Chapter 3

Magneto-Hydrodynamical Simulation Software CANS

Originally written by H. Hanayama, translated by S. Miyaji [1]

3.1 What is CANS?

3.1.1 Astrophysical Magneto-Hydrodynamical Simulation Software

Based on the development of computers and numerical computation techniques, numerical computation which simulates the time evolution of the developing system under physical laws with computer is established as the third method to pursue the science except experiment and theory. Since Astrophysical phenomena are impossible to mimic on the earth, the method of computational science is very efficient in Astrophysics, with which we create and simulate ideal universe in the computer, and examine adequacy of the model by comparing its numerical results with observations, in order to explain physical mechanism of the phenomena.

Hence, we construct an Astro/ Space Simulator as the product of ACT-JST project "Development of Space Simulation Net Laboratory System". By this simulator, we construct and maintain code library and provide a virtual laboratory by which one can simulate, analyze, and visualize the phenomena through network. As the Astrophysical component of this Net Laboratory, we create an Astrophysical Magneto-Hydrodynamic Simulation Software library package CANS (Coordinated Astronomical Numerical Software).

3.1.2 What we can do with CANS?

By using CANS, we can perform various Astrophysical Magneto-Hydrodynamical simulations and visualize their results. Among many dynamical and various phenomena in the Universe, there are some phenomena in which macroscopic motion of plasma strongly coupled with magnetic field is the origin of the energy release and mass transport, as like flare and jet. We can simulate with CANS these phenomena which could be described with macroscopic Hydrodynamical/ Magneto-Hydrodynamical equations.

[1] This chapter is based on NetLaboratory home page URL: www.astro.phys.s.chiba-u.ac.jp/netlab/astro/ and Dr. Tanuma's (Kyoto Univ.) "How to use CANS" Web page (in Japanese) URL: www.kwasan.kyoto-u.ac.jp/~tanuma/cans_howtouse.html

In CANS, we can include many physical processes such as self-gravity, anisotropic diffusion coefficients by magnetic field orientation, magnetic diffusion, radiation cooling, and so on, and we can select from 1 dimensional to 3 dimensional codes for various Astrophysical Magneto-Hydrodynamical simulation.

3.1.3 Merit of CANS

Among some Magneto-Hydrodynamical simulation packages, the most merit of CANS is that we provide not only the simulation code itself but also a complete set of simulation model (initial condition, boundary condition, etc.), recommended parameter set, explanation of physics involved, movie of the results, and visualization and analysis tools for typical problems (basic subjects) of Astrophysical simulation. Basic subjects cover wide variety of Astrophysical phenomena such as many Hydrodynamical and Magneto-Hydrodynamical instabilities, shock wave propagation, jet, flare, and star formation.

Normally, when the simulation code becomes to be opened to the public, it is hard to thoroughly use the code by a simple-minded user. Therefore we prepare a various source codes with their initial models, their explanations, and samples of initial parameters so that one can easily reconstruct the simulation model based on these supplies and can start new simulation. We adopted netCFD format as a standard input and output format of the data so that one can use many visualization software which offers netCFD format.

3.1.4 Structure of CANS

CANS consists of two parts; common part for Hydrodynamical/ Magneto-Hydrodynamical simulation code and special part for basic subjects with modules and explanations of each subject. As the simulation engine to solve Magneto-Hydrodynamical equation, there are 3 engines such as Modified Lax-Wendroff scheme, Roe scheme, and CIP-MOCCT scheme. One can execute the simulation with other scheme by exchanging the engine without changing other parts. Not only the modules for scalar machine but also parallelized module package with MPI for parallel machine are included in 1, 2, and 3 dimension packages.

3.2 Let's Use CANS!

At first. we briefly explain how to use CANS from installation, execution, and visualization of the result. Please refer following Web pages from developers; Dr. R. Matsumoto (Chiba Univ.) and Dr. T. Yokoyama (Univ. of Tokyo).

CANS Web page (over all description and some English documents)

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

CANS download page

`http://www-space.eps.s.u-tokyo.ac.jp/~yokoyama/etc/cans/`

CANS document (in Japanese)

`http://www.kwasan.kyoto-u.ac.jp/~tanuma/JST.html`

3.2.1 Working Environment

CANS code is written in Fortran. Therefore its working environment should be at least Fortran compilable. Based on compilation tests on various UNIX environment by the developers, on the following environments CANS is compilable and executable. But, CANS is written to be

executable on many UNIX environments so that even on other environments there should be only a slight or less change needed.

FreeBSD4.2
Linux 2.2
IRIX64 6.5
SunOS5.7
HP-UX 11.0
SUPER-UX 11.1 (SX5)
UXP/V (VPP)

3.2.2 Installation

Download CANS data file from CANS home page to your home directory and extract the data from compressed format.

There are "beta version" and "released version (version 3.0)" of CANS; the beta version is under the test period for developing the code and it includes the newest patches but may contain some bugs. The released version is already examined with various code checks and simulation tests and certified to use by the developers. Therefore released version is reliable but may be slower than the beta version.

Here, we explain the released version. Its full code is downloadable from following Web page.

`http://www-space.eps.s.u-tokyo.ac.jp/~yokoyama/etc/cans/src/cans-release.tgz`

Download CANS released version to your home directory and extract the files by tar command.

```
>cd
>wget http://www-space.eps.s.u-tokyo.ac.jp/~yokoyama/etc/cans/src/cans-release.tgz
>tar -zxvf cans-release.tgz
```

3.2.3 Check the Contents

After the extraction of CANS, check the contents of CANS sub-directory.

```
>cd cans
>ls
Develop.txt Models.pdf README     cans1d/ cans3d/ htdocs/
Makefile    NonLTE     Readme.pdf cans2d/ cansnc/ idl/
```

'*.pdf' files and README file is written by Dr. R. Matsumoto (Chiba Univ.), Dr. T. Yokoyama (Univ. of Tokyo) and Dr. N. Fukuda (Okayama Univ. Sci.) and describe explanations and guides of various CANS modules so that you should read them first (if you can read Japanese). In each sub-directory, there are README files which describe functions of each programs and we recommend you to read them before you use these modules/ subroutines.

In the directories of 'cans1d', 'cans2d', and 'cans3d', there is basic subjects and

common modules for 1, 2, and 3 dimensional simulation, respectively. In the 'cansnc' directory, there are the programs for input and output by netCFD format. In the 'idl' subdirectory, there are visualization programs for IDL visualization software. IDL itself is a commercial software and please refer following Web page.

```
http://www.rsinc.com/idl/
```

Let's check the contents of cans2d sub-directory.

```
>cd cans2d
>ls
bc/          cndbicg/        cndsor/       cndsormpi/    common/
commonmpi/   hdmlw/          htcl/         md_advect/    md_awdecay/
md_cloud/    md_cme/         md_cndsp/     md_cndtb/     md_corjet/
md_efr/      md_itmhdshktb/  md_itshktb/   md_jetprop/   md_kh/
md_mhd3dkh/  md_mhd3dshktb/  md_mhdcloud/  md_mhdcondtb/ md_mhdgwave/
md_mhdkh/    md_mhdshktb/    md_mhdsn/     md_mhdwave/   md_mri/
md_parker/   md_reccnd/      md_recon/     md_recon3/    md_rt/
md_sedov/    md_shkref/      md_shktb/     md_sndwave/   md_thinst/
mdp_awdecay/ mdp_cme/        mdp_cndsp/    mdp_cndtb/    mdp_corjet/
mdp_efr/     mdp_itmhdshktb/ mdp_itshktb/  mdp_jetprop/  mdp_kh/
mdp_mhd3kh/  mdp_mhd3shktb/  mdp_mhdcndtb/ mdp_mhdkh/    mdp_mhdshktb/
mdp_mhdsn/   mdp_mhdwave/    mdp_mri/      mdp_recon/    mdp_recon3/
mdp_rt/      mdp_sedov/      mdp_shkref/   mdp_shktb/    mdp_thinst/
```

Followings are the explanations of each sub-directory.

```
hdmlw/  module to solve Hydrodynamical/ MHD equations with Modified Lax-Wendroff
        + artificial viscosity.
bc/     procedure to define boundary condition.
common/ module for various common routines for computation.
cndsor/ module to solve thermal conduction with implicit method (1st order in time:
        matrix conversion is based on Red Black SOR scheme).
cndbicg/ module to solve thermal conduction with implicit method (1st order in time:
        matrix conversion is based on BICG scheme)
htcl/   module to solve radiation cooling and static heating with explicit method.
commonmpi/ module of common routines for MPI.
cndsormpi/ module to solve thermal conduction with implicit method (1st order in time:
          matrix conversion is based on Red Black SOR scheme) for MPI.
md_***/ module for 2 dimension basic subjects for scalar machine.
mdp_***/ module for 2 dimension basic subjects for parallel machine.
(For *** part, please refer the next screen.)
```

'md' and 'mdp' directories are the modules for basic subjects. For almost all 2 dimensional basic subjects, there are packages for scalar machine ('md' module for 1CPU computation) and for parallel machine ('mdp' module for plural CPUs computation). For example, 'md_mhdsn' and 'mdp_mhdsn' compute basically the same supernova remnant model except by scalar and parallel procedures, respectively.

Following is a summary of 2 dimensional basic subjects.

```
md_advect/ simple advection [advection]
md_awdecay/ large amplitude Alfv\'en wave attenuation instability [isothermal 3 component MHD]
md_cloud/ isothermal self-gravity collapse [isothermal hydrodynamics, self-gravity]
md_cme/ coronal mass ejection: Low solution [MHD, spherical coordinate]
md_cndsp/ spherical symmetric thermal conduction [thermal conduction, cylindrical/ spherical
            coordinate]
md_cndtb/ simple thermal conduction [thermal conduction]
md_corjet/ solar coronal jet [MHD, gravity, resistivity]
md_efr/ solar emerging flux of magnetic fields: Parker instability [MHD, gravity]
md_itmhdshktb/ isothermal MHD shock tube [isothermal MHD]
md_itshktb/ isothermal shock tube [isothermal Hydrodynamics]
md_jetprop/ propagation of jet [Hydrodynamics, cylindrical coordinate]
md_kh/ Kelvin-Helmholtz instability [Hydrodynamics]
md_mhd3kh/ 3 components MHD Kelvin-Helmholtz instability [3 components MHD]
md_mhd3shktb/ 3 components MHD shock tube [3 components MHD]
md_mhdcloud/ isothermal MHD self-gravity collapse [isothermal MHD, gravity]
md_mhdcndtb/ simple MHD thermal conduction [MHD, thermal conduction]
md_mhdgwave/ MHD wave transport in stratosphere [MHD]
md_mhdkh/ MHD Kelvin-Helmholtz instability [MHD]
md_shktb/ shock tube [Hydrodynamics]
md_mhdsn/ MHD supernova remnant [MHD, cylindrical/ spherical coordinate]
md_mhdwave/ linear MHD wave transport [MHD]
md_mri/ magneto-rotational (Balbus-Hawley instability [MHD, tidal Coriolis force]
md_parker/ Parker instability in galaxy [MHD, gravity]
md_mhdshktb/ MHD shock tube [MHD]
md_reccnd/ magnetic field reconnection with thermal conduction [MHD, resistivity,
            thermal conduction]
md_recon/ magnetic field reconnection [MHD, resistivity]
md_recon3/ 3 components magnetic field reconnection [3 components MHD, resistivity]
md_rt/ Rayleigh-Taylor instability [Hydrodynamics, gravity]
md_sedov/ supernova remnant: Sedov solution [Hydrodynamics, cylindorical/ spherical
            coordinate]
md_shkref/ reflection of shock wave [Hydrodynamics]
md_shktb/ shock tube [Hydrodynamics]
md_thinst/ thermal instability [radiation cooling]
```

3.2.4 Compilation

In '~/cans' directory, compile the program by "make" command and create library files such as 'libcans1d.a' etc. (If you change the machine or something wrong has happened, you should recompile the program.)

```
>cd ~/cans
>make 'FC=f77'
>ls
libcans1d.a, libcans2d.a, libcans3d.a, libcansnc.a
```

Here, 'FC=f77' means the command name of the Fortran compiler of your system. If your system has other version (f77 always mean Fortran 77 compiler) such as f90 (for Fortran 90), g77 (for GNU project Fortran 77), frt, and so on, use proper name of the Fortran compiler of your system.

If you want to execute parallel computation, please follow following procedure.

1. At ~/cans/cansnc directory, "make clean" for assurance.
2. Compile the module by "make 'FC=mpif77'".
3. Move to ~/cans/cans2d directory, and "make clean".
4. Compile again by "make 'FC=mpif77'".
5 "make mpi" at the same directory.

```
>cd ~/cans/cansnc
>make clean
>make 'FC=mpif77'
>cd ../cans2d
>make clean
>make 'FC=mpif77'
>make mpi
```

3.2.5 Execution of CANS program

As an example, we will describe in the followings for the case to compute 'MHD supernova remnant' model from the 2 dimensional basic subjects.

At first, move to '`cans2d/md_mhdsn`', let's check whether the files listed in Table 3.1 are created.

```
>cd cans2d/md_mhdsn
>ls
Makefile Makefile-nc Makefile-pgnc anime.pro bnd.f
main.f   model.f     pldt.pro      rddt.pro  rdnc.pro
```

File Name	Content/Function
Makefile	content of 'make' command
Makefile-nc	use when output to netCFD format
Makefile-pgnc	use when visualize by pgplot
main.f	main program
model.f	initialization subroutine
bnd.f	boundary condition subroutine
rddt.pro	file to read output data by IDL
rdnc.pro	file to read output data in netCFD format
pldt.pro	file to plot 2 dimensional contour map by IDL
anime.pro	file to show animation by IDL

Table 3.1: Explanation of Initial File in MHD Supernova Remnant Module

When "make" command is executed, beside 'a.out', output files of simulation result (extension: dac) such as 'ay.dac' and simulation parameter output file 'params.txt' are created[2]

[2]If you faced "Segmentation fault" error, you might face compilation quota limitation. In such a case, you may avoid the error by changing dimension parameters at line 5 of '/md_mhdsn/main.f' such as 'parameter(ix=50,jx=50)'. Of course you can change the quota limit of your system by checking "limit" command.

```
>make 'FC=f77'   <= For parallel computation, set 'FC=mpif77'.
>ls
Makefile Makefile-nc Makefile-pgnc a.out    anime.pro
ay.dac   bnd.f       bnd.o         bx.dac   bz.dac
main.f   main.o      model.f       model.o  params.txt
pldt.pro pr.dac      rddt.pro      rdnc.pro ro.dac
t.dac    vx.dac      vz.dac        x.dac    z.dac
```

3.2.6 Preparation for IDL

We expect that in your system, visualization software IDL IDL (The Interactive Data Language) has already installed. IDL is a commercial software but it is popular at least in Astrophysical community. Detailed explanation of the visualization by IDL will be given in somewhere.

In your '~/.cshrc', please add a path to IDL as follows.

```
setenv IDL_PATH +/usr/local/rsi/idl/lib:~/cans/idl/
```

Then you can use IDL programs in ~/cans/idl (dacgetparam.pro etc.). However, the path to IDL software itself (/usr/local/rsi/idl/lib part) is system dependent so that you should consult with your system administrator.

3.2.7 Read Data by IDL and Display

Starting IDL, read the result data outputted file 'dac file'.

```
>idl         <= start IDL.
IDL>.r rddt     <= read data.
```

[Sample of Read Data]

Display 2 dimensional plot of density contour.

```
IDL>.r pldt   <= plot data.
Plot columns & rows ? : 1,1   <= set plotting matrix for plot window.
Variable for color-maps ? (ro,pr,te) : ro <=select physical variables to be plotted
                                          (ro: density, pr: pressure, te: temperature).
Start step ? : 10 <= set time (output sequence number).
```

When input parameters like above, the simulation result will be displayed in the window like Figure 3.1.

If you want to display by color contour, set color table like follows.

```
IDL>device,decomposed=0
IDL>loadct,5
```

When you want to change the color table, type 'xloadct' and select new color table.

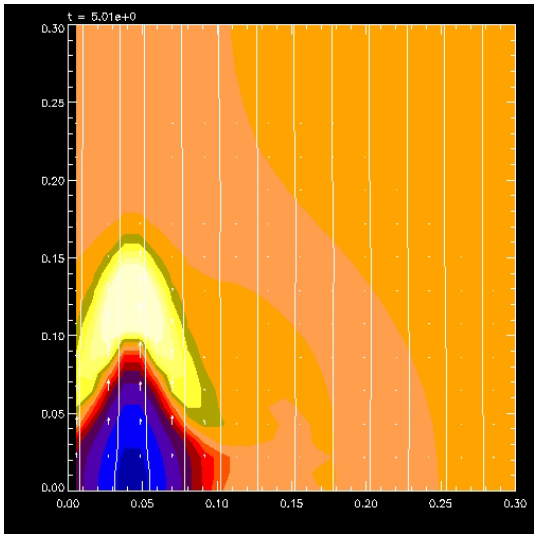

Figure 3.1: 2 Dimensional Plot of Density Contour. White lines show magnetic field line.

By a similar procedure, you can display the animation.

```
IDL>.r anime   <= animation display.
Select a varibale ? (ro,pr,te,vx,vy,bx,by,az):ro
```

For the case of other modules, please refer following Web page in which their results are disclosed as demo pages.

CANS 2D demo page

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

3.3 Computation Flow

The flow of computation procedure is based on the following main program.

`~/cans/cans2d/md_mhdsn/main.f`

Though 'main.f' program consists of many subroutines which represents model setting, computation engine, and so on, basically we can categorize them into three parts; initial setup, main computation, and end processing. In the followings, we will explain these parts according to refer an actual program.

3.3.1 Initial Set Up - main.f

In the initial setup part, you should set up initial condition to start the computation; model setup, definition of each parameters, output setup, grid size, continuation program setting, boundary condition, magnetic field setting, error check setup, and so on.

```
c=======================================================================|
c     array definitions
c=======================================================================|
      implicit double precision (a-h,o-z)
      parameter (ix=103,jx=102)
! ix is the number of mesh points in X axis.
! jx is the number of mesh points in Z axis.
      dimension x(ix),xm(ix),dx(ix),dxm(ix)
      dimension z(jx),zm(jx),dz(jx),dzm(jx)
      dimension ro(ix,jx),pr(ix,jx)
      dimension vx(ix,jx),vz(ix,jx)
      dimension bx(ix,jx),bz(ix,jx)
      dimension ay(ix,jx)
! define  parameter matrix
c=======================================================================|
c     prologue
c=======================================================================|
      mcont=0
      ndi=1000
! 'mcont' and 'ndi' are parameters for continuation. mcont=1; start continuation
! "continuation" means that re-start the computation after the 1st round computation
! as a continued computation.  Set up parameters for continuation such as terminate
! time 'tend' and/or terminate step number 'nstop'.
c-----------------------------------------------------------------------|
c   parameters
c   margin: 1 in MLW, 2 in Roe, 4 in CIP
      margin=4
! margin is the grid number which will be placed as a ample margin at the outside of
! computation area.
c-----------------------------------------------------------------------|
c  file open
! (skip netCFD part)
! we will not discuss 'netCFD' format here.
      mf_params=9
      call dacdefparam(mf_params,'params.txt')
      mf_t =10
      call dacdef0s(mf_t,'t.dac',6)
      mf_ro=20
      call dacdef2s(mf_ro,'ro.dac',6,ix,jx)
      mf_pr=21
      call dacdef2s(mf_pr,'pr.dac',6,ix,jx)
      mf_vx=22
      call dacdef2s(mf_vx,'vx.dac',6,ix,jx)
      mf_vz=24
      call dacdef2s(mf_vz,'vz.dac',6,ix,jx)
      mf_bx=25
      call dacdef2s(mf_bx,'bx.dac',6,ix,jx)
      mf_bz=27
      call dacdef2s(mf_bz,'bz.dac',6,ix,jx)
      mf_ay=28
      call dacdef2s(mf_ay,'ay.dac',6,ix,jx)
      call dacputparamc(mf_params,'comment','cans2d md_mhdsn')
      call dacputparami(mf_params,'ix',ix)
      call dacputparami(mf_params,'jx',jx)
      call dacputparami(mf_params,'margin',margin)
! output of the data is done by subroutine 'dacdefparam.f' etc. in '/cansnc' directory.
! There, creation of output file ('dac' files) and parameters output have done.
```

```
c-----------------------------------------------------------------------|
c  initialize counters
      nd=1
! 'nd' is output sequence number.
      time  = 0.0
! 'time' is time, of course.
      timep = 0.0
      ns    = 0
! 'ns' is the number of computational step.
      merr  = 0
! 'merr' is error account number.
c-----------------------------------------------------------------------|
c   time control parameters
c     nstop : number of total time steps for the run
      tend=5.0
! 'tend' is the time when the computation should be stopped.
      dtout=0.5
! 'dtout' is the time interval for writing the data.
      nstop=1000000
! 'nstop' is the step number at where the computation should be stopped.
! when 'time > tend' or 'ns > nstop', the computation should be stopped there.
c     dtout=1.d-10
c     nstop=3
c-----------------------------------------------------------------------|
c   setup numerical model (grid, initial conditions, etc.)
      call model(idf,ro,pr,vx,vz,bx,bz,gm,margin,x,ix,z,jx
     &    ,mf_params)
! subroutine model.f
! setting of grid interval, environmental gas, magnetic field,
! initial explosion velocity, etc.
! details of each subroutine will be given later.
c-----------------------------------------------------------------------|
! skip continuation (re-computation) program part
c-----------------------------------------------------------------------|
c     ready

      call grdrdy(dx,xm,dxm,x,ix)
      call grdrdy(dz,zm,dzm,z,jx)
! subroutine grdrdy.f
! set intervals between cells and grids.

      call bbtoaa_c(ay,bz,bx,dzm,dxm,x,ix,jx)
! subroutine bbtoaa_c.f
! create a vector potential in order to draw magnetic field lines.

      call bnd(margin,ro,pr,vx,vz,bx,bz,ay,ix,jx)
! subroutine bnd.f
! apply boundary condition

      floor=1.d-9
! floor is the lower limit of physical parameters; pressure,
! density, etc.

      call chkdav(n_floor,ro,vx,vz,floor,ix,jx)
      call chkdav(n_floor,pr,vx,vz,floor,ix,jx)
! subroutine chkdav.f
! check the value of physical variables whether they are below 'floor',
! if they are lower than 'floor' set these value to be 'floor'.
```

```
c-----------------------------------------------------------------------|
c     data output
! (skip 'netCFD' part.)
      mf_x=11
      call dacdef1d(mf_x,'x.dac',6,ix)
      write(mf_x) x
      mf_z=12
      call dacdef1d(mf_z,'z.dac',6,jx)
      write(mf_z) z
      call dacputparamd(mf_params,'gm',gm)
      write(mf_t) time
      write(mf_ro) ro
      write(mf_pr) pr
      write(mf_vx) vx
      write(mf_vz) vz
      write(mf_bx) bx
      write(mf_bz) bz
      write(mf_ay) ay
      write(6,913) ns,time,nd
      nd=nd+1
! output initial value.
```

3.3.2 Selection of computation scheme - main.f

Here computation time span dt is determined under CFL (Courant-Friedrich- Lewy) condition. Modified Lax-Wendroff scheme is adopted for advection calculation. After applying boundary condition, error check is performed, and outputs the result under output setup condition.

```
c=======================================================================|
c     time integration
c=======================================================================|
1000  continue
         ns = ns+1
         mwflag=0
! 'mwflag' is a flag when subroutine 'cfl_m.f' is terminated under abnormal condition.
! In such a case, flag is kept to be '0' and output the values.
c-----------------------------------------------------------------------|
c     obtain time spacing

         safety=0.4d0
! safety is a safety parameter to keep 'dt' value less than the Courant number.

         dtmin=1.d-10
! 'dtmin' is the minimum value of 'dt'.  When 'dt' is less than 'dtmin', the program
! regards that some instabilities take place so that computation will be terminated.
     call cfl_m(dt,safety,dtmin,merr,gm,ro,pr,vx,vz,bx,bz
     &      ,dx,ix,dz,jx)
! subroutine cfl_m.f
! determine 'dt' with CFL condition.
```

```
         if (merr.ne.0) goto 9999
! check error flag and if the error takes place terminate here.

         timep = time
         time = time+dt
! advance time by adding 'dt' to previous 'time'.
c-----------------------------------------------------------------------|
c     solve hydrodynamic equations
c                                                    hdmlw - start >>>
         qav=3.d0
! qav is a parameter to define the strength of artificial visicosity.
         call mlw_m_c(ro,pr,vx,vz,bx,bz,ay
     &          ,dt,qav,gm,x,xm,dx,dxm,ix,dz,dzm,jx)
! subroutine mlw_m_c.f
! main engine which solves differential equations with Modified Lax-Wendroff scheme
c                                                    hdmlw - end   <<<
! (skip Roe scheme)

         call bnd(margin,ro,pr,vx,vz,bx,bz,ay,ix,jx)
! subroutine bnd.f
! apply boundary condition

         floor=1.d-9
         call chkdav(n_floor,ro,vx,vz,floor,ix,jx)
         call chkdav(n_floor,pr,vx,vz,floor,ix,jx)
! subroutine chkdav.f
! check pressure & density are lower than lower limit.
c-----------------------------------------------------------------------|
c     data output
! here, data is outputted.
         mw=0
! 'mw' is an output parameter and normally has '0' value.
         nt1=int(timep/dtout)
         nt2=int(time/dtout)
         if (nt1.lt.nt2) mw=1
! when 'dtout' has past from previous output time, 'mw' flag is set to be '1'
! and the output data is written.

         if (mw.ne.0) then
!(skip netCFD part)
       write(mf_t) time
       write(mf_ro) ro
       write(mf_pr) pr
       write(mf_vx) vx
       write(mf_vz) vz
       write(mf_bx) bx
       write(mf_bz) bz
       write(mf_ay) ay
             write(6,913) ns,time,nd
             nd=nd+1
             mwflag=1
! when the computation is terminated without any problem, 'mwflag' is set to be '1'
! and no output at the end.
         endif

c-----------------------------------------------------------------------|
c     loop test
       if (ns .lt. nstop .and. time .lt. tend) goto 1000
! if 'ns' exceeds 'nstop' or 'time' exceeds 'tend', the computaion is stopped here.
! Otherwise, it will go back to 1000 and loop the sequence.
```

3.3.3 End Procedure - main.f

At the end, the computation would be stopped under initial termination condition. When the error took place, by checking the errors, the system would output the data of previous stage before the error took place.

```
c=====================================================================|
c     epilogue
c=====================================================================|
9999  continue
! when error took place, error routine starts from here.
c---------------------------------------------------------------------|
c  data output
      if (mwflag.eq.0) then
! if error took place in subroutine 'cfl_m.f', because of error flag
! 'mwflag=0', the result would be output here.
           write(6,913) ns,time,nd
! (skip netcdf part)
           write(mf_t) time
           write(mf_ro) ro
           write(mf_pr) pr
           write(mf_vx) vx
           write(mf_vz) vz
           write(mf_bx) bx
           write(mf_bz) bz
           write(mf_ay) ay
      endif
c---------------------------------------------------------------------|
c  file close
!(skip netCFD part)
c---------------------------------------------------------------------|
c  ending message
      write(6,915) ns,time
      if (merr.eq.0) then
        write(6,*) '  ### normal stop ###'
      else
        write(6,*) '  ### abnormal stop ###'
      endif
! when there is no error; 'normal stop', or with error; 'abnormal
! stop' will be printed out.
      stop
913   format (1x,' write    ','step=',i8,' time=',e10.3,' nd =',i3)
915   format (1x,' stop     ','step=',i8,' time=',e10.3)
      end
```

3.3.4 Model Definition - model.f

Subroutine 'model.f' is in the same directory with 'main.f'. In this subroutine, the coordinates of X and Z axes and their values will be defined.

For this model (supernova remnant), the central density 'ro(1.0)' and the pressure of interstellar matter 'prism (10^{-8})' and the plasma 'beta (10^{-5})'[3], the strength of magnetic field 'b_0(0.1853)' are defined.

Explosion energy is given as the pressure (peak value = 1) with Gauss distribution.

[3]plasma beta = gas pressure / magnetic pressure

```
c=======================================================================|
      subroutine model(idf,ro,pr,vx,vz,bx,bz,gm,margin,x,ix,z,jx
     &   ,mf_params)
c=======================================================================|
      implicit double precision (a-h,o-z)
c-----------------------------------------------------------------------|
      dimension x(ix),dxm(ix)
      dimension z(jx),dzm(jx)
      dimension ro(ix,jx),pr(ix,jx),vx(ix,jx)
      dimension vz(ix,jx),bx(ix,jx),bz(ix,jx)
! 'x(i)' and 'z(j)' are the value of grid points on X- and Z-axes.
! For example, x(5)=0.d0 and z(5)=0.d0 mean the point where
! (i,j)=(5,5) corresponds the origin (0,0) of XZ coordinates.
! dxm(i) and dzm(j) are intervals of grid points in XZ coordinates.

c-----------------------------------------------------------------------|
c   parameters
c-----------------------------------------------------------------------|
      gm=5./3.
! 'gm' is the specific heat gamma.

c-----------------------------------------------------------------------|
c   grid
c-----------------------------------------------------------------------|
      dx0=1./real(ix-margin*2+1)
! define the basis of grid interval in X-axis.
! be aware that 'i' starts from 1.

      do i=1,ix
         dxm(i)=dx0
      enddo
! up to 'ix', grid interval is kept constant 'dx0'.

      izero=margin+1
      izero=1
! in order to place the origin on the mirror boundary (on the axis), 'izero' would be set
! 'margin+1' but is defined 'izero=1' here in order to avoid numerical problem on Z-axis.
  x(izero)=0.5*dx0
! this means 'i' is between 'izero' and 'izero-1', i.e., the origin of the X-axis is on
! the cell.
      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
! define coordinate from 'x(izero)' and add 'dxm' for each grid point.

      dz0=1./real(jx-margin*2+1)
! define interval for Z-axis. Pay attention that j starts from '1'.

      do j=1,jx
         dzm(j)=dz0
      enddo
! up to 'jx' mesh, interval is kept 'dz0'.
```

```
      jzero=margin+1
      z(jzero)=0.
! for X-axis, there is no problem for computation so that the
! origin of Z-axis on the grid (it could be on the cell).
      do j=jzero+1,jx
         z(j) = z(j-1)+dzm(j-1)
      enddo

      do j=jzero-1,1,-1
         z(j) = z(j+1)-dzm(j)
      enddo
! define coordinate from 'z(jzero)' and add 'dzm' for each grid point.

c-----------------------------------------------------------------------|
c     store initial condition into common area
c-----------------------------------------------------------------------|
      prism=1.e-8
! 'prism' is the pressure of interstellar gas
      wexp=0.02
! 'wexp' is the width of Gaussian distribution where the pressure has
! 1/e of peak value.
c     prism=1/gm
      betai=1.0d5
! 'betai' is the inverse of plasma beta (=P-gas /P-magnetic field)
      pi = acos(-1.0d0)
      b0=sqrt(prism*8*pi*betai)
! set the strength of magnetic field from pressure and plasma beta

      do j=1,jx
      do i=1,ix
         ro(i,j)  = 1.
! set initial interstellar gas density to be '1.'.
         vx(i,j) = 0.0
         vz(i,j) = 0.0
! set initial velocity of the interstellar gas to be '0'.
         ss=sqrt(x(i)**2+z(j)**2)
! 'ss' is the distance from the origin in XZ coordinate.
         pr(i,j)  = prism+(1.-prism)*exp(-(ss/wexp)**2)
! explosion is given as pressure in Gauss distribution.
! center of the explosion is the origin.
c        pr(i,j) = 1/gm+0.1/gm*exp(-(ss/wexp)**2)
         bx(i,j) = 0.0
         bz(i,j) = b0
! magnetic field is assumed uniform in Z-direction.
      enddo
      enddo
c-----------------------------------------------------------------------|
c     write parameters to file
c-----------------------------------------------------------------------|
! (skip netcdf part)
      call dacputparamd(mf_params,'gm',gm)
      call dacputparamd(mf_params,'wexp',wexp)
      call dacputparamd(mf_params,'prism',prism)
      call dacputparamd(mf_params,'betai',betai)
! parameters needed have been output.

      return
      end
```

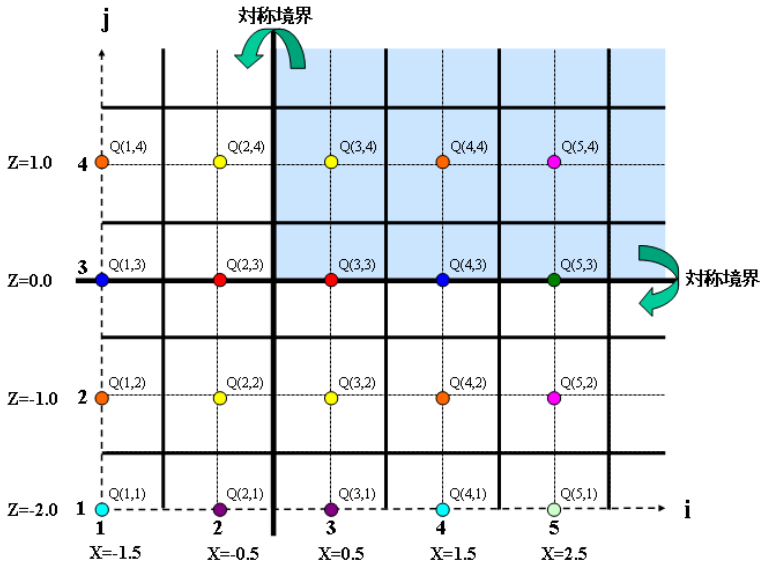

Figure 3.2: Boundary Condition near X-axis and Z-axis. It is shown with the color coding that physical quantities of each grid point is symmetric at the boundary (thick solid lines).

3.3.5 Setup of Grid Interval - grdrdy.f

Subroutine 'grdrdy.f' is in 'cans2d/common' directory. There, cell interval 'dx', grid interval 'dxm', and the middle point of grids 'xm' are defined. Details is given in README of 'cans2d/common' directory.

3.3.6 Setup for Vector Potential - bbtoaa_c.f

Subroutine 'bbtoaa_c.f' is in 'cans2d/common' directory. There, vector potential 'a_v' is calculated in order to draw magnetic field lines. Details is given in README of 'cans2d/common' directory.

3.3.7 Setup for Boundary Condition - bnd.f

Though 'bnd.f' itself is in the same directory where 'main.f' is, but subroutines of which apply boundary condition to the computating region is in 'cans2d/bc' directory and subroutines assigned in 'bnd.f' will work at the boundary. In case of the MHD supernova remnant module, because it is computed in 2 dimensional cylindrical coordinate, mirror boundary on the axes and free boundary at the outer boundary are applied. Details is given in README of 'cans2d/bc' directory. Figure 3.2 is the physical quantities at the boundary near axes for the case of 'margin=2'.

3.3.8 Check the Lower Limit - chkdav.f

Subroutine 'chkdav.f' is in 'cans2d/common' directory. When a parameter is lower than 'floor', set the parameter to be the value defined by 'floor'. Simultaneously, it is counted in 'n_events' that how many times 'floor' has used during the computation. Details is given in README of 'cans2d/common' directory.

3.3.9 Setup of CFL Condition - cfl_m.f

Subroutine 'cfl_m.f' is in 'cans2d/common' directory. According to CFL condition, it determines dt. It calculates the Alfvén velocity, the interstellar sound speed, and the gas velocity for each cell and get the maximum value among them for each cell. Using the intervals between cells, 'dx' and 'dy', it calculate 'dt' according to CFL condition with safety parameter '$safety$' in order to constrain the growth of any numerical instability. Details is given in README of 'cans2d/common' directory.

3.4 Modified Lax-Wendroff Scheme

For your convenience to understand the contents in the subroutine 'mlw_m_c.f', we will describe the solving method of Magneto-Hydrodynamic differential equations hereafter. As an actual example to differentiate basic physical equations, we adopt a numerical computation scheme, Modified Lax-Wendroff scheme here.

3.4.1 Basic Equations

As basic equations for Magneto-Hydrodynamics in 2 dimensional cylindrical coordinate, mass conservation, momentum conservation, energy conservation, and induction equation of the magnetic field are written as follows;

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0. \tag{3.1}$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) = -\nabla \mathrm{p} - \nabla \left(\frac{\mathbf{B}^2}{8\pi} \right) + \frac{1}{4\pi} (\mathbf{B} \cdot \nabla)\mathbf{B}. \tag{3.2}$$

$$\frac{\partial}{\partial t} \left(e + \frac{\mathbf{B}^2}{8\pi} \right) + \nabla \cdot \left[(e + \mathrm{p})\mathbf{v} + \frac{1}{4\pi} \{ \mathbf{B} \times (\mathbf{v} \times \mathbf{B}) \} \right] = 0. \tag{3.3}$$

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v} \times \mathbf{B}). \tag{3.4}$$

Here, $\rho, \mathbf{v}, p, t, \mathbf{B}$ mean the density, velocity, pressure, time, and magnetic field of the interstellar gas, respectively. We used that $e = \mathrm{p}/(\gamma - 1) + \rho v^2/2$ (total energy is equal to the sum of thermal and kinetic energies par volume) and specific heat $\gamma = 5/3$.

We assume isothermal evolution of the remnant so that we can neglect ϕ component of the cylindrical coordinate and only include radial (R-coordinate) and vertical (Z-coordinate) directions.

For the case of 2 dimensional cylindrical coordinate, we should pay attention on the following equations, assuming $\mathbf{A}$ and ψ are any vector and scalar.

$$\nabla \cdot \mathbf{A} = \frac{1}{r} \frac{\partial}{\partial r} (r A_r) + \frac{\partial A_z}{\partial z}. \tag{3.5}$$

$$\nabla\psi = \left(\frac{\partial\psi}{\partial r}, \frac{\partial\psi}{\partial z}\right). \tag{3.6}$$

Here, $\rho\mathbf{vv}$ is a tensor product,

$$\begin{pmatrix} \rho v_r v_r & \rho v_r v_z \\ \rho v_z v_r & \rho v_z v_z \end{pmatrix}, \tag{3.7}$$

so that

$$(\nabla\cdot\rho\mathbf{vv})_r = \frac{1}{r}\frac{\partial}{\partial r}(r\rho v_r v_r) + \frac{\partial\rho v_r v_z}{\partial z}, \tag{3.8}$$

$$(\nabla\cdot\rho\mathbf{vv})_z = \frac{1}{r}\frac{\partial}{\partial r}(r\rho v_z v_r) + \frac{\partial\rho v_z v_z}{\partial z}. \tag{3.9}$$

Before explaining the solving method of basic equations with the use of advection equation, we will briefly describe the Magneto-Hydrodynamics itself.

3.4.2 Magneto Hydrodynamics

Maxwell equations for Electro-Magnetic fields are written as;

$$\nabla\times\mathbf{E} = -\frac{1}{c}\frac{\partial\mathbf{B}}{\partial t}, \tag{3.10}$$

$$\nabla\times\mathbf{B} = \frac{4\pi}{c}\mathbf{j}. \tag{3.11}$$

Here, 'c' is the speed of light and '$\mathbf{j}$' is current density. Since, for the case of interstellar gas, hydrodynamic time scale is much longer than the vibration period of electro-magnetic waves, we neglect displacement current. And also, since the gas density is very low, we assumed that magnetic permeability is equal to 1 as like true vacuum. With (3.10), (3.11) and Ohm's law

$$\mathbf{j} = \sigma\left(\mathbf{E} + \frac{\mathbf{v}\times\mathbf{B}}{c}\right), \tag{3.12}$$

we get induction equation for magnetic field as;

$$\frac{\partial\mathbf{B}}{\partial t} = \nabla\times(\mathbf{v}\times\mathbf{B}) + \frac{c^2}{4\pi\sigma}\nabla^2\mathbf{B}. \tag{3.13}$$

Here, 'σ' is the electric conductivity.

If the fluid is the perfect conductor, the electric conductivity should be $\sigma\to\infty$, then induction equation of the magnetic field becomes eq. (3.4). We call this limit of the electric conductivity $\sigma\to\infty$ as ideal Magneto-Hydrodynamical limit.

Lorentz force by the magnetic field is

$$\begin{aligned} \frac{\mathbf{j}\times\mathbf{B}}{c} &= \frac{(\nabla\times\mathbf{B})\times\mathbf{B}}{4\pi} \\ &= -\nabla\left(\frac{B^2}{8\pi}\right) + \frac{1}{4\pi}(\mathbf{B}\cdot\nabla)\mathbf{B}, \end{aligned} \tag{3.14}$$

so that the equation of motion is

$$\rho\left\{\frac{\partial\mathbf{v}}{\partial t} + (\mathbf{v}\cdot\nabla)\mathbf{v}\right\} = -\nabla\mathrm{p} - \nabla\left(\frac{B^2}{8\pi}\right) + \frac{1}{4\pi}(\mathbf{B}\cdot\nabla)\mathbf{B}. \tag{3.15}$$

From the equation of motion eq. (3.15) and the mass conservation eq. (3.1), the momentum conservation is derived as eq. (3.2). The energy conservation eq. (3.3) is led by adding the magnetic energy to the total energy and the Poynting flux '$(c/4\pi)\mathbf{E}\times\mathbf{B}$' to the energy flux.

3.4.3 Conservation Forms of Basic Equations

In order to solve basic equations of cylindrical coordinate (3.1) - (3.4), we will rewrite basic equations (3.1) - (3.4) to advection equation in conservation form.

Assuming 'Q' is the physical variable, 'F_r' and 'F_z' are the flow to each direction, and the source term 'S'', we will rewrite advection equation in conservation form with substituting eqs. (3.5) and (3.6) as

$$\frac{\partial Q}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(rF_r) + \frac{\partial}{\partial z}F_z = S', \tag{3.16}$$

eq. 1 so that, for each component, they are rewritten as follows. Eq. (3.1) is

$$\frac{\partial \rho}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(r\rho v_r) + \frac{\partial}{\partial z}(\rho v_z) = 0. \tag{3.17}$$

eq. 2 The radial (R) component of eq. (3.2) is

$$\frac{\partial \rho v_r}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}[r\{\rho v_r^2 + \mathrm{p} + \frac{1}{8\pi}(B_z^2 - B_r^2)\}] + \frac{\partial}{\partial z}(\rho v_r v_z - \frac{1}{4\pi}B_r B_z) = \frac{1}{r}(\mathrm{p} + \frac{B_r^2 + B_z^2}{8\pi}). \tag{3.18}$$

eq. 3 The vertical(Z) component of eq. (3.2) is

$$\frac{\partial \rho v_z}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}\{r(\rho v_z v_r - \frac{B_r B_z}{4\pi})\} + \frac{\partial}{\partial z}(\rho v_z^2 + \mathrm{p} + \frac{B_r^2 - B_z^2}{8\pi}) = 0. \tag{3.19}$$

eq. 4 Eq. (3.3) is

$$\frac{\partial}{\partial t}(e + \frac{B^2}{8\pi}) + \frac{1}{r}\frac{\partial}{\partial r}\{r(e_p v_r + \frac{B_z e_y}{4\pi})\} + \frac{\partial}{\partial z}(e_p v_z - \frac{B_r e_y}{4\pi}) = 0. \tag{3.20}$$

eq. 5 The radial (R) component of eq. (3.4) is

$$\frac{\partial B_r}{\partial t} + \frac{\partial}{\partial z}(-e_y) = 0, \tag{3.21}$$

eq. 6 and its vertical (Z) component is

$$\frac{\partial B_z}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(re_y) = 0. \tag{3.22}$$

eq. 7 Here, we use $e_p = \gamma \mathrm{p}/(\gamma - 1) + \rho v^2/2$ and $e_y = -v_z B_r + v_r B_z$.

3.4.4 Differential Equation

In the MHD supernova remnant module, 2 steps Modified Lax-Wendroff scheme is adopted as the solving scheme of advection equation. In order to apply Modified Lax-Wendroff scheme, equations (eq. (3.16) and below) are re-written to the advection equations in conservation form in Cartesian Coordinate as follows.

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial r}F_r + \frac{\partial}{\partial z}F_z = S' - \frac{F_r}{r}. \tag{3.23}$$

Here, we set $S = S' - F_r/r$ and rewrite 'r' as 'x' in eq. (3.23). Then we have

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}F_x + \frac{\partial}{\partial z}F_z = S. \tag{3.24}$$

Physical Quantity Q	Flux F_x	Flux F_z	Source Term S
ρ	ρv_x	ρv_z	$-\frac{\rho v_x}{x}$
E	$e_p v_x + \frac{B_z e_y}{4\pi}$	$e_p v_z - \frac{B_x e_y}{4\pi}$	$-\frac{e_p v_x + \frac{B_z e_y}{4\pi}}{x}$
ρv_x	$\rho {v_x}^2 + p + \frac{{B_z}^2 - {B_x}^2}{8\pi}$	$\rho v_x v_z - \frac{B_x B_z}{4\pi}$	$\frac{\rho {v_x}^2}{x} + \frac{{B_x}^2}{4\pi x}$
ρv_z	$\rho v_x v_z - \frac{B_x B_z}{4\pi}$	$\rho {v_z}^2 + p + \frac{{B_x}^2 - {B_z}^2}{8\pi}$	$-\frac{\rho v_x v_z - \frac{B_x B_z}{4\pi}}{x}$
B_x	0	$-e_y$	0
B_z	e_y	0	$-\frac{e_y}{x}$

Table 3.2: Physical Quantities Q, Fluxes F_x, F_z and Source Terms S

Applying same procedure of rewriting on eqs. (3.17) - (3.22), and rearranging to each component, we will get as listed in Table 3.2.

Where,

$$e = \rho\varepsilon + \frac{1}{2}\rho v^2, \ \rho\varepsilon = \frac{p}{\gamma - 1}, \tag{3.25}$$

$$\begin{aligned} E &= \rho\varepsilon + \frac{1}{2}\rho v^2 + \frac{B^2}{8\pi} \\ &= \frac{p}{\gamma - 1} + \frac{1}{2}\rho v^2 + \frac{B^2}{8\pi}, \end{aligned} \tag{3.26}$$

$$\begin{aligned} e_p &= e + p = \frac{p}{\gamma - 1} + \frac{1}{2}\rho v^2 + p \\ &= \frac{\gamma p}{\gamma - 1} + \frac{1}{2}\rho v^2, \end{aligned} \tag{3.27}$$

$$e_y = -v_z B_x + v_x B_z. \tag{3.28}$$

Now, we will consider how to solve this 2 dimensional advection equations by differentiating them (here, we adopt 2 order Modified Lax-Wendroff scheme).

In order to solve 2 dimensional scalar advection equation with difference method, we will obtain the value of 'Q' after the time step Δt by replacing differentiations ($\partial Q/\partial r$, $\partial F_x/\partial x$, $\partial F_z/\partial z$) of 2 dimensional scalar advection equations with differences ($\Delta Q/\Delta x$, $\Delta F_x/\Delta x$, $\Delta F_z/\Delta z$). Here, Δ is the operator to take the difference between grids.

Therefore, we regard eq. (3.24) as

$$\frac{\Delta Q}{\Delta t} + \frac{\Delta F_x}{\Delta x} + \frac{\Delta F_z}{\Delta z} = S. \tag{3.29}$$

Then, we will consider how to obtain $Q_{i,j}^{n+1}$ from $Q_{i,j}^{n}$ when subscripts n is time and i and j are grid number of X and Z axes, respectively.

Lax-Wendroff scheme is the difference method based on Taylor expansion, so that it is led as follows for the case of 1 dimensional advection equation.

$$Q_i^{n+1} = Q_i^n + \Delta t \frac{\partial F}{\partial t} + \frac{1}{2}\Delta t^2 \frac{\partial^2 F}{\partial t^2} + O(\Delta t^3). \tag{3.30}$$

Substituting following equation into second and third terms of right hand side,

$$\frac{\partial Q}{\partial t} = -\frac{\partial F}{\partial t}, \tag{3.31}$$

$$\frac{\partial^2 Q}{\partial t^2} = \frac{\partial^2 F}{\partial t^2}, \tag{3.32}$$

then we obtain

$$Q_i^{n+1} = Q_i^n - \Delta t \frac{\partial F}{\partial x} + \frac{1}{2}\Delta t^2 \frac{\partial^2 F}{\partial x^2} + O(\Delta t^3). \tag{3.33}$$

Approximating space differentials ($\partial Q/\partial x$, $\partial^2 Q/\partial x^2$) with their central differences, we have

$$Q_i^{n+1} = Q_i^n - \frac{1}{2}\Delta t \frac{F_i^n - F_{i-1}^n}{\Delta x} + \frac{1}{2}\left(\frac{\Delta t}{\Delta x}\right)^2 (F_{i+1}^n - 2F_i^n + F_{i-1}^n). \tag{3.34}$$

This is the Lax-Wendroff scheme. As you can see from above derivation,

Lax-Wendroff scheme is the solver of 2nd order both in time and space. In case of 1 dimensional scalar advection equation, Lax-Wendroff scheme is the same following 2 steps scheme. This 2 steps scheme is called 2 steps Lax-Wendroff scheme.

$$Q_{i+1/2}^{n+1/2} = \frac{Q_{i+1}^n + Q_i^n}{2} - \frac{1}{2}\frac{\Delta t}{\Delta x}(F_{i+1}^n - F_i^n). \tag{3.35}$$

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}(F_{i+1/2}^{n+1/2} - F_{i-1/2}^{n+1/2}). \tag{3.36}$$

The 2 steps Modified Lax-Wendroff scheme which is used in CANS is the scheme which is improved this 2 steps Lax-Wendroff. For simplicity, we will describe the 2 steps Modified Lax-Wendroff scheme as a solver of 1 dimensional advection equation.

In this scheme, as 0-th step, physical quantities on the cell (the boundary between grids) is derived from the physical quantities on the grid Q_i^n, and physical quantities after Δt $Q_{i+1/2}^{n+1}$ are computed with the flux on grid points F_i^n, F_{i-1}^n (cf. Table 3.2 and Figure 3.3)

0-th step: Derivation of physical quantities on the cell.

$$\tilde{Q}_{i+1/2}^{n+1} = Q_{i+1/2}^n - \Delta t \frac{F_{i+1}^n - F_i^n}{\Delta x} + \Delta t \cdot S_{i+1/2}^n. \tag{3.37}$$

Hence,

$$\tilde{Q}_{i+1/2}^{n+1} = \frac{Q_{i+1}^n + Q_i^n}{2} - \Delta t \frac{F_{i+1}^n - F_i^n}{\Delta x} + \Delta t \cdot \frac{S_{i+1}^n + S_i^n}{2}. \tag{3.38}$$

Here, physical quantities on the cell are expressed with ˜.

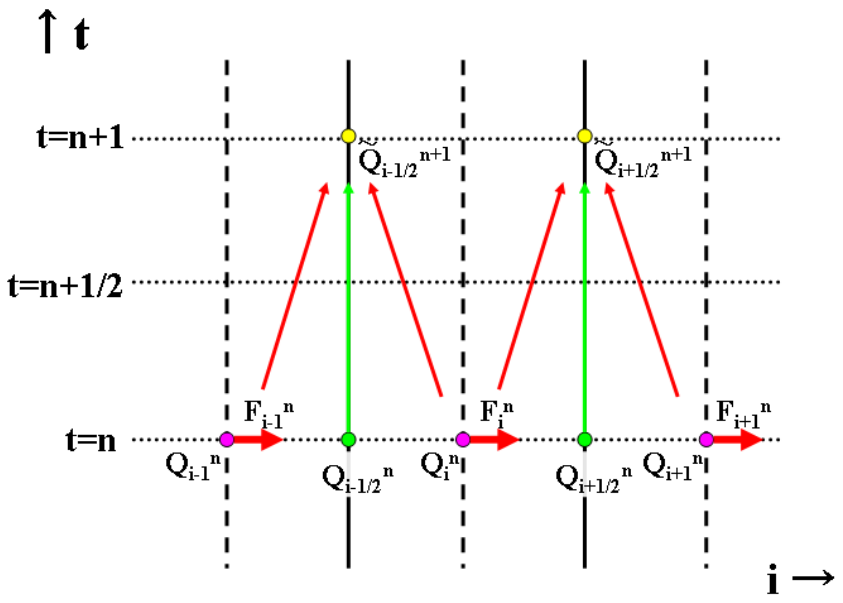

Figure 3.3: 1 Dimension 0-th Step Image

Next as 1-st step, from the central difference of both sides fluxes on grids F_{i-1}^{n}, F_{i+1}^{n} (cf. Table 3.2) are used for computing physical quantities $Q_i^{n+1/2}$ after $\Delta t/2$ has elapsed (cf. Figure 3.4).

1-st step: Central Difference of the Flux.

$$Q_i^{n+1/2} = Q_i^n - \frac{\Delta t}{2}\frac{F_{i+1}^n - F_{i-1}^n}{2\Delta x} + \frac{\Delta t}{2}\cdot S_i^n. \tag{3.39}$$

At last as 2nd step, the quantities Q_i^{n+1} after Δt are calculated by adding the difference of the fluxes on the cell $\tilde{F}_{i+1/2}^{n+1} and \tilde{F}_{i-1/2}^{n+1}$ (they have been derived from physical quantities already calculated at 0-th step $\tilde{Q}$ based on Table 3.2) on to the physical quantities $Q_i^{n+1/2}$ (cf. Figure 3.5)

2-nd step: Central Difference of Flux On the Cell.

$$Q_i^{n+1} = Q_i^{n+1/2} - \frac{\Delta t}{2}\frac{\tilde{F}_{i+1/2}^{n+1} - \tilde{F}_{i-1/2}^{n+1}}{\Delta x} + \frac{\Delta t}{2}\cdot \tilde{S}_i^{n+1}. \tag{3.40}$$

Hence,

$$Q_i^{n+1} = Q_i^{n+1/2} - \frac{\Delta t}{2}\frac{\tilde{F}_{i+1/2}^{n+1} - \tilde{F}_{i-1/2}^{n+1}}{\Delta x} + \frac{\Delta t}{2}\cdot \frac{\tilde{S}_{i+1/2}^{n+1} + \tilde{S}_{i-1/2}^{n+1}}{2}. \tag{3.41}$$

In the similar way, equations expanded into 2 dimensions are described as follows (cf. Figures 3.6, 3.7, 3.8).

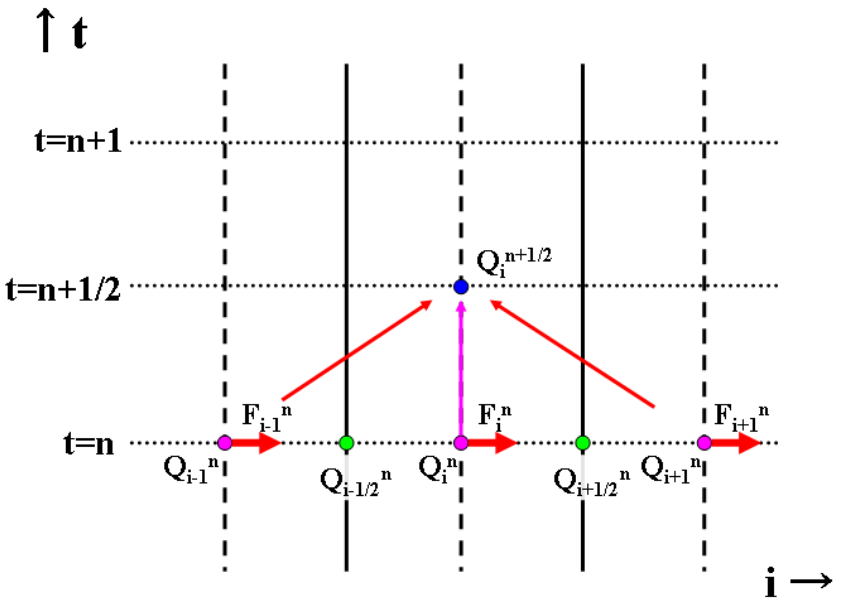

Figure 3.4: 1 Dimension 1-st Step Image

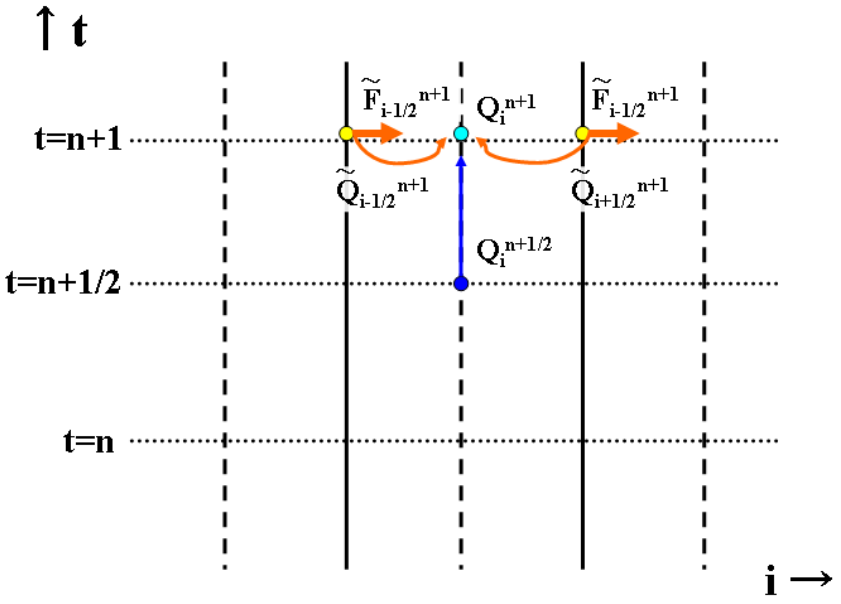

Figure 3.5: 1 Dimensional 2nd Step Image

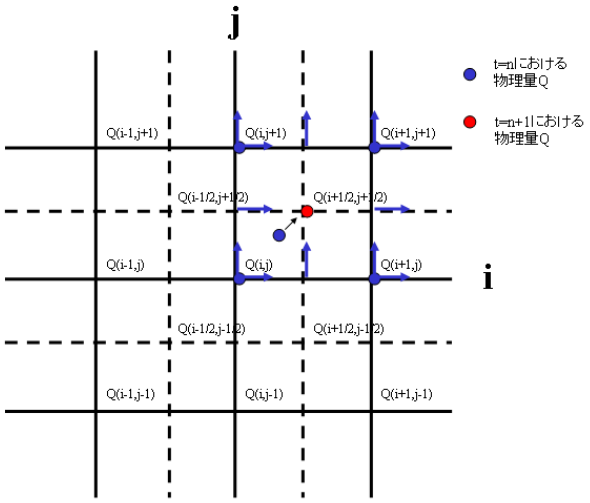

Figure 3.6: 2 Dimensional 0th Step Image

0-th step: Derivation of physical quantities on the cell.

$$\begin{aligned}\tilde{Q}^{n+1}_{i+1/2,j+1/2} &= Q^{n}_{i+1/2,j+1/2} - \Delta t \frac{F^{n}_{i+1,j+1/2} - F^{n}_{i,j+1/2}}{\Delta x} - \Delta t \frac{F^{n}_{i+1/2,j+1} - F^{n}_{i+1/2,j}}{\Delta z} \\ &\quad + \Delta t \cdot S^{n}_{i+1/2,j+1/2}.\end{aligned} \tag{3.42}$$

Hence,

$$\begin{aligned}\tilde{Q}^{n+1}_{i+1/2,j+1/2} &= \frac{1}{2}\left(\frac{Q^{n}_{i+1,j+1} + Q^{n}_{i+1,j}}{2} + \frac{Q^{n}_{i,j+1} + Q^{n}_{i,j}}{2}\right) \\ &\quad - \Delta t \frac{1}{\Delta x}\left(\frac{F^{n}_{i+1,j+1} + F^{n}_{i+1,j}}{2} - \frac{F^{n}_{i,j+1} + F^{n}_{i,j}}{2}\right) \\ &\quad - \Delta t \frac{1}{\Delta z}\left(\frac{F^{n}_{i+1,j+1} + F^{n}_{i,j+1}}{2} - \frac{F^{n}_{i+1,j} + F^{n}_{i,j}}{2}\right) \\ &\quad + \Delta t \cdot \frac{1}{2}\left(\frac{S^{n}_{i,j+1} + S^{n}_{i,j}}{2} + \frac{S^{n}_{i+1,j+1} + S^{n}_{i+1,j}}{2}\right).\end{aligned} \tag{3.43}$$

1-st step: Central Difference of the Flux.

$$\begin{aligned}Q^{n+1/2}_{i,j} &= Q^{n}_{i,j} - \frac{\Delta t}{2}\frac{F^{n}_{i+1,j} - F^{n}_{i-1,j}}{2\Delta x} - \frac{\Delta t}{2}\frac{F^{n}_{i,j+1} - F^{n}_{i,j-1}}{2\Delta z} \\ &\quad + \frac{\Delta t}{2} \cdot S^{n}_{i,j}.\end{aligned} \tag{3.44}$$

2-nd step: Central Difference of Flux On the Cell.

$$\begin{aligned}Q^{n+1}_{i,j} &= Q^{n+1/2}_{i,j} - \frac{\Delta t}{2}\frac{\tilde{F}^{n+1}_{i+1/2,j} - \tilde{F}^{n+1}_{i-1/2,j}}{\Delta x} - \frac{\Delta t}{2}\frac{\tilde{F}^{n+1}_{i,j+1/2} - \tilde{F}^{n+1}_{i,j-1/2}}{\Delta z}. \\ &\quad + \frac{\Delta t}{2} \cdot \tilde{S}^{n+1}_{i,j}.\end{aligned} \tag{3.45}$$

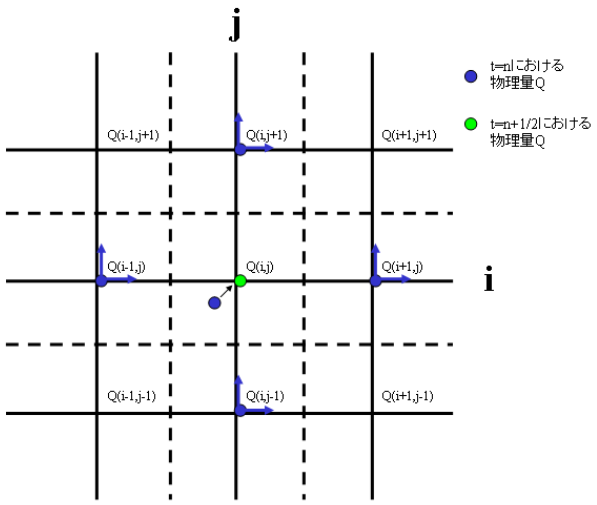

Figure 3.7: 2 Dimensional 1-st Step Image

Hence,

$$
\begin{aligned}
Q_{i,j}^{n+1} \;=\; & Q_{i,j}^{n+1/2} \\
& -\frac{\Delta t}{2}\frac{1}{\Delta x}\left(\frac{\tilde{F}_{i+1/2,j+1/2}^{n+1}+\tilde{F}_{i+1/2,j-1/2}^{n+1}}{\Delta 2}-\frac{\tilde{F}_{i-1/2,j+1/2}^{n+1}+\tilde{F}_{i-1/2,j-1/2}^{n+1}}{\Delta 2}\right) \\
& -\frac{\Delta t}{2}\frac{1}{\Delta z}\left(\frac{\tilde{F}_{i+1/2,j+1/2}^{n+1}+\tilde{F}_{i-1/2,j+1/2}^{n+1}}{\Delta 2}-\frac{\tilde{F}_{i+1/2,j-1/2}^{n+1}+\tilde{F}_{i-1/2,j-1/2}^{n+1}}{\Delta 2}\right) \\
& +\frac{\Delta t}{2}\cdot\frac{1}{2}\left(\frac{\tilde{S}_{i+1/2,j+1/2}^{n+1}+\tilde{S}_{i-1/2,j+1/2}^{n+1}}{2}+\frac{\tilde{S}_{i+1/2,j-1/2}^{n+1}+\tilde{S}_{i-1/2,j-1/2}^{n+1}}{2}\right).
\end{aligned}
\tag{3.46}
$$

3.4.5 Introduction of the Artificial Viscosity

Though the Modified Lax-Wendroff scheme is the second order scheme in both time and space, it has a defect that at the discontinuity it may create numerical instability. Details are easily found in many textbook of numerical simulation (see e.g., 'Numerical Simulation of Hydrodynamics', (1994) Univ. Tokyo Press, K. Fujii). In order to avoid this numerical instability, the term of artificial viscosity is inserted in every basic equation as a diffusion term. The diffusion term normally has diffusion coefficient κ and is given as $\kappa\nabla^2 Q$ with using physical quantity Q.

The artificial viscosity diffusion coefficients (κ_x, κ_z) used in CANS have defined using Q_v as a parameter in order to have large values at the velocity discontinuity plane,

$$
(\kappa_x, \kappa_z) = Q_v\left(\left|\frac{\partial v_x}{\partial x}\right|, \left|\frac{\partial v_z}{\partial z}\right|\right). \tag{3.47}
$$

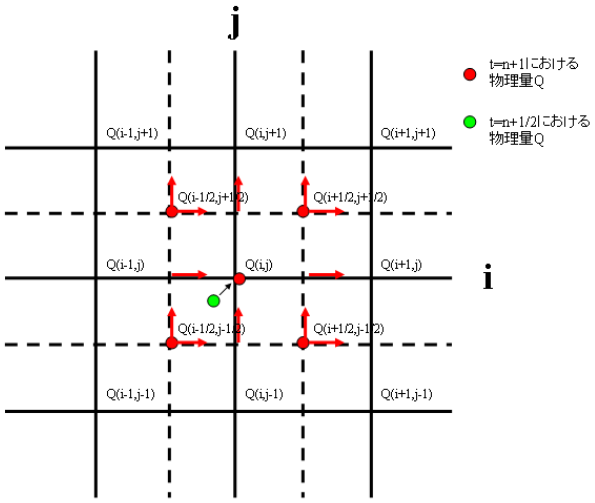

Figure 3.8: 2 Dimensional 2nd Step Image

Then, basic equations with diffusion term in CANS are written with physical quantities Q of each equations as follows.

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}F_x + \frac{\partial}{\partial z}F_z = S + \left(\frac{\partial}{\partial x}(\kappa_x \nabla Q) + \frac{\partial}{\partial z}(\kappa_z \nabla Q)\right). \tag{3.48}$$

3.4.6 Computational Engine - mlw_m_c.f

Computational engine for Modified Lax-Wendroff scheme is in '/cans2d/hdmlw' directory. It consists of subroutines of which corresponds each step of 2 steps Modified Lax-Wendroff scheme; 'mlwhalf.f', 'mlwsrch.f','mlwfull.f','mlwsrcf.f', and the subroutine to calculate artificial viscosity; 'mlwartv.f'. The 0-th and 1-st steps are computed in 'mlwhalf.f' and 'mlwsrch.f', and the 2-nd step is computed in 'mlwfull.f' and 'mlwsrcf.f'. In the previous section, we described the procedure of the Modified Lax-Wendroff scheme, we will describe the contents of the actual program 'mlw_m_c.f' here.

```
c=======================================================================|
      subroutine mlw_m_c(ro,pr,vx,vz,bx,bz,ay
     &                  ,dt,qav,gm,x,xm,dx,dxm,ix,dz,dzm,jx)
! 'ro' is density, 'pr' is pressure, 'vx' is velocity in X-direction
! 'vz' is velocity in Z-direction, 'bx' is magnetic field in X-direction,
! 'bz' is magnetic field in Z-direction, 'ay' is vector potential.
c=======================================================================|
      implicit double precision (a-h,o-z)
      dimension dx(ix),dxm(ix)
      dimension dxi(ix),dxim(ix)
      dimension ux0(ix),ux1(ix)
      dimension x(ix),xm(ix)
      dimension dz(jx),dzm(jx)
      dimension dzi(jx),dzim(jx)
      dimension uz0(jx),uz1(jx)
      dimension ro(ix,jx),pr(ix,jx),vx(ix,jx),vz(ix,jx)
     &         ,bx(ix,jx),bz(ix,jx)
      dimension ey(ix,jx)
      dimension ay(ix,jx)
      dimension ee(ix,jx),rx(ix,jx),rz(ix,jx)
      dimension roh(ix,jx),eeh(ix,jx),rxh(ix,jx),rzh(ix,jx)
     &         ,bxh(ix,jx),bzh(ix,jx)
      dimension eyh(ix,jx)
      dimension ayh(ix,jx)
      dimension prh(ix,jx),vxh(ix,jx),vzh(ix,jx)
      dimension dro(ix,jx),dee(ix,jx),drx(ix,jx),drz(ix,jx)
     &         ,dbx(ix,jx),dbz(ix,jx)
     &         ,day(ix,jx)
      dimension fx(ix,jx),qx(ix,jx)
      dimension fz(ix,jx),qz(ix,jx)
      dimension ss(ix,jx)
c-----------------------------------------------------------------------|
      pi = acos(-1.0d0)
      pi8i=1./pi/8.
      pi4i=1./pi/4.
c-----------------------------------------------------------------------|
c     ready
c-----------------------------------------------------------------------|
      do i=1,ix
         dxi(i) = 1.0/dx(i)
         dxim(i) = 1.0/dxm(i)
      enddo
! define the inverses of dx and dxm
! both dx and dxm are created in 'grdrdy.f'.

      do i=2,ix-1
         ux1(i)  = 0.5*dxm(i-1)/dx(i)
         ux0(i)  = 0.5*dxm(i)/dx(i)
      enddo
! ux1(i) and ux0(i) will be used in 'mlwfull.f' and 'mlwsrcf.f'.
      do j=1,jx
         dzi(j)  = 1.0/dz(j)
         dzim(j) = 1.0/dzm(j)
      enddo
      do j=2,jx-1
         uz1(j)  = 0.5*dzm(j-1)/dz(j)
         uz0(j)  = 0.5*dzm(j)/dz(j)
      enddo
! same as in X-direction.
```

```
c-----------------------------------------------------------------------|
c     initialize dro etc.
c-----------------------------------------------------------------------|
      do j=1,jx
      do i=1,ix
         dro(i,j)= 0.0
         dee(i,j)= 0.0
         drx(i,j)= 0.0
         drz(i,j)= 0.0
         dbx(i,j)= 0.0
         dbz(i,j)= 0.0
         day(i,j)= 0.0
      enddo
      enddo
! 'dro' etc. are the variation of the quantities (e.g., 'ro') from time 'n' to 'n+1'.
! Variables which will be calculated later should be initial zero cleared here.
c-----------------------------------------------------------------------|
c     calculate energy from pressure
c-----------------------------------------------------------------------|
      do j=1,jx
      do i=1,ix
         vv=vx(i,j)**2+vz(i,j)**2
         bb=bx(i,j)**2+bz(i,j)**2
         ee(i,j) = pr(i,j)/(gm-1)+0.5*ro(i,j)*vv+pi8i*bb
         rx(i,j) = ro(i,j)*vx(i,j)
         rz(i,j) = ro(i,j)*vz(i,j)
      enddo
      enddo
! physical quantities needed are calculated from 'ro', 'pr', 'vx', 'vz', 'bx', and 'bz'.
! 'ee' is the total energy which corresponds 'Q' in energy conservation equation.
! 'rx' and 'rz' are the momentum which correspond 'Q' in momentum conservation equation.
! Though 'rx' and 'rz' also correspond fluxes 'Fx' and 'Fz' in mass conservation
! equation, at mass conservation equation calculated below these fluxes are calculated
! from 'ro*vx' and 'ro*vz' instead of 'rx' and 'rz'.     Refer Table 1.2.
      do j=1,jx
      do i=1,ix
         ey(i,j) = -vz(i,j)*bx(i,j)+vx(i,j)*bz(i,j)
      enddo
      enddo
! ey is the quantity to be used to compute the flux in energy conservation equation and
! also used as the flux in induction equation of the magnetic field.  Refer Table 1.2.
c-----------------------------------------------------------------------|
c     step intermediate results for flux calculation
c-----------------------------------------------------------------------|
! from here, 0-th and 1-st steps computation start.
c---  density ---
      do j=1,jx
      do i=1,ix
         fx(i,j)= ro(i,j)*vx(i,j)
         fz(i,j)= ro(i,j)*vz(i,j)
         ss(i,j)= -fx(i,j)/x(i)
      enddo
      enddo
! 'fx' is the flux Fx, 'fz' is the flux Fz, 'ss' means source term.
! Fluxes Fx and Fz and source term S of mass conservation equation
! are calculated here. Refer Table 1.2.
      call mlwhalf(ro ,roh ,dro,dt,fx,dxi,dxim,ix,fz,dzi,dzim,jx)
      call mlwsrch(roh ,dro ,dt,ss,ix,jx)
```

```
! The calculation of the 0-th and 1-st steps for physical quantity 'ro'
! which corresponds eq. (1.43) and (1.44) are done here.
! Pay attention that at 0-th step 'roh' is the physical quantity on the cell updated
! after time span 'dt' and that at 1-st step 'dro' is the difference of physical
! quantity in time in order to update time span 'dt'/2.
c---  energy ---
      do j=1,jx
      do i=1,ix
         vv=vx(i,j)**2+vz(i,j)**2
! 'vv' is used for calculating 'ep'.
         ep    = pr(i,j)*gm/(gm-1.)+0.5*ro(i,j)*vv
! 'ep' is used for calculating flux 'Fx' and 'Fz' in energy conservation equation.
! Refer Table 1.2.
         fx(i,j)= ep*vx(i,j) +(bz(i,j)*ey(i,j))*pi4i
         fz(i,j)= ep*vz(i,j) +(-bx(i,j)*ey(i,j))*pi4i
         ss(i,j)= -fx(i,j)/x(i)
! 'ss' is the source term which corresponds Table 1.2.
      enddo
      enddo
! calculation of fluxes 'Fx' and 'Fz' and source term 'S'.  Refer Table 1.2.
      call mlwhalf(ee ,eeh ,dee ,dt,fx,dxi,dxim,ix,fz,dzi,dzim,jx)
      call mlwsrch(eeh ,dee ,dt,ss,ix,jx)
! Be careful about that at 0-th step 'eeh' is the physical quantity
! on the cell updated after time span 'dt' and that at 1-st step 'dee' is the difference
! of physical quantity in time in order to update time span 'dt'/2.
c---  x-momentum ---
      do j=1,jx
      do i=1,ix
         fx(i,j)= ro(i,j)*vx(i,j)**2+pr(i,j)
     &          +pi8i*(bz(i,j)**2-bx(i,j)**2)
         fz(i,j)= ro(i,j)*vx(i,j)*vz(i,j)-pi4i*bx(i,j)*bz(i,j)
         ss(i,j)= -(ro(i,j)*(vx(i,j)**2)
     &              +pi4i*(-bx(i,j)**2))/x(i)
      enddo
      enddo
! calculation of fluxes 'Fx' and 'Fz' and source term 'S' of X-direction.  Refer Table 1.2.

      call mlwhalf(rx,rxh,drx,dt,fx,dxi,dxim,ix,fz,dzi,dzim,jx)
      call mlwsrch(rxh ,drx ,dt,ss,ix,jx)
! Be careful about that at 0-th step 'rxh' is the physical quantity
! on the cell updated after time span 'dt', and that at 1-st step 'drx' is the difference
! of physical quantity in time in order to update time span 'dt'/2.
c---  z-momentum ---
      do j=1,jx
      do i=1,ix
         fx(i,j)= ro(i,j)*vz(i,j)*vx(i,j)-pi4i*bz(i,j)*bx(i,j)
         fz(i,j)= ro(i,j)*vz(i,j)**2+pr(i,j)
     &          +pi8i*(bx(i,j)**2-bz(i,j)**2)
         ss(i,j)= -fx(i,j)/x(i)
      enddo
      enddo
! calculation of fluxes 'Fx' and 'Fz' and source term 'S'
! of Z-direction.  Refer Table 1.2.
      call mlwhalf(rz,rzh,drz,dt,fx,dxi,dxim,ix,fz,dzi,dzim,jx)
      call mlwsrch(rzh ,drz ,dt,ss,ix,jx)
        ! Be careful about that at 0-th step 'rzh' is the physical quantity
    ! on the cell updated after time span 'dt', and that at 1-st step 'drz' is the
    ! difference of physical quantity in time in order to update time span 'dt'/2.
```

```
c---  x-magnetic ---
      do j=1,jx
      do i=1,ix
         fx(i,j)= 0.
         fz(i,j)=-ey(i,j)
      enddo
      enddo
! calculate fluxes 'Fx' and 'Fz' in induction equation of the magnetic field of X-direction.
! Refer Table 1.2.
! Because the source term is 0 so that it is not calculated here.
      call mlwhalf(bx,bxh,dbx,dt,fx,dxi,dxim,ix,fz,dzi,dzim,jx)
! Be careful about that at 0-th step 'bxh' is the physical quantity on the cell updated
! after time span 'dt' and that at 1-st step 'dbx' is the difference of physical
! quantity in time in order to update time span 'dt'/2.
! Because the source term is 0 so that 'mlwsrch.f' is not used.
c---  z-magnetic ---
      do j=1,jx
      do i=1,ix
         fx(i,j)= ey(i,j)
         fz(i,j)= 0.
         ss(i,j)= -fx(i,j)/x(i)
      enddo
      enddo
! calculate fluxes 'Fx' and 'Fz' and source term 'S' in induction equation of the
! magnetic field of X-direction.  Refer Table 1.2.

      call mlwhalf(bz,bzh,dbz,dt,fx,dxi,dxim,ix,fz,dzi,dzim,jx)
      call mlwsrch(bzh ,dbz ,dt,ss,ix,jx)
! Be careful about that at 0-th step 'bzh' is the physical quantity
! on the cell updated after time span 'dt' and that at 1-st step 'dbz' is the difference
! of physical quantity in time in order to update time span 'dt'/2.

c---  y-magnetic potential ---
      do j=1,jx
      do i=1,ix
         ss(i,j)= -ey(i,j)
      enddo
      enddo
      call mlwsrch(ayh ,day ,dt,ss,ix,jx)
! calculate vector potential
! 'ss(i,j)= -ey(i,j)' is easily derived from vector potential and induction equations.

c-----------------------------------------------------------------------|
c     convert from total energy to pressure
c-----------------------------------------------------------------------|
      do j=1,jx-1
      do i=1,ix-1
         vxh(i,j)   = rxh(i,j)/roh(i,j)
         vzh(i,j)   = rzh(i,j)/roh(i,j)
         vv=vxh(i,j)**2+vzh(i,j)**2
         bb=bxh(i,j)**2+bzh(i,j)**2
         prh(i,j)   = (gm-1)*(eeh(i,j)-0.5*roh(i,j)*vv-pi8i*bb)
      enddo
      enddo      call mlwhalf(rz,rzh,drz,dt,fx,dxi,dxim,ix,fz,dzi,dzim,jx)
      call mlwsrch(rzh ,drz ,dt,ss,ix,jx)
! Be careful about that at 0-th step 'rzh' is the physical quantity
! on the cell updated after time span 'dt', and that at 1-st step 'drz' is the
! difference of physical quantity in time in order to update time span 'dt'/2.
```

```
! Physical quantities such as 'vxh', 'vzh' and 'prh' of which have not calculated from
! 'dt' updated variables  'roh', 'eeh', 'rxh', 'rzh', 'bxh', and 'bzh'.
! With this procedure, we can calculate the flux on the cell at 'dt' updated.
      do j=1,jx
      do i=1,ix
        eyh(i,j)=-vzh(i,j)*bxh(i,j)+vxh(i,j)*bzh(i,j)
      enddo
      enddo
! 'eyh' corresponds 'ey' in the 1-st step.
! This quantity is used in calculating the flux of energy conservation and also used
! as the flux in induction equation of the magnetic field.  Refer table 1.2.
c-----------------------------------------------------------------------|
c     step intermediate results for full step
c-----------------------------------------------------------------------|
! From here, 2-nd step calculation starts.
c---  density  ---
      do j=1,jx-1
      do i=1,ix-1
         fx(i,j)= roh(i,j)*vxh(i,j)
         fz(i,j)= roh(i,j)*vzh(i,j)
         ss(i,j)= -fx(i,j)/xm(i)
      enddo
      enddo
! 'fx' is flux Fx, 'fz' is flux Fz, and 'ss' is the source term.
! Calculation of fluxes 'Fx' and 'Fz' and source term 'S' in
! mass conservation equation.  Refer table 1.2.
      call mlwfull(dro ,dt,fx,dxi,ux0,ux1,ix,fz,dzi,uz0,uz1,jx)
      call mlwsrcf(dro,dt,ss,ux0,ux1,ix,uz0,uz1,jx)
! For physical quantity 'roh', 2nd step calculation for eq. (1.46) is done here.
! Be careful on that 'time difference of the physical quantity' from already updated
! 'dt'/2 value to be updated again 'dt'/2 is 'dro'.  Therefore, if we add this 'dro'
! to 'ro', time step is updated 'dt' from 'n' to 'n+1'.
c---  energy     ---
      do j=1,jx-1
      do i=1,ix-1
         vv=vxh(i,j)**2+vzh(i,j)**2
! 'vv' is used for calculating 'ep'.
         ep    = prh(i,j)*gm/(gm-1.)+0.5*roh(i,j)*vv
! 'ep' is used for calculating fluxes 'Fx' and 'Fz' in energy
! conservation equation.  Refer table 1.2.
         fx(i,j)= ep*vxh(i,j)
     &            +(bzh(i,j)*eyh(i,j))*pi4i
         fz(i,j)= ep*vzh(i,j)
     &            +(-bxh(i,j)*eyh(i,j))*pi4i
         ss(i,j)= -fx(i,j)/xm(i)
! 'ss' is the source term which corresponds table 1.2.
      enddo
      enddo
! 'fx' is flux Fx, 'fz' is flux Fz, and 'ss' is the source term.
! Calculation of fluxes 'Fx' and 'Fz' and source term 'S' in
! energy conservation equation.  Refer table 1.2.

      call mlwfull(dee ,dt,fx,dxi,ux0,ux1,ix,fz,dzi,uz0,uz1,jx)
      call mlwsrcf(dee,dt,ss,ux0,ux1,ix,uz0,uz1,jx)
! For physical quantity 'eeh', 2nd step calculation for eq. (1.46) is done here.
! Be careful on that 'time difference of the physical quantity' from already updated
! 'dt'/2 value to be updated again 'dt'/2 is 'dee'.  Therefore, if we add this 'dee'
! to 'ee', time step is updated 'dt' from 'n' to 'n+1'.
```

```
c---  x-momentum ---
      do j=1,jx-1
      do i=1,ix-1
         fx(i,j)= roh(i,j)*vxh(i,j)**2+prh(i,j)
     &          +pi8i*(bzh(i,j)**2-bxh(i,j)**2)
         fz(i,j)= roh(i,j)*vxh(i,j)*vzh(i,j)-pi4i*bxh(i,j)*bzh(i,j)
         ss(i,j)= -(roh(i,j)*(vxh(i,j)**2)
     &              +pi4i*(-bxh(i,j)**2))/xm(i)
      enddo
      enddo
! Calculate fluxes 'Fx' and 'Fz' and the source term 'S' in
! the momentum equation of X-direction. Refer table 1.2.

      call mlwfull(drx,dt,fx,dxi,ux0,ux1,ix,fz,dzi,uz0,uz1,jx)
      call mlwsrcf(drx,dt,ss,ux0,ux1,ix,uz0,uz1,jx)
! For physical quantity 'rxh', 2nd step calculation for eq. (1.46) is done here.
! Be careful on that 'time difference of the physical quantity' from already updated
! 'dt'/2 value to be updated again 'dt'/2 is 'drx'.  Therefore, if we add this 'drx'
! to 'rx', time step is updated 'dt' from 'n' to 'n+1'.
c---  z-momentum ---
      do j=1,jx-1
      do i=1,ix-1
         fx(i,j)= roh(i,j)*vzh(i,j)*vxh(i,j)-pi4i*bzh(i,j)*bxh(i,j)
         fz(i,j)= roh(i,j)*vzh(i,j)**2+prh(i,j)
     &          +pi8i*(bxh(i,j)**2-bzh(i,j)**2)
         ss(i,j)= -fx(i,j)/xm(i)
      enddo
      enddo
! Calculate fluxes 'Fx' and 'Fz' and the source term 'S' in the momentum equation
! of Z-direction.  Refer table 1.2.
      call mlwfull(drz,dt,fx,dxi,ux0,ux1,ix,fz,dzi,uz0,uz1,jx)
      call mlwsrcf(drz,dt,ss,ux0,ux1,ix,uz0,uz1,jx)
! For physical quantity 'rzh', 2nd step calculation for eq. (1.46) is done here.
! Be careful on that 'time difference of the physical quantity' from already updated
! 'dt'/2 value to be updated again 'dt'/2 is 'drz'.  Therefore, if we add this 'drz'
! to 'rz', time step is updated 'dt' from 'n' to 'n+1'.
c---  x-magnetic ---
      do j=1,jx-1
      do i=1,ix-1
         fx(i,j)= 0.
         fz(i,j)= -eyh(i,j)
      enddo
      enddo
! Calculate fluxes 'Fx' and 'Fz' in the induction equation of the magnetic field of
! Z-direction. Refer table 1.2.  Because the source term is 0 and is not calculated.
      call mlwfull(dbx,dt,fx,dxi,ux0,ux1,ix,fz,dzi,uz0,uz1,jx)
! For physical quantity 'bxh', 2nd step calculation for eq. (1.46) is done here.
! Be careful on that 'time difference of the physical quantity' from already updated
! 'dt'/2 value to be updated again 'dt'/2 is 'dbx'.  Therefore, if we add this 'dbx'
! to 'bx', time step is updated 'dt' from 'n' to 'n+1'.
c---  z-magnetic ---
      do j=1,jx-1
      do i=1,ix-1
         fx(i,j)= eyh(i,j)
         fz(i,j)= 0.
         ss(i,j)= -fx(i,j)/x(i)
      enddo
      enddo
! Calculate fluxes 'Fx' and 'Fz' and the source term 'S' in the induction equation
! of the magnetic field of Z-direction.  Refer table 1.2.
```

```
      call mlwfull(dbz,dt,fx,dxi,ux0,ux1,ix,fz,dzi,uz0,uz1,jx)
      call mlwsrcf(dbz,dt,ss,ux0,ux1,ix,uz0,uz1,jx)
! For physical quantity 'bzh', 2nd step calculation for eq. (1.46) is done here.
! Be careful on that 'time difference of the physical quantity' from already updated
! 'dt'/2 value to be updated again 'dt'/2 is 'dbz'.  Therefore, if we add this 'dbz'
! to 'bz', time step is updated 'dt' from 'n' to 'n+1'.
c---  y-magnetic potential ---
      do j=1,jx
      do i=1,ix
         ss(i,j)= -eyh(i,j)
      enddo
      enddo
      call mlwsrcf(day,dt,ss,ux0,ux1,ix,uz0,uz1,jx)
! calculate vector potential
! 'ss(i,j)= -eyh(i,j)' is derived from vector potential and the induction equations.
c-----------------------------------------------------------------------|
c     diffusion coefficients for artificial viscosity
c-----------------------------------------------------------------------|
c     qav=3.0
      zero=0.0
      do j=1,jx-1
      do i=1,ix-1
         qx(i,j)=qav*dxm(i)*max(zero,abs(vx(i+1,j)-vx(i,j))-1.0e-4)
      enddo
      enddo
      do j=1,jx-1
      do i=1,ix
         qz(i,j)=qav*dzm(j)*max(zero,abs(vz(i,j+1)-vz(i,j))-1.0e-4)
      enddo
      enddo
! Here, diffusion coefficients for artificial viscosity is calculated.  Please refer
! section 1.4.5 Introduction of the artificial viscosity and equation (1.47)
c-----------------------------------------------------------------------|
c     apply artificial viscosity
c-----------------------------------------------------------------------|
      call mlwartv(ro,dro,dt,qx,dxi,dxim,ix,qz,dzi,dzim,jx)
      call mlwartv(ee,dee,dt,qx,dxi,dxim,ix,qz,dzi,dzim,jx)
      call mlwartv(rx,drx,dt,qx,dxi,dxim,ix,qz,dzi,dzim,jx)
      call mlwartv(rz,drz,dt,qx,dxi,dxim,ix,qz,dzi,dzim,jx)
      call mlwartv(bx,dbx,dt,qx,dxi,dxim,ix,qz,dzi,dzim,jx)
      call mlwartv(bz,dbz,dt,qx,dxi,dxim,ix,qz,dzi,dzim,jx)
! The amount of artificial viscosity is calculated in the subroutine 'mlwartv.f' and
! added on to time differences of all physical quantities.
! Please refer section 1.4.5 Indtoduction of the artificail viscosity and eq. (1.47)
c-----------------------------------------------------------------------|
c     update internal points
c-----------------------------------------------------------------------|
      do j=2,jx-1
      do i=2,ix-1
         ro(i,j) = ro(i,j) +dro(i,j)
         ee(i,j) = ee(i,j) +dee(i,j)
         rx(i,j) = rx(i,j) +drx(i,j)
         rz(i,j) = rz(i,j) +drz(i,j)
         bx(i,j) = bx(i,j) +dbx(i,j)
         bz(i,j) = bz(i,j) +dbz(i,j)
         ay(i,j) = ay(i,j) +day(i,j)
      enddo
      enddo
! The differences calculated through steps from 0 to 2 are added on the previous
! quantities (at time $t=n$) and update the time 'dt'.
```

```
c-------------------------------------------------------------------------|
c     convert from total energy to pressure
c-------------------------------------------------------------------------|
      do j=2,jx-1
      do i=2,ix-1
         vx(i,j) = rx(i,j)/ro(i,j)
         vz(i,j) = rz(i,j)/ro(i,j)
         vv=vx(i,j)**2+vz(i,j)**2
         bb=bx(i,j)**2+bz(i,j)**2
         pr(i,j) = (gm-1)*(ee(i,j) - 0.5*ro(i,j)*vv - pi8i*bb)
      enddo
      enddo
! At last, using 'dt' updated variables 'ro', 'ee', 'rx', 'rz', 'bx', and 'bz',
! new quantities such as 'vx', 'vz', and 'pr' are calculated.
! With this procedure, all physical quantities on the grid point
! have updated to time span 'dt'.
      return
      end
```

3.5 Acknowledgment

For this chance I would thank Prof. T. Hanawa at Chiba University. My Ph. D. adviser Prof. K. Tomisaka at NAOJ recommends me to attend the "Summer School for Astro and Space Plasma Simulation" and looks after the preparation of this draft so that I would express my deep gratitude to him. The developers of CANS, Prof. R. Matsumoto at Chiba University and Prof. T. Yokoyama at University of Tokyo gave me many comments throughout this manual. Prof. T. Matsumoto at Hosei University, Dr. K. Saigo at NAOJ, and Dr. M. Machida at Chiba University gave me valuable comments on the computation method. And I thank all co-authors who cooperate during revision and peoples of Astrophysical laboratory at Chiba University.

Bibliography

[1] Space Hydrodynamics (1997) Baifukan, Shiro Sakashita and Satoru Ikeuchi

[2] Computation Method for Hydrodynamics (1994) Univ. Tokyo Press, Kouzou Fujii

[3] Text of Numerical Astrophysics Summer School (2000) Koji Tomisaka and Tomoyuki Hanawa

[4] Hydrodynamics (1972) Baifukan, Tomomasa Tatsumi