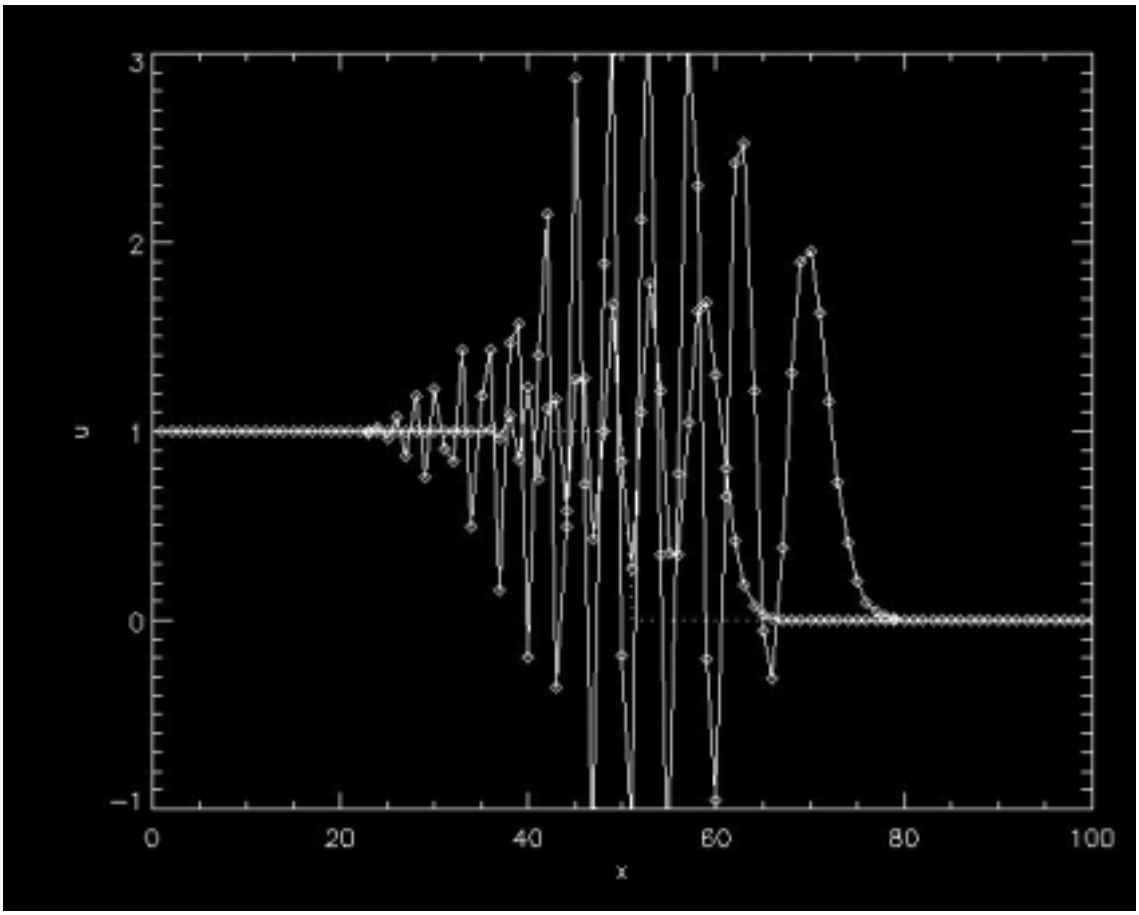


図 1: scalar パッケージの計算結果の例

1.3.4 IDL の終了 (end)

end を入力すると IDL を終了することができます。

```
IDL> end
```

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。

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
```

Version 2.0 (2002/08/12) 福田尚也

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、章末のリストや以下の Web ページを参考にして下さい。

CANS 1D デモページ

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

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)
```

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 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 ###
```

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
```

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
```

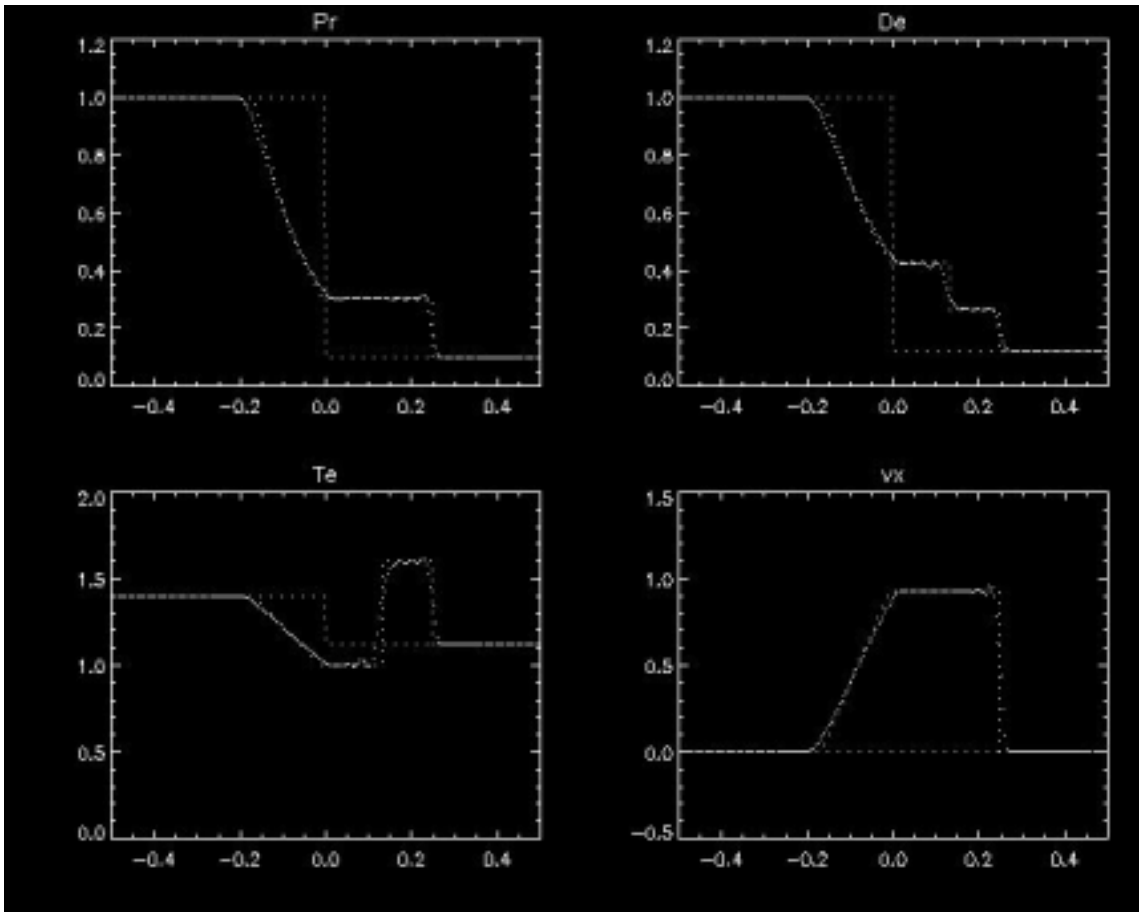


図 1: md_shktb の結果

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て下さい。

```
IDL> .r anime
```

1.3.5 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
```

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。

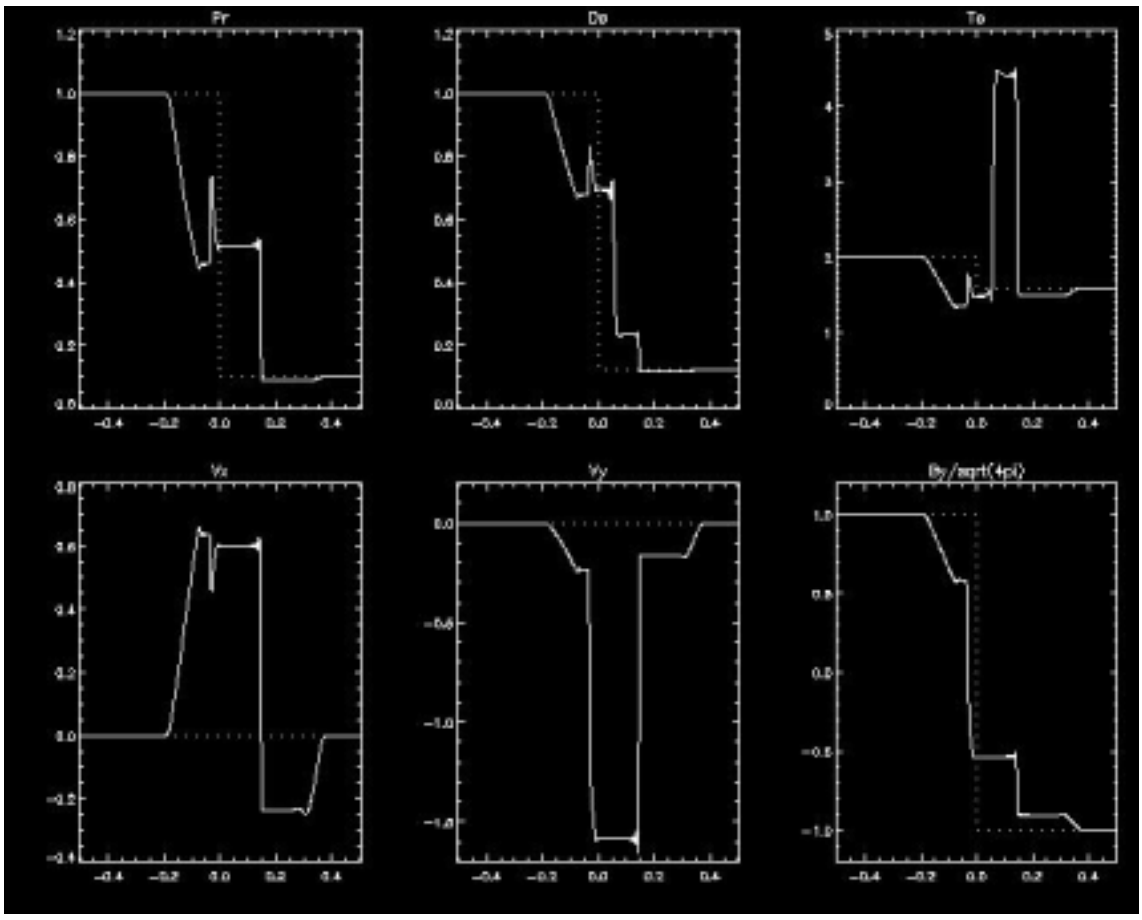


図 2: md_mhdshktb の結果

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 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毎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.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。

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 >>>
         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 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         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   <<<
```

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-----------------------------------------------------------------------|
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 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 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。

CANS 1D モジュール一覧 (02/08/12 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（時間精度 1 次：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（時間精度 1 次：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

ドキュメント

README
htdocs/ HTML 形式のドキュメント
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

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]
```

境界条件モジュール

```
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。ただし微分一定。　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 パッケージの説明

CANS 2D 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。

CANS 2D デモページ

http://www.astro.phys.s.chiba-u.ac.jp/netlab/cans/movie2/frame.html

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)。

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 -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。

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
```

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 IDL の終了 (end)

IDL を終了してみましょう。

```
IDL> end
```

CANS 2D モジュール一覧 (02/08/12 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 (開発中)
hdcip/ CIP 法で流体計算をするためのモジュール (開発中)

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
nc/ netCDF フォーマットでデータ出力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
cndbicg/ 熱伝導を陰解法（時間精度 1 次：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 (開発中)

commonmpi/ 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/ 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)

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_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_mhdrt/ MHD Rayleigh-Taylor 不安定性 [MHD、重力]
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（Balbus-Hawley）不安定 [MHD、潮汐 Coriolis 力]
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_diskflare/ 降着円盤フレア [MHD、円柱座標、抵抗]
md_cme/ 太陽コロナ質量放出：Low 解 [MHD、球座標]
md_sg/ 自己重力テスト [自己重力]
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、自己重力、円柱・球座標]

名古屋大学情報連携基盤センター

スーパーコンピュータ VPP5000 利用の手引

2002 年 4 月版

～　目次　～

１．スーパーコンピュータ VPP5000 の特徴

スーパーコンピュータ VPP5000 は、分散メモリ型のベクトル並列計算機です。VPP5000 の特徴を以下に列挙します。

・PE 台数：64（8 台は IOPE）
・PE の理論最大性能 9.6Gflops
・総メモリ容量 1024GB
・1PE のメモリ容量：16GB
・クロスバネットワークのデータ転送速度：最大 1.6GB／秒
・スカラ演算の性能向上
・4 倍精度演算の高速化

２．VPP5000 のソフトウェア

VPP5000 で利用できるソフトウェアの一覧を表１に示します。

表 1：VPP5000 のソフトウェア一覧

種類	ソフトウェア名
プログラミング言語	Fortran (JIS X3001-1:1998 準拠)
	C (ANSI X3.159:1989 準拠)
	C++（USL release 3.0 準拠）
並列処理言語	VPP Fortran
	HPF (High Performance Fortran Forum 仕様 V2.0 準拠)
	DPCE (Data Parallel C Extension)
数値計算ライブラリ	SSLⅡ（科学用サブルーチンライブラリ）
	C- SSLⅡ（科学用サブルーチンライブラリ）
	NUMPAC（数値計算ライブラリ）
	BLAS（ベクトル処理向き線形計算ライブラリ）
	LAPACK（ベクトル処理向き線形計算ライブラリ）
	ScaLAPAC（MPI 並列処理向き線形計算ライブラリ）
メッセージパッシングライブラリ	MPI（MPI2.0 仕様準拠）
	PVM（PVM3.3 仕様準拠）
アプリケーション	α－FLOW（汎用 3 次元流体解析システム）*
	STAR-CD（非構造格子汎用熱流体解析ソフトウェア）
	POPLAS/FEM5（有限要素法による構造解析プログラム）
	LS-DYNA3D（非線形動的構造解析ソフトウェア）
	MASPHYC-2，MASPHYC-SP（材料設計システム）*
	Gaussian98（分子軌道計算プログラム）*
	AMBER（分子構造計算プログラム）*
	FastDNAml（最尤法による進化系統樹推定プログラム）
	VisLink（可視化ソフトウェア）
	MOLPRO（分子軌道計算プログラム）

*印のアプリケーションは、並列版が用意されている。

３．利用形態とジョブ種別

スーパーコンピュータ VPP5000 と汎用計算サーバおよびファイルサーバのシステム概略を図１に示します。

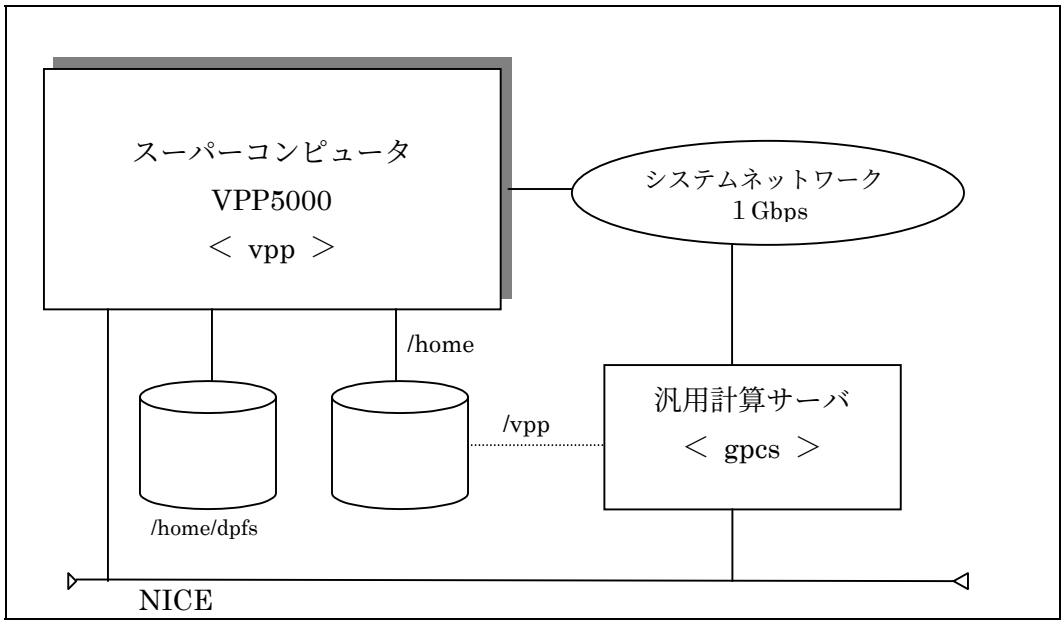


図１：スーパーコンピュータ VPP5000 のシステム概略

図１に示すようにスーパーコンピュータは、ネットワークから直接 TELNET 接続して利用可能です。また、汎用計算サーバ gpcs から NQS により利用することも可能です。

スーパーコンピュータで利用できるファイルは、2 種類あります。一つは、スーパーコンピュータに login したときのホームディレクトリのファイルで、/home のファイル[1]です。このファイルは、汎用計算サーバ gpcs からは、/vpp のディレクトリとして見えます。このファイルは、fpfs(Flexible and high performance file system)と呼ばれるファイルシステムで、小容量のファイルから大容量のファイルまでを高速に処理することができます。一方、/home/dpfs のディレクトリのファイルは、大容量のファイルや並列ジョブからの入出力に適したファイルシステムです。dpfs の利用については、１１章を参照してください。

スーパーコンピュータ VPP5000 の利用形態には、バッチジョブを投入して利用する方法と TSS で会話的に利用する方法があります。実行が長時間にわたるプログラムはバッチ処理がお薦めです。一方、短時間で終了するものや、翻訳処理やデバッグは TSS が便利です。

バッチジョブの投入経路としては、

1. スーパーコンピュータの TSS からコマンドによりバッチジョブを投入する。
2. 汎用計算サーバ gpcs から NQS のコマンドによりバッチジョブを投入する。
3. 本センターのホームページにアクセスし、Web ブラウザからバッチジョブを投入する。

[1] pwd コマンドを実行するとホームディレクトリが、*/G/pe0M/usrN/login-name* と表示される場合がありますが、ホームディレクトリを指定する場合には、*/home/usrN/login-name* と指定してください。

があります。どの経路を選択するかは、各自の利用環境にもよりますが、スーパーコンピュータだけを利用する場合には、1.の経路が、ポスト処理等で汎用計算サーバを利用する場合には、2.の経路が良いでしょう。3.の経路は、手元のパソコンから利用し、UNIX のファイル編集に馴染みの薄い利用者には便利な方法です。

スーパーコンピュータ VPP5000 のホスト名は、vpp.cc.nagoya-u.ac.jp[2]、汎用計算サーバのホスト名は、gpcs.cc.nagoya-u.ac.jp です。本センターのホームページの URL は、http://www.cc.nagoya-u.ac.jpです。

スーパーコンピュータで利用可能な PE 数、CPU 時間やメモリ量などの計算資源はジョブの種別により異なります。表２にジョブ種別を示します。

表２：スーパーコンピュータ VPP5000 のジョブ種別

利用形態	キュー名	使用可能 PE 数	CPU 使用時間	メモリサイズ		経過時間	利用目的
				標準値	制限値	標準値	
バッチジョブ	c	1	60 分	500MB	2GB	―	非並列ジョブ
	x	1	1200 分	2GB	15.5GB	―	非並列ジョブ
	z	2～16	600 分	1PE あたり 2GB	15.5GB	―	並列ジョブ
	ze	17～32	600 分	1PE あたり 2GB	15.5GB	―	並列ジョブ
TSS	―	1	60 分	500MB	2GB	―	会話型非並列
	―	1～8	300 分	1PE あたり 2 GB	15.5GB*	720 分	会話型並列

＊）使用 PE 数が 5 以上の場合には、1 台の PE に 2 個の仮想プロセッサを割り当てるため、使用できるメモリの最大は、7.5GB の 1／2 になります。

４．翻訳コマンド

VPP で利用できる言語と翻訳コマンド[3]を表３に示します。なお、frt コマンドでは、Fortran95 の言語仕様が標準で動きます。

表３：翻訳コマンド

言語	コマンド
Fortran/VP(ベクトル化)	frt
Fortran/VPP(並列化)	frt -Wx
C	cc
C/VP	cc -Kvp　　または vcc
C++/VP(ベクトル化)	CC
C++	CC -Knovp

翻訳コマンドのオプションは、VPP に login して、man コマンドで確認してください。

[2] vpp.cc.nagoya-u.ac.jp に TELNET 接続した場合、端末への表示は、"UXP/V TELNET (pe2c)となります。
[3] 汎用計算サーバ gpcs で VPP 用のクロスコンパイルができます。詳細は、14 章を参照してください。

５．ライブラリの利用

ライブラリを使用する場合には、frt コマンドの-l オプションの指定が必要です。使用するライブラリと指定するオプションを表４に示します。

表４：ライブラリとオプション

ライブラリ名	オプション
NUMPAC：数値計算ライブラリ	-l numpac
SSLⅡ：科学用サブルーチンライブラリ	-l ssl2vp
SSLⅡ：科学用サブルーチンライブラリ並列版	-l ssl2vpp
SSLⅡ：科学用サブルーチンライブラリ HPF 版	-l ssl2hpf
ユーザ登録のセンターライブラリ	-l ulib
図形出力ライブラリ	-l ps
BLAS（ベクトル処理向き線形計算ライブラリ）	-l blasvp
LAPACK（ベクトル処理向き線形計算ライブラリ）	-l lapackvp -l blasvp

６． バッチジョブの投入

NQS のコマンドによるバッチジョブの投入手順を以下に示します。

1. 実行するコマンドをバッチリクエスト（スクリプトファイル）としてファイルに作成する。
2. qsub コマンドでバッチリクエストの実行依頼をする。
3. qstat コマンドでジョブの実行状況の確認をする。
4. ジョブの実行が終了していたら、出力結果のファイルをチェックする。

6.1 バッチリクエスト（スクリプトファイル）の作成例と実行依頼

以下の例では、スクリプトファイルは nqs のディレクトリに作成するものとする。なお、ここでは qsub のオプションはスクリプトファイルに記述する。qsub のオプションについては 6.4.1 を参照されたい。

6.1.1 1PE 用のモジュールの実行（メモリ 2GB 以内[4]）

実行キューとして x を指定する。

1) スクリプトファイル nqs/exec_s.sh の内容

実行可能ファイル名：pg/vtest_s

```
#  @$-q x  -eo  -o vtest_s.out
cd  pg
./vtest_s
```

2) 1PE 用モジュールの実行依頼

スクリプトファイル:exec_s.sh

qsub コマンドで依頼する。

```
vpp% qsub exec_s.sh
```

[4] 実行時にメモリが不足した場合には、” Not enough space “のメッセージが出力される。

6.1.2 1PE 用のモジュールの実行（メモリ 2GB を超え 15.5GB 以内）

実行キューとして x を指定し、qsub の-lM オプションで必要メモリ量を指定する。

1）スクリプトファイル nqs/exec_s5g.sh

実行可能ファイル名：pg/vtest_s5g

```
#  @$-q x  -eo  -o vtest_s.out  -lM 5gb
cd  pg
./vtest_s5g
```

2）1PE 用モジュールの実行依頼

スクリプトファイル:exec_s5g.sh

qsub コマンドで依頼する。

```
vpp% qsub exec_s5g.sh
```

6.1.3 並列処理用モジュールの実行

並列化処理を行う場合には、実行キューとして z（16PE まで利用可能）または ze（32PE まで利用可能）を指定する。PE 数の指定は、 -lP オプションで行う。使用するメモリが 2GB を超える場合には、-lM オプションで必要メモリ量を指定する。以下の例では、実行の終了をメール[5] で通知するように-me オプションを指定している。また、 -lT オプションに使用する CPU 時間（この例では 1 時間 30 分）を指定している。

3-1）スクリプトファイル nqs/exec_p5.sh の内容

実行可能ファイル名：pg/vtest_p5

```
#  @$-q z  -lP 5  -eo  -o vtest_p5.out
#  @$-me  -lT 1:30:00    -lM 5gb
cd  pg
./vtest_p5
```

3-2）並列用モジュールの実行依頼

スクリプトファイル:exec_p5.sh

qsub コマンドで依頼する。

```
vpp% qsub exec_p5.sh
```

6.2 スーパーコンピュータでのジョブ投入

スーパーコンピュータ VPP5000 ではバッチジョブの投入に関して、次に示すコマンド[6]が利用できます。

バッチリクエストの実行依頼	qsub コマンド
バッチリクエストの実行状態の確認	qstat コマンド

---

[5] qsub を実行しているホスト以外のシステムにメールで通知したい場合には、 qsub を実行しているホストの.forward ファイルにメールアドレスを記述する。

[6] これらのコマンドは、/usr/local/bin にあります。これらのコマンドの動きがおかしい場合には、which qsub としてコマンドのパス名を確認してください。

バッチリクエストの削除　　qdel コマンド

実行中のバッチリクエストの詳細情報の表示　　jstat コマンド

上記コマンドのうち qsub、qstat、qdel コマンドの使用方法とオプションは、NQS のコマンドと同じです。コマンドのオプションの詳細については、man コマンドで確認してください。

実行中のジョブの状況を見るために jstat コマンドが用意されています。jstat コマンドでは、実行中のバッチリクエストの経過時間、CPU 時間、VU 時間、メモリ量、I/O 量が確認できます。なお、並列処理では、表示される CPU 時間と VU 時間はいずれも最大のものです。

（使用例）

```
vpp% qsub  exec_p.sh
Request 291.pe00 submitted to queue: x.
vpp% qstat x
x@pe00;  type=BATCH;  [ENABLED, RUNNING];  pri=31
  0 exit;   1 run;   0 queued;   0 wait;   0 hold;   0 arrive;
    REQUEST NAME   REQUEST ID    USER   PRI    STATE  JOB-ID  PHASE
 1:      exec     291.pe00    a49999a  31  RUNNING     137  RUN
vpp% jstat
Job Infomation on vpp5000 (ver 1.00)     2000/04/19-15:27
------------------------------------------------------------
QUEUE USER REQUEST-No  PE   ELAPS     CPU      VU       %    MEM   I/O
------------------------------------------------------------
x  a49999a   291.pe00  1  0:55:25  0:50:18  0:43:08  88  384MB 84KB
------------------------------------------------------------
vpp% qdel -k 291.pe00
Request 291.pe00 is running, and has been signalled.
vpp% qstat x
x@pe00;  type=BATCH;  [ENABLED, INACTIVE];  pri=31
```

【留意点】TSS から qsub コマンドでジョブの実行依頼をすると、標準出力ファイルに下記のメッセージが出力されますが、実行には影響ありませんので、無視してください。

```
Warning: no access to tty; thus no job control in this shell…
```

6.3 汎用計算サーバでのジョブ投入

スーパーコンピュータで使用するファイルはすべて/vpp/home/usr*N*/*login-name* のディレクトリの下に作成してください。gpcs の cdvpp コマンドにより/vpp のディレクトリに移動できます。

バッチリクエストの実行依頼　　qsub コマンド

バッチリクエストの実行状態の確認　　qstat コマンド

バッチリクエストの削除　　qdel コマンド

汎用計算サーバ gpcs から qstat、qdel コマンドを実行する場合にはシステム名も付けて指定してください。

【例】gpcs% qstat　x@vpp-g　　キューx の状態表示

gpcs% qdel -k -r vpp-g　リクエスト ID　　バッチリクエストの削除

6.4 NQS のコマンド

6.4.1 qsub コマンド

機能：バッチリクエストの実行依頼をする。

形式： qsub [オプション] [スクリプトファイル]

qsub コマンドの主なオプション：

-q　バッチリクエストを依頼するキュー名を指定する。キュー名については表 2 を参照のこと。

-e　標準エラー出力ファイルを指定する。

-o　標準出力ファイルを指定する。

-eo　標準エラー出力を標準出力ファイルに出力する。-e を指定した場合にはこのオプションは指定できない。

-lP　並列処理の場合に使用する PE 数を指定する。

-lT　使用する CPU 時間を指定する。
時間の記述形式は、[[時間：]分：]秒[.ミリ秒]である。

-lM　使用するメモリ量を指定する。記述の形式は、整数[.小数]gb である。

-mb　バッチリクエストの実行を開始したときにメールを送る。

-me　バッチリクエストの実行を終了したときにメールを送る。

-mu　バッチリクエストに関するメールを指定ユーザに送る。

-s　バッチリクエストシェルスクリプトを解釈するのに用いるシェルのフルパスを指定する。デフォルトでは C シェルが用いられる。B シェルを用いるときは次のように指定する。

-s /usr/bin/sh

使用例：

```
qsub　-q キュー名　スクリプトファイル名
```

なお、スクリプトファイル内には qsub コマンドのオプションも記述することができます。qsub コマンドのオプションの詳細は、man コマンドで確認してください。

6.4.2 qstat コマンド

機能：バッチリクエストの実行状態を確認する。

形式： qstat [キュー名]

オプションの説明：

汎用計算サーバ gpcs からキューの状態を表示する場合はシステム名も付けて以下の形式で指定する。

x@vpp-g キューx の状態を表示

z@vpp-g キューz の状態を表示

使用例：

```

```

qstat　x@vpp-g

6.4.3　qdel コマンド

機能：バッチリクエストの削除

形式： qdel　[-k|-シグナル番号]　[-r システム名]　リクエスト ID

オプションの説明：

-k | -シグナル番号

-k　　指定したパッチリクエストのうち走行中のバッチリクエストに対してシグナル SIGKILL を送信する。

-シグナル番号　指定したパッチリクエストのうち走行中のバッチリクエストに対してシグナル SIGKILL 以外のシグナルを送信する。

このオプションを指定しない場合は走行待ちのバッチリクエストを削除する。

-r　システム名

汎用計算サーバ gpcs から行う場合は、システム名として vpp-g を指定する。

使用例：

```
qdel  -k  -r vpp-g  リクエスト ID
```

【注意】バッチリクエストが swap out されている場合には、 qdel コマンドが効かない旨のメッセージが表示されるが、そのバッチリクエストが swap in された時点で、 qdel コマンドが有効となり、バッチリクエストは削除される。

【qstat コマンドと qdel コマンドの使用例】

```
gpcs% qstat x@vpp-g
 x@pe00;  type=BATCH;  [ENABLED, RUNNING];  pri=31
   0 exit;   1 run;   0 queued;   0 wait;   0 hold;   0 arrive;
       REQUEST NAME     REQUEST ID      USER PRI    STATE  JOB-ID  PHASE
    1:   exec_s.sh  70494.gpcs      a49999a  31  RUNNING     173  RUN
 gpcsf% qdel -k -r vpp-g 70494.gpcs
 Request 70494.gpcs is running, and has been signalled.
```

6.4.4　qcat コマンド

機能：バッチリクエストのスクリプトファイル、出力ファイルの表示

形式： qcat　[-e|-o|-s]　[-u ユーザ名]　リクエスト ID

オプションの説明：

-e　　リクエストの標準エラー出力ファイルを表示。

-o　　リクエストの標準出力ファイルを表示。

-s　　リクエストのスクリプトファイルを表示。

-e、-o、-s オプションはいずれか一つのみが指定可能。これらのオプションを省略した場合には、-o が指定されたものとみなされる。

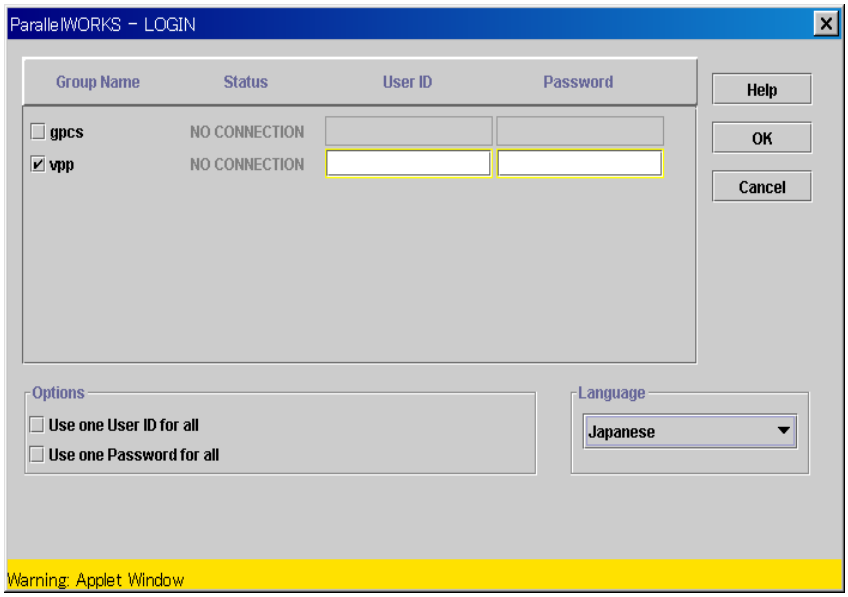
6.5　Web ブラウザからのジョブ投入

本センターのホームページからスーパーコンピュータへジョブを投入することができます。このウィンドウでは、ジョブ投入の他にジョブの実行状況の表示、ファイルの編集、ファイルの内容表示、ファイルの複写・移動、ファイルの消去、ブラウザを起動しているパソコンとの間でのファイルのアップロード／ダウンロードなどもできます。パソコンをホームにして仕事をしている利用者には便利な機能です。ジョブの投入手順を次に示します。

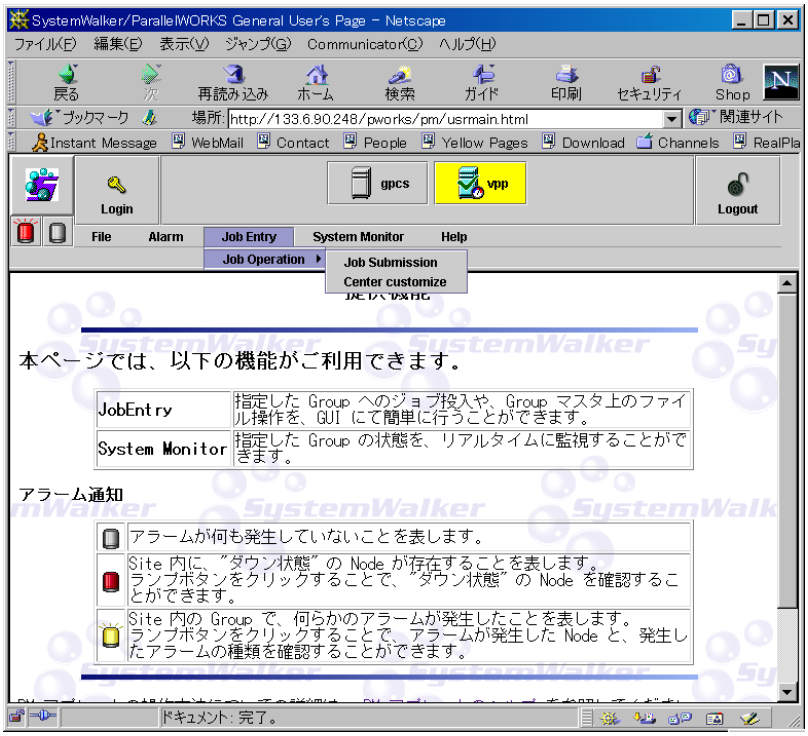
1）「システム利用案内」の「スーパーコンピュータ VPP5000/64 利用案内」の項目をクリックする。

2）「Web ブラウザからのジョブ投入」をクリックする。このページを表示するには Plug-in が必要である。Plug-in が必要な場合は、該当するアイコンをクリックして Plug-in のソフトを入手する。なお、汎用計算サーバ gpcs で netscape を立ち上げる場合には、実行に先立ち `mkplugin` コマンドを実行すると Plug-in の環境が整う。

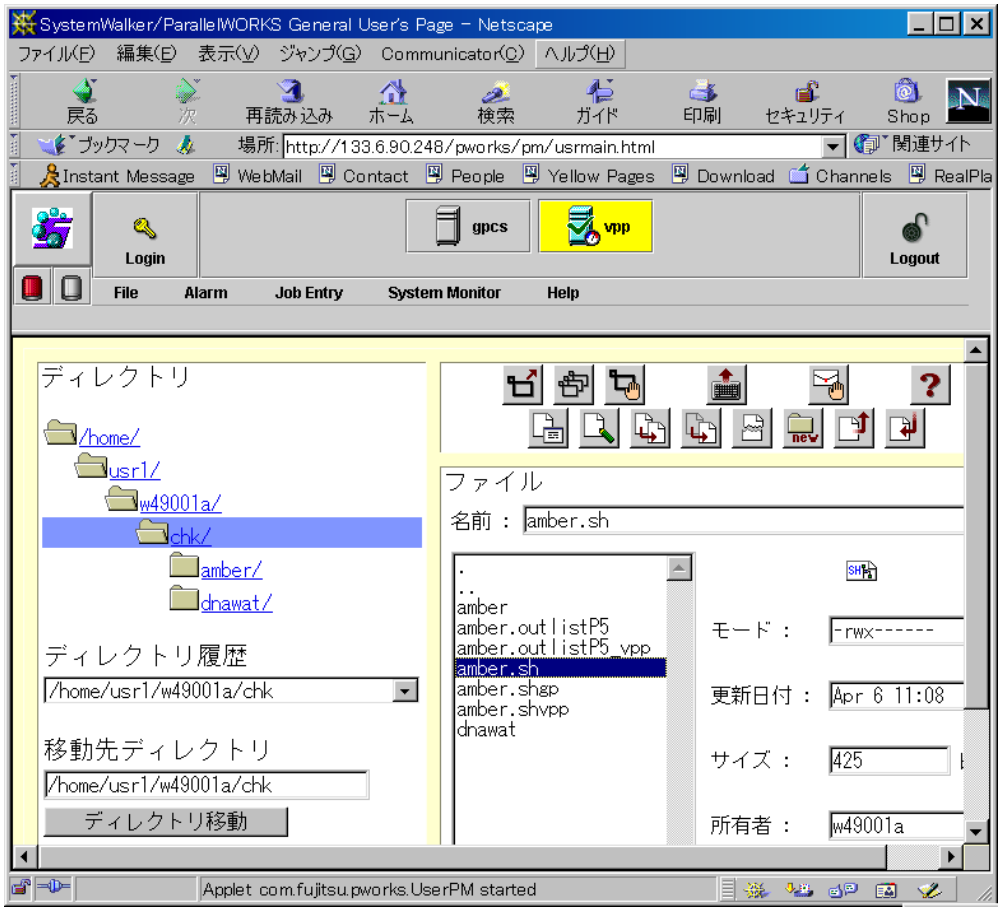
3）次に示すウィンドウが表示されるので、”vpp”の項目をチェックし、”User ID”、”Password”の項目を入力する。更に”Language”を”Japanese”にして、”OK”をクリックする。



4）次に示すウィンドウが表示されるので、”Job Entry”の”Job Submission”の項目をクリックする。”Job Entry”の項目がクリックできないときは、その上の”vpp”のアイコンをクリックする。



5）次に示すウィンドウが表示される。このウィンドウで、ファイルを選択し、ウィンドウ右上の"Submit job"のアイコンを選択することにより、スーパーコンピュータにジョブを投入することができる。このとき、指定したファイル名のサフィックスが.shの場合には、qsub コマンドが実行され、ジョブの投入が完了する。ファイル名のサフィックスが.sh 以外のときは、指定したファイルはロードモジュール(実行可能ファイル)と判断され、実行依頼を行うためのシェルスクリプトファイルを作成するウィンドウが表示されるので、必要な項目を入力する。



6）“Operate own job”のアイコンで、投入したジョブの状況を表示する。

7）ジョブが終了したら、”Browse file”のアイコンで結果を確認する。結果のファイルをブラウザを起動しているパソコンにダウンロードする場合には、”Transfer file to browser computer”のアイコンでファイルを転送する。

8）すべての処理が終了したら、”logout”のアイコンをクリックする。

7．TSS での実行

7.1 TSS での実行環境

VPP5000 で利用できるエディタ[7]には以下のものがあります。

コマンド名	エディタ名	備考
emacs	Emacs-20.4	日本語入力システムは SKK。
ng	miniEMACS	日本語表示可能。 日本語の入力は、パソコンまたは CDE の日本語入力システムを使用してください。
vi	vi	

また、`less`、`nkf` などのコマンドも利用できます。

[7] VPP5000 の/home のファイルは、gpcs から/vpp のファイルとしてみえますので、それらのファイルは、汎用計算サーバ gpcs で編集することも可能です。

TSS では、CPU 使用時間は 60 分、メモリは標準で 500MB まで、指定をすれば 2 GB まで使用できます。現在のシステム資源は、limit コマンドで表示できます。これらの設定を変更する場合にも limit コマンドで行います。limit コマンドの使用例を以下に示します。

（例）limit コマンドによりメモリを 2 GB まで使用可能にする。

```
vpp% limit
cputime         1:00:00
filesize        2097152 kbytes
datasize        2097152 kbytes
stacksize       16384 kbytes
coredumpsize    16384 kbytes
memoryuse       524288 kbytes
descriptors     512
vpp% limit memoryuse 2097152kb
vpp% limit
cputime         1:00:00
filesize        2097152 kbytes
datasize        2097152 kbytes
stacksize       16384 kbytes
coredumpsize    16384 kbytes
memoryuse       2097152kb kbytes
descriptors     512
```

7.2 翻訳処理

VPP5000 では TSS で翻訳・実行ができます。翻訳コマンドについては“ 4．翻訳コマンド ”を参照ください。

7.2.1 Fortran の翻訳処理と逐次実行

Fortran のソースのファイル名：prog.f

実行可能ファイル名：prog

```
vpp% frt  -o prog  prog.f
vpp% ./prog
```

7.2.2 VPP Fortran による並列処理の翻訳

VPP Fortran で並列処理を行うためには、コンパイラオプション-Wx を指定します。

Fortran のソースのファイル名：prog_p.f

実行可能ファイル名：prog_p

```
vpp% frt  -Wx  -o prog_p  prog_p.f
```

7.3 TSS での並列実行

VPP5000 の TSS では、並列実行ができます。TSS の並列処理で使用できる PE は 4 台ですが、折り返し機能により 8 並列まで実行可能です[8]。デバッグやインタラクティブに結果を表示したい場合には便利です。課金の対象となる CPU は、バッチ型処理と同じで使

用した CPU の演算時間のうち最大のものです。演算負担経費は、会話型処理と同じで、1秒につき 2 円です。TSS の並列処理での利用可能なシステム資源については“ 3．利用形態とジョブ種別 ”の項を参照してください。

TSS での並列実行は、jobexec コマンドで行います。jobexec コマンドの主なオプションを以下に挙げておきます。オプションの詳細は、man コマンドで確認してください。

-vp　使用する PE 数を指定する。

-ct　CPU 使用時間を指定する。省略値は、300 分。
　　値は、[[時：] 分：] 秒の形式で指定する。

-et　経過時間を指定する。省略値は、720 分。
　　値は、[[時：] 分：] 秒の形式で指定する。

-mem　使用するメモリサイズを指定する。省略値は 2048MB。
　　値はメガバイト単位で指定する。

なお、jobexec コマンドのオペランドで指定するスクリプトファイルには、実行権が必要です。また、このスクリプトファイルにはホームからのパス名を指定してください。

以下に使用例を示します。

【使用例】4 台の PE を使用し、メモリサイズを 3GB に、CPU 使用時間を 30 分に、経過時間を 60 分に指定して、スクリプトファイル（ファイル名：$HOME/ex/para_tss）に指定されたジョブを実行する。

```
vpp% jobexec  -vp 4  -mem 3072  -ct 30:  -et 60:  ex/para_tss
```

８．実行時間の計測方法

実行に要した CPU 時間は、 timex コマンドにより知ることができます。VU（ベクトルユニット）の使用時間も表示されます。

```
cd  pg
timex   ./vtest_s
```

timex コマンドの出力例を次に示します。ここで、real は経過時間を示しています。実行に要した CPU 時間は、user と sys の項目を合わせたものです。時間はいずれも秒単位です。並列処理の場合には、user と sys の項目は、すべての PE の合計値となります。並列実行の場合 -P オプション[9]を指定すると各 PE での CPU 時間が表示されます。この timex コマンドの出力は標準エラー出力に出力されます。

```
real          6:31.22
user          6:23.52
sys              2.11
vu-user       6:23.24
vu-sys           0.00
```

---

[8] 現在の運用では、1 台の PE に 2 個の仮想プロセッサを割り当てることも可能になっているため。

[9] Gaussian98 で並列処理を行う場合には-P オプションを指定すると出力が多くなるので、-P オプションの指定は行わない方がよい。

９．OPEN 文によらないファイルの結合方法＜Fortran＞

Fortran のプログラムで装置番号とファイルを結合する場合、OPEN 文ではなく環境変数によりファイルの対応づけを行なうことができます。環境変数によるファイル名の指定は、次のように行います。

setenv　fuxx　ファイル名

ここで、fu：固定、xx：装置番号(00～2147483647)

装置番号とファイルの結合例を次に示します。

```
setenv  fu01  testin.dat
setenv  fu02  testout.dat
./a.out
```

この例では、装置番号 1 に対して testin.dat を、装置番号 2 に対して testout.dat を割り当てています。なお、環境変数によるファイルの結合と OPEN 文の両方が指定されている場合には、OPEN 文が優先されます。

１０．図形出力ライブラリの利用方法

図形出力ライブラリを用いて PS ファイルを作成する場合には、 frt コマンドの-l オプションで ps を指定してください。使用例を次に示します。

```
cd pg
frt  -o xyout  xyout.f  -l ps
```

なお､PS 用図形出力ライブラリでは装置番号 99 に対して PS ファイルを出力しているので、プログラムの実行に際しては PS ファイルの出力先の指定をしてください。指定方法には、以下の３つがあります。

・OPEN 文

・環境変数による 99 へのファイル割り当て

・XINT サブルーチンの引数の file=で指定

なお、上記のいずれの指定もしなかった場合には、fort.99 という名前のファイルが作成されます。

１１．スーパーコンピュータのファイル利用について

11.1　高速大容量ファイルの利用について

スーパーコンピュータ VPP5000 には、高速処理できる大容量向きのファイルシステム dpfs(distributed parallel file system)が用意されています。これは大容量のファイルや並列ジョブからの入出力に適したファイルシステムです。

dpfs を利用する場合には、利用に先立ちディレクトリを作成する必要があります。ディレクトリの作成は、dpfsdir コマンドで行います。dpfsdir コマンドを実行すると /home/dpfs/usr*N*/*login-name* のディレクトリが作成されますので、これらのディレクトリの下にファイルを作成してください。dpfs のファイルの課金や保存期間は、通常のファ

イルと同様です。

また、dpfs では実行時に入出力作業領域を指定すると読み書きの処理が更に高速になります。入出力作業領域の指定方法を以下に示します。

【例】実行時に 10MB の入出力作業領域を指定する。

```
./a.out  -Wl,-g10000
```

11.2 ファイル入出力のエラー

ファイルの入出力関係でエラーが出た場合、エラー番号は表示されますが、エラーの内容までは表示されません。エラー番号の内容は、ferrno コマンドで表示できます。

＜エラーメッセージ＞

```
jwe0022i-s line 10 入出力エラーが発生しました(system call=open, errno=2, unit=
1) .
  error occurs at MAIN__     line 9 loc 0000000000000830 offset
00000000000002b0 (2c,2c)
  MAIN__       at loc 0000000000000580 called from o.s.
 jwe0903i-u エラー識別番号 0022 のエラーが発生し, エラー打切り回数に達しました.
 error summary (Fortran)
 error number  error level  error count
   jwe0022i        s          1
 total error count = 1
```

【ferrno コマンドの使用例】

```
vpp% ferrno 2
      #define ENOENT 2 /* No such file or directory  */
```

これがエラーの内容

11.3 dpfs_alloc コマンド

実行に先立ちあらかじめ dpfs のファイルの領域を確保しておきたい場合に利用します。

形式：dpfs_alloc -i 初期割当て量 -e 拡張割当て量 ファイル名

ここで、割当量は、いずれも 2 MB 単位のデータブロック数で指定します。1 以上 8388608 の値が指定できます。

11.4 mrfs(memory resident file system)の利用

VPP5000 では、mrfs と呼ばれるメモリ常駐ファイルシステムが他のファイルシステムと同じように利用できます。mrfs は、一時的にメモリ上にファイルを作成しますので、その入出力性能は数百 MB/sec と高速です。しかし、ジョブの終了時点で自動的に開放され削除されます。mrfs のファイルは、NQS の qsub コマンドの-cc または-lV オペランドの指定により利用できます。

1. qsub コマンドの-cc オペランドを指定することによりバッチリクエストの実行ディレクトリを mrfs 上とする。

【例】

```
vpp% qsub -cc go.sh
```

2. qsub コマンドの-lV オペランドで、使用する mrfs のファイルサイズを指定する。このファイルサイズは、-lM オペランドで指定されたメモリ量から確保される。-cc オペランドを指定しない場合は、環境変数 MRFSDIR によりファイルの入出力をおこなう。

【例】

```
#  @$-q x  -eo  -o prog1.out -lV 1gb -lM 3gb
setenv  fu01  prog1/testin.dat
cd $MRFSDIR
setenv fu10  temp.dat
$HOME/prog1/a.out
```

１２．MPI の利用について

12.1 翻訳

Fortran　　mpifrt

C　　mpicc

【例】 vpp% mpifrt -o mpi_prog mpi_prog.f

12.2 実行

vpp で MPI を実行する場合には、full モードと limited モードの２つの機能モードがある。

12.2.1 full モードでの実行

mpiexec コマンドで起動し、MPI2 規格のすべての言語仕様のプログラムを実行することが可能であるモードである。full モードでの実行に必要なプロセッサ数は、mpiexec の使用するプロセッサ分（１）増加したものになる。

12.2.1.1 full モードの実行方法

mpiexec コマンドにより行う。mpiexec コマンドの形式を次に示す。

mpiexec -n *np* *prog* *args*

ここで、

np ：ユーザの実行可能プログラムが最初に静的に必要とするプロセッサ数

prog ：最初に静的に生成される実行可能プログラムのパス名

args ：*prog* への引数

バッチジョブとして動作させるには、qsub コマンドにより実行依頼する。次に示すようなスクリプトファイルを作成し、qsub コマンドで実行する。

スクリプトファイル名：mpi_full.sh

```
 #  @$-q z  -eo -o mpi_full.out
 #  @$-lP 5
mpiexec -n 4 mpi_prog
```

キューの指定

PE 数の指定

プロセッサ数の指定

```
vpp% qsub  mpi_full.sh
```

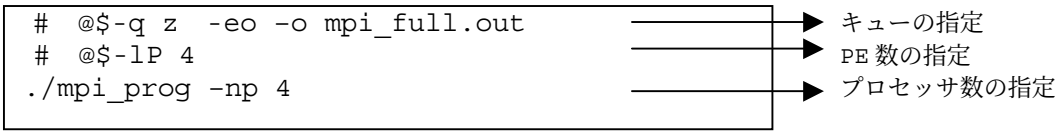
12.2.2 limited モードでの実行

limited モードでは、MPI2 規格の言語仕様のうち、プロセス管理および MPI I/O 機能が実行できない。

12.2.2.1 limited モードの実行方法

バッチジョブとして動作させるには、qsub コマンドにより実行依頼する。次に示すようなスクリプトファイルを作成し、qsub コマンドで実行する。

スクリプトファイル名：mpi_limited.sh




```
vpp% qsub  mpi_limited.sh
```

12.2.3 TSS での実行

vpp では、TSS で並列実行ができる。TSS の並列処理で使用できる PE は４台である。デバッグやインタラクティブ実行に便利である。TSS の並列実行は jobexec コマンドで行う。PE 数の指定は-vp オプションで、プロセッサ数の指定は-np オプションで行う。

```
vpp%  jobexec -vp 4
jobexec start
cd  MPI_pro
./mpi_prog  -np 4
```

12.2.4 実行時の環境変数

環境変数一覧　　『UXP/V　MPI 使用手引書』　表 2-3（p.27）

通常は、環境変数を意識する必要はないが、転送するメッセージが多くなると環境変数 VPP_MBX_SIZE で､メッセージプールの大きさ(バイト単位)を指定する必要がある｡なお、省略値は、4,194,304 バイトである。

【例】　setenv VPP_MBX_SIZE 10485760

上記の環境変数でメッセージプールを大きく指定することにより回避できたエラー

（その１）

```
MPLIB-sr2.3.1 Exception Handler
DeadLock detected during parallel execution,
By process 119369939, running as MASTER on processing element 0
Time Stamp : Fri Dec 15 20:11:30 2000
Deadlock Diagnostic : Process is part of cycle.
          :
```

（その２）

```
MPLIB-sr2.3.1 Signal Handler
Signal 6 (code 6 - errno 16) received during parallel execution,
By process 1180762371, running as NODE on processing element 2
Executable : ./mpi_prog
Time Stamp : Wed Dec 13 15:11:30 2000
Diagnostic : Abort
    Receive queue/mailbox overflow
    Interrupt vpid is 2
           :
```

１３．必要メモリ量の試算

VPP5000 上で実行するプログラムの必要メモリ量は、次の手順で算出[10]することにより知ることができます。

(1) size コマンドにより data と bss のサイズを知る。

【例】

```
vpp% size vtest_p32
2547536 + 355248 + 819120208  = 822022992
```

text　　data　　bss　　total

(2) gsize コマンドにより PE 数と max_global サイズを知る。

【例】

```
vpp% gsize vtest_p32
File vtest_p32 global array information is following.
-------------------------------------------------------------
  global array information :
    total size        -            62812624KB(   4) on    32 pe's
      partitioned     -            62812624KB(   4)
      non-partitioned -                   0KB(   0)
    max. size / pe    -             1962894KB
```

これが max_global サイズ

(3) LOCAL のメモリ量を次式より求める。

LOCAL = [data + bss + 1.5MB + 22KB*PE 数　+ i/o buffer 域] + 128MB

ここで、[ ]内は 128MB に切り上げる。なお、i/o buffer 域の default 値は、ファイルシステムにより異なり、次のようになっている。

fpfs(/home のファイル)　　――――　　512KB

dpfs(/home/dpfs のファイル)　　――――　　2MB

[10] ここでは、計算を簡単にするために１K=1000、１M=10^6で計算していますが、正確には１K=1024、１M=1024^2で計算してください。

【例】

LOCAL = [355KB + 819MB + 1.5MB + 22KB*32 + 512KB] + 128MB

= [822MB] + 128MB

= 896MB + 128MB = 1024MB

(4) GLOBAL のメモリ量を次式より求める。

GLOBAL = [max_global + 256KB + 2KB*PE 数]

【例】

GLOBAL = [1962MB + 256KB + 64KB]

= 2048MB

(5) 必要メモリ量 = LOCAL + GLOBAL

【例】

必要メモリ量 = 1024MB + 2048MB = 3072MB

１４．クロスコンパイル

汎用計算サーバ gpcs で VPP 用のクロスコンパイルができます。クロスコンパイラのコマンド名と機能を表 5 に示します。

表 5：クロスコンパイラのコマンド名と機能

コマンド名	機　　能
frtpx	Fortran/VP、Fortran/VPP、HPF のコンパイル
fccpx	C（VPP 上の C）のコンパイル
CCpx	C++（VPP 上の C++）のコンパイル
mpifrtpx	MPI(Fortran 用)のコンパイル
mpiccpx	MPI(C 用)のコンパイル

これらのクロスコンパイラを利用するためには、以下の環境設定が必要です。

１）環境設定ファイルの.cshrc の変更

通常の環境設定では、クロスコンパイラが動作する環境になっていません。そこで、ホームディレクトリの下の.cshrc のファイルの内容を以下のように変更してください。

source /etc/skel/local.cshrc ⇒ source /etc/skel/local.cshrc.all

２）VPP システム利用者情報の設定

gpcs の vppnetpx コマンドで利用者情報の設定を行います。以下に使用例を示します。

【使用例】

```
gpcs% vppnetpx
 Enter VPP Host name    : vvp
 Enter VPP Login name   : a49999a
 Enter VPP Password     :
 Enter VPP output directory : /home/usr9/a49999a/pg
```

オブジェクトと実行可能ファイルを格納する VPP 上のディレクトリを指定する。

利用者情報に変更がある場合には、再度 vppnetpx コマンドで変更してください。

3）frtpx コマンドでクロスコンパイルする場合には、環境変数 FORT90C の設定を削除[11]してから行ってください。なお、VPP との Fortran の環境を合わせるためコンパイラオプション -X9（Fortran95 の言語仕様を採用）の指定をしてください。

（使用例）

```
gpcs%  unsetenv  FORT90C
gpcs%  frtpx  -X9 -o prog1  prog.f
```

クロスコンパイラの各コマンドのオプションは、man コマンドで確認してください。

１５．大容量ファイルの退避

計算結果のファイルが大量にある場合、その退避装置として、DVD の貸出しボリュームが利用できます。一旦ファイルを DVD へ転送する手間が要りますが、ディスクファイルに比較して 1 桁程度低額になっています。DVD の利用申請は、センター4 階事務受付（052-789-4355）で行います。利用の概要は次のとおりです。

1. 利用負担金
   - 1 ボリューム 5GB とし、1 ボリュームにつき月額 200 円とする。
   - 利用申請は、5 ボリューム（25GB）以上とする。
   - DVD の利用負担金額は、DVD 利用申請時および月初めに請求を行う。
2. DVD 貸出しボリュームシステムの利用方法
   - ホスト名

     NICE（FastEther）経由：dvdserv.cc.nagoya-u.ac.jp

     システムネットワーク（GigaEther）経由：dvdserv-g）
   - ログイン名、パスワード：他のシステム（vpp、gpcs、nucc 等）と同じ。
   - DVD 利用申請ごとに DVD ボリュームが作成される。ボリューム名は、登録番号＋1 桁。

     （例）登録番号：a49999a　———　ボリューム名：a49999a1
   - DVD ボリュームのマウントポイントは、$HOME/ボリューム名。

     （例）登録番号（login 名）：a49999a　ボリューム名：a49999a1

     マウントポイント：/home/usr9/a49999a/a49999a1
   - DVD 貸出しボリュームシステムでは、2GB を超えるファイルは扱えないので、ファイルを圧縮したり、split コマンドでファイルを分割すること。
   - ホームディレクトリには、環境設定ファイル以外のファイルは作成しないこと。

3. 利用例

【例１】chkdvd.dat のファイルを DVD ボリューム a49999a1 へ転送する。ホスト名は、dvdserv-g を指定する。

```
vpp% ftp dvdserv-g
```

[11] 環境変数 FORT90C の設定を戻すときは、再 login してください。

```
Connected to 192.168.1.19.
220 dvdserv FTP server (Version wu-2.6.0(1) Fri Apr 21 12:59:33
JST 2000) ready.
Name (dvdserv-g:a49999a):
331 Password required for a49999a.
Password:
230 User a49999a logged in.
ftp> cd a49999a1
250 CWD command successful.
257 "/home/usr9/a49999a/a49999a1" is current directory.
ftp> bin
200 Type set to I.
ftp> put dvdchk.dat
200 PORT command successful.
150 Opening BINARY mode data connection for dvdchk.dat.
226 Transfer complete.
local: dvdchk.dat remote: dvdchk.dat
367048480 bytes sent in 82 seconds (4369.69 Kbytes/s)
```

dvdserv のパスワードを入力する。gpcs と同じパスワード。

DVD ボリュームのマウントポイントに移る。

DVD ボリュームにファイルを転送する。

【例 2】2GB を超えるファイル(chk2g.dat)を DVD ボリュームへ転送する。

（その１）2GB を超えるファイルは、DVD ボリュームには転送できないので、split コマンドでファイルを分割する。ファイルの大きさで分割する場合には、-b オプションでその大きさを指定する。この例では、2000MB 単位に分割し、分割されるファイルの接頭辞を chk2g.datX と指定している。なお、この接頭辞のオペランドを省略すると、x となる。split コマンドのオペランドの詳細については、man コマンドで確認すること。

```
vpp% split  -b 2000m  chk2g.dat  chk2g.datX
vpp% ls
chk2g.dat  chk2g.datXaa  chk2g.datXab  chk2g.datXac
```

（その２）分割されたファイルを DVD ボリュームに転送する。

```
vpp% ftp dvdserv
Connected to 133.6.90.19.
220 dvdserv FTP server (Version wu-2.6.0(1) Fri Apr 21 12:59:33
JST 2000) ready.
Name (dvdserv:a49999a):
331 Password required for a49999a.
Password:
230 User a49999a logged in.
ftp> cd a49999a1
```

dvdserv のパスワードを入力する。vpp と同じパスワード。

DVD ボリュームのマウントポイントに移る。

```
250 CWD command successful.
257 "/home/usr9/a49999a/a49999a1" is current directory.
ftp> bin
200 Type set to I.
ftp> put chk2g.datXaa
200 PORT command successful.
150 Opening BINARY mode data connection for chk2g.datXaa.
226 Transfer complete.
local: chk2g.datXaa remote: chk2g.dataXaa
2097152000 bytes sent in 7.6e+02 seconds (2.7e+03 Kbytes/s)
       :
       :
```

DVD ボリュームにファイルを転送する。

（補足）

分割されたファイルを１つのファイルに連結するのは、cat コマンドで行う。

```
vpp% cat chk2g.datXaa chk2g.datXab chk2g.datXac >chk2g.dat
```

【参考資料】

＜Fortran 関係＞

(1) UXP/V Fortran 使用手引書 V20 用(J2U5-0410)

(2) UXP/V Fortran/VPP 使用手引書 V20 用(J2U5-0430)

(3) UXP/V Fortran メッセージ説明書 V20 用(J2U5-0460)

(4) UXP/V HPF 使用手引書 V20 用(J2U5-0450)

＜C 関係＞

(5) UXP/V C 言語使用手引書 V20 用(J2U5-0121)

(6) UXP/V DPCE 使用手引書 V20 用

＜C++＞

(7) UXP/V C++使用手引書 V20 用 (J2U5-0311)

＜ライブラリ＞

(8) 富士通 SSLⅡ使用手引書(99SP-4020)

(9) FUJITSU SSLⅡ拡張機能使用手引書(99SP-4070)

(10) FUJITSU SSLⅡ拡張機能使用手引書Ⅱ(J2X0-1360)

(11) FUJITSU SSLⅡ/VPP 使用手引書(J2X0-1372)

(12) FUJITSU C-SSLⅡ使用手引書(J2X0-3330)

(13) UXP/V BLAS/VP LAPACK/VP ScaLAPACK 使用手引書 V20 用(J2U5-0480)

＜Fortran 並列処理技法＞

(14) UXP/V VPP Fortran プログラミングハンドブック V20 用

(15) VPP Fortran 入門（改定版）名古屋大学大型計算機センターニュース
Vol.31,No.3,2000.8

(16) VPP Fortran から HPF へ,名古屋大学大型計算機センターニュース
Vol.31,No.4,2000.11

＜ベクトル化技法＞

(17) UXP/V Fortran プログラミングハンドブック V20 用

＜プログラミング支援＞

(18) UXP/V アナライザ使用手引書 V20 用

＜メッセージパッシングライブラリ＞

(19) UXP/V MPI 使用手引書 V20 用(J2U5-0272)

(20) UXP/V PVM 使用手引書 V20 用(J2U5-0141)

(21) FUJITSU MPTools 使用手引書(J2X0-2641)

なお、上記参考資料のうち(15)、(16)以外は、名古屋大学大型計算機センターの利用者に限りホームページで参照することができます。

【VPP のデータ形式】

VPP の浮動小数点の形式は、IEEE 形式（ANSI/IEEE Std 754-1985 により規格化されているもの）と呼ばれるもので、多くのワークステーションで採用されているものと同じ形式である。単精度(REAL*4)の形式と倍精度(REAL*8)の形式を図２に示す。ここで、ｓは符号部、e は指数部、f は仮数部とする。

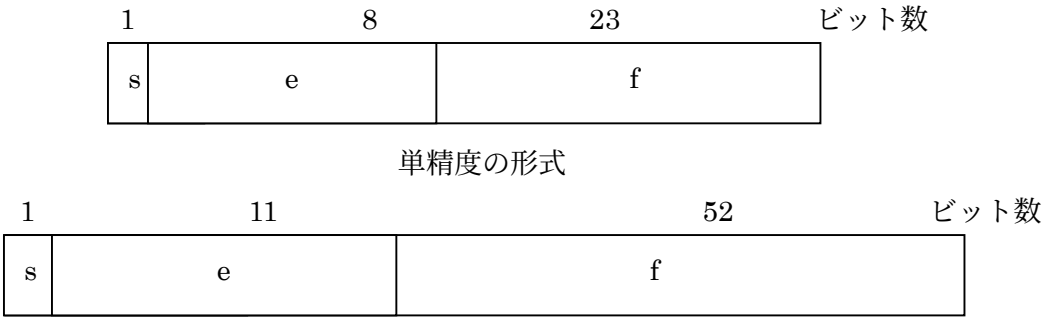


図２：VPP の浮動小数点の形式

この形式で表される単精度の浮動小数点の値を表６に、倍精度の浮動小数点の値を表７に示す。表中で INF は *Infinity* を NaN は *Not a Number* を表す。

表５：単精度の浮動小数点の値

指数部と仮数部の値	表現される浮動小数点の値
0 < e < 255	$(-1)^s 2^{e-127}(1.f)$
e = 0, f≠0	$(-1)^s 2^{e-126}(0.f)$
e = 0, f = 0	$(-1)^s(0.0)$
e = 255 , f = 0	$(-1)^s(INF)$
e =255, f≠0	*NaN*

表６：倍精度の浮動小数点の値

指数部と仮数部の値	表現される浮動小数点の値
0 < e < 2047	$(-1)^s 2^{e-1023}(1.f)$
e = 0, f≠0	$(-1)^s 2^{e-1022}(0.f)$
e = 0, f = 0	$(-1)^s(0.0)$
e = 2047 , f = 0	$(-1)^s(INF)$
e =2047, f≠0	*NaN*