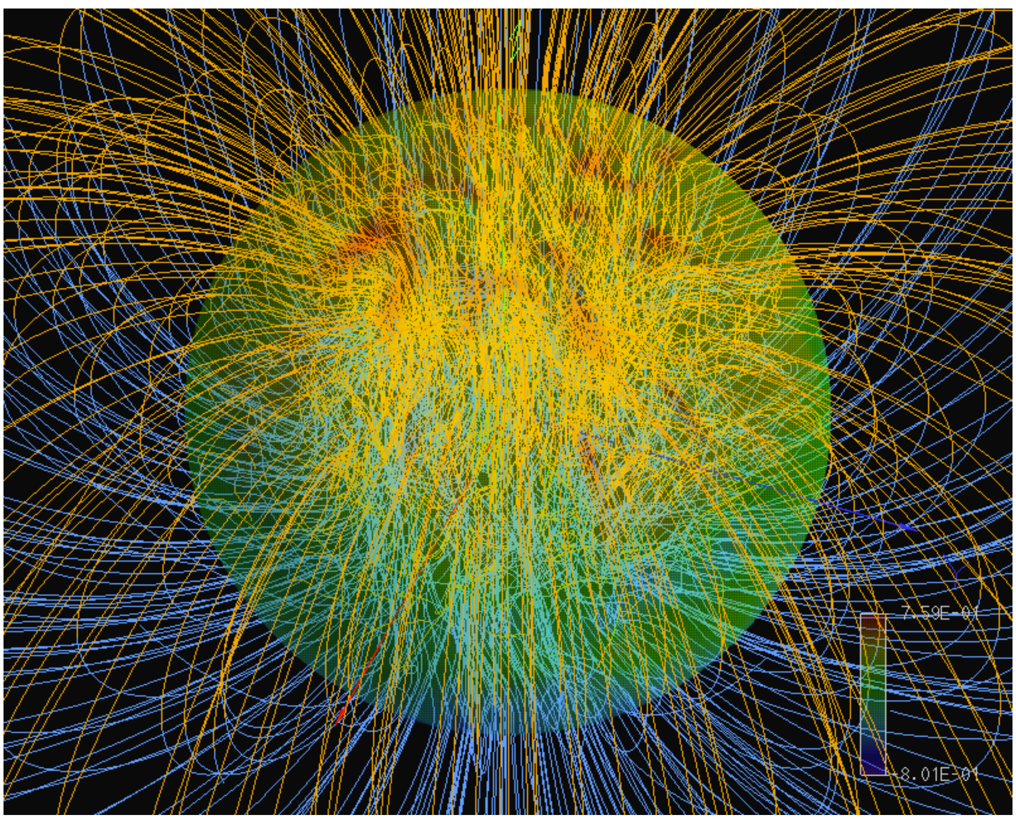
COMPUTATIONAL INFRASTRUCTURE FOR GEODYNAMICS (CIG)

Calypso

User Manual
Version 1.2



Hiroaki Matsui

www.geodynamics.org

Preface

Calypso is a program package of magnetohydrodynamics (MHD) simulations in a rotating spherical shell for geodynamo problems. This package consists of the simulation program, preprocessing program, post processing program to generate field data for visualization programs, and several small utilities. The simulation program runs on parallel computing systems using MPI and OpenMP parallelization.

Contents

1 Introduction

Calypso is a program package for magnetohydrodynamics (MHD) simulations in a rotating spherical shell for geodynamo problems. This package consists of the simulation program, preprocessing program, post processing program to generate field data for visualization programs, and several small utilities. The simulation program runs on parallel computing systems using MPI and OpenMP parallelization.

Calypso solves the equations that govern convection and magnetic-field generation in a rotating spherical shell. Flow is driven by thermal or compositional buoyancy in a Boussinesq fluid. Calypso also support various boundary conditions (e.g. fixed temperature, heat flux, composition, and compositional flux), and permits a conductive and rotatable inner core. Results are written as spherical harmonics coefficients, Gauss coefficients for the region outside of the fluid shell, and field data in Cartesian coordinate for easily visualization with a number of visualization programs.

This user guide describes the essentials of the magnetohydrodynamics theory and equations behind Calypso, and provides instructions for the configuration and execution of Calypso.

2 History

Calypso has its origins in two earlier projects. One is a dynamo simulation code written by Hiroaki Matsui in 1990's using a spectral method. This code solves for the poloidal and toroidal spectral coefficients, like Calypso, but it calculates the nonlinear terms in the spectral domain using a parallelization for SMP architectures. The other project is the thermal convection version of GeoFEM, which is Finite Element Method (FEM) platform for massively parallel computational environment, originally written by Hiroshi Okuda in 2000.

Hiroaki Matsui was responsible for adding routines to GeoFEM to perform magnetohydrodynamics simulation in a rotating frame. In 2002 this code successfully performed dynamo simulations in a rotating spherical shell using insulating magnetic boundary conditions. The following year Matsui implemented a subgrid scale (SGS) model in the FEM dynamo model in collaboration with Bruce Buffett. A module to solve for double diffusive convection was added to the FEM dynamo model by Hiroaki Matsui in 2009.

Progress in understanding the role of subgrid scale models in magnetohydrodynamic simulations relies on quantitative estimates for the transfer of energy between spatial scales. This information is most easily obtained from a spherical harmonic expansion of the simulation results, even when the simulation is performed by FEM. Hiroaki Matsui implemented the spherical harmonic transform in 2007 using a combination of MPI and

OpenMP, and later included the spherical harmonic transform routines into his old dynamo code to create Calypso. Additional software in the program package for visualization is based on data formats from the FEM model. In addition, the control parameter file format is adapted from the input formats used in GeoFEM.

Calypso Ver. 1.0 supports the following features and capabilities

- Magnetohydrodynamics simulation for a Boussinesq fluid in a rotating spherical shell.
- Convection driven by thermal and compositional buoyancy.
- Temperature or heat flux is fixed at boundaries
- Composition or compositional flux is fixed at boundaries
- Non-slip or free-slip boundary conditions
- Outside of the fluid shell is electrically insulated or pseudo vacuum boundary.
- A conductive inner core with the same conductivity as the surrounding fluid
- A rotating inner core driven by the magnetic and viscous torques.

2.1 Updates for Ver 1.1

In Version 1.1, a number of bug fixes and additional comments for Doxygen are completed. The following large bugs are fixed:

- `configure` command is updated to find appropriate GNU make command. (see Section 6.1)
- Label for radial grid type in the file `ctl_sph_shell` `raidal_grid_type_ctl` is changed to `radial_grid_type_ctl`. If the old name is used in the control file, program `gen_sph_grid` will crash.

And, the following features are implemented

- New ordering is used for spherical harmonics data to reduce communication time. The old version of spectrum indexing data, which is generated by `gen_sph_grids` in Ver. 1.0 is also supported in Ver. 1.1.

- Evaluation of Coriolis term is updated. Now, Adams-Gaunt integrals are evaluated in the initialization process in the simulation program `sph_mhd`, so the data file for Adams-Gaunt integrals which is made by `gen_sph_grids` is not required.

- Add a program `sph_add_initial_field`. to modify existed initial field data. This program is used to modify or add new fields in spectrum data. (See Section 13.)

- Heat and composition source terms are implemented. These source terms are fixed with time, and defined as spectrum data. The source terms are defined by using initial field generation program
`sph_initial_field` or `sph_add_initial_field`. (See section 12 and 13.)

- The boundary conditions for temperature and composition can be defined by using spherical harmonics coefficients. (i.e. inhomogeneous boundary conditions can be applied.) These boundary conditions are defined by using single external data file. (See Section 10.3)

3 Acknowledgements

Calypso was primarily developed by Dr. Hiroaki Matsui in collaboration with Prof. Bruce Buffett at the University of California, Berkeley. The following NSF grants supported the development of Calypso,

- B.A. Buffett, NSF EAR-0509893; Models of sub-grid scale turbulence in the Earths core and the geodynamo; 2005 - 2007.

- B.A. Buffett and D. Lathrop, NSF EAR-0652882; CSEDI Collaborative Research: Integrating numerical and experimental geodynamo models, 2007 - 2009

- B.A. Buffett, NSF EAR-1045277; Development and application of turbulence models in numerical geodynamo simulations ; 2010 - 2012

4 Citation

Computational Infrastructure for Geodynamics (CIG) and the Calypso developers are making the source code to Calypso available to researchers in the hope that it will aid their research and teaching. A number of individuals have contributed a significant amount of time and energy into the development of Calypso. We request that you cite the appropriate papers and make acknowledgements as necessary. The Calypso development team asks that you cite the following papers:

Matsui, H., E. King, and B.A. Buffett, Multi-scale convection in a geodynamo simulation with uniform heat flux along the outer boundary, *Geochemistry, Geophysics, Geosystems*, **15**, 3212 – 3225, 2014.

5 Model of Simulation

5.1 Governing equations

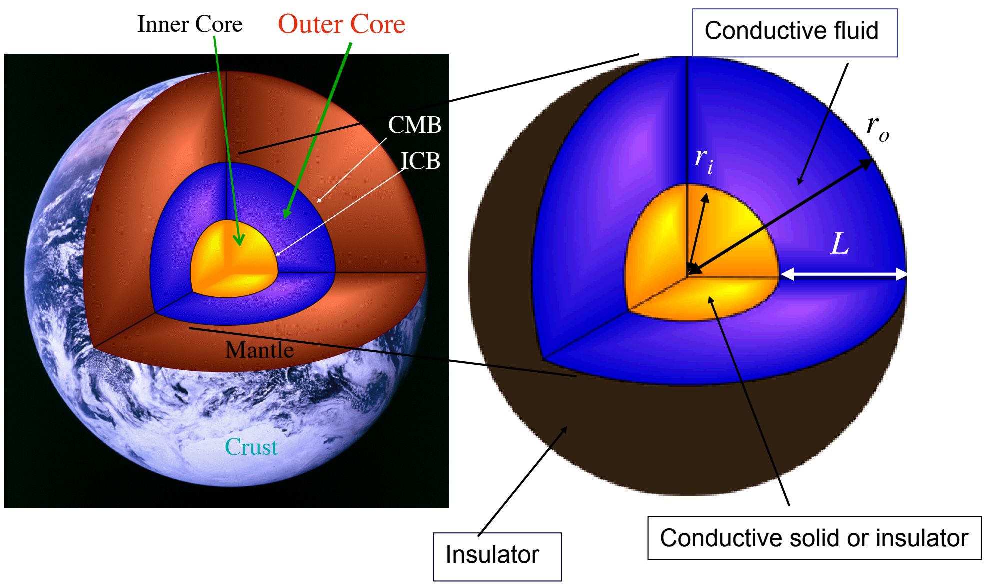


Figure 1: Rotating spherical shell modeled on the Earth's outer core.

This model performs a magnetohydrodynamics (MHD) simulation in a rotating spherical shell modeled on the Earth's outer core (see Figure 1). We consider a spherical shell from the inner core boundary (ICB) to the core mantle Boundary (CMB) in a rotating frame which constantly rotates with angular velocity $\boldsymbol{\Omega} = \Omega \hat{z}$. The fluid shell is filled with a conductive fluid with constant diffusivities (kinematic viscosity ν, magnetic diffusivity η, thermal diffusivity κ_T, and compositional diffusivity κ_C). The inner core ($0 < r < r_i$) is solid, and may be considered an electrical insulator or may have the same conductivity as the outer core. We assume that the region outside of the core is an electrical insulator. The rotating spherical shell is filled with Boussinesq modeled fluid. The governing equations of the MHD dynamo problem are the following,

$$\begin{aligned}
\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{\omega} \times \boldsymbol{u}) &= -\nabla \left(P + \frac{1}{2} u^2 \right) - \nu \nabla \times \nabla \times \boldsymbol{u} \\
&\quad -2\Omega \left(\hat{z} \times \boldsymbol{u} \right) + \left(\frac{\rho}{\rho_0} \boldsymbol{g} \right) + \frac{1}{\rho_0} \left(\boldsymbol{J} \times \boldsymbol{B} \right),
\end{aligned}$$

$$
\begin{aligned}
\frac{\partial \boldsymbol{B}}{\partial t} &= -\eta \nabla \times \nabla \times \boldsymbol{B} + \nabla \times (\boldsymbol{u} \times \boldsymbol{B}), \\
\frac{\partial T}{\partial t} + (\boldsymbol{u} \cdot \nabla) T &= \kappa_T \nabla^2 T + q_T, \\
\frac{\partial C}{\partial t} + (\boldsymbol{u} \cdot \nabla) C &= \kappa_C \nabla^2 C + q_C, \\
\nabla \cdot \boldsymbol{u} &= \nabla \cdot \boldsymbol{B} = 0, \\
\boldsymbol{\omega} &= \nabla \times \boldsymbol{u},
\end{aligned}
$$

and

$$
\boldsymbol{J} = \frac{1}{\mu_0} \nabla \times \boldsymbol{B},
$$

where, $\boldsymbol{u}$, $\boldsymbol{\omega}$, P, $\boldsymbol{B}$, $\boldsymbol{J}$, T, C, q_T, and q_C are the velocity, vorticity, pressure, magnetic field, current density, temperature, compositional variation, heat source, and source of light element, respectively. Coefficients in the governing equations are the kinetic viscosity ν, thermal diffusivity κ_T, compositional diffusivity κ_C, and magnetic diffusivity η. The density ρ is written as a function of T, C, average density ρ_0, thermal expansion α_T, and density ratio of light element to main composition α_C,

$$
\rho = \rho_0 \left[1 - \alpha_T (T - T_0) - \alpha_C (C - C_0)\right]
$$

In Calypso, the vorticity equation and divergence of the momentum equation are used for solving $\boldsymbol{u}$, $\boldsymbol{\omega}$, and P as,

$$
\begin{aligned}
\frac{\partial \boldsymbol{\omega}}{\partial t} + \nabla \times (\boldsymbol{\omega} \times \boldsymbol{u}) &= -\nu \nabla \times \nabla \times \boldsymbol{\omega} - 2\Omega \nabla \times (\hat{z} \times \boldsymbol{u}) \\
&\quad + \nabla \times \left(\frac{\rho}{\rho_0} \boldsymbol{g} \right) + \frac{1}{\rho_0} \nabla \times (\boldsymbol{J} \times \boldsymbol{B}),
\end{aligned}
$$

and

$$
\begin{aligned}
\nabla \cdot (\boldsymbol{\omega} \times \boldsymbol{u}) &= -\nabla^2 \left(P + \frac{1}{2} u^2 \right) - 2\Omega \nabla \cdot (\hat{z} \times \boldsymbol{u}) \\
&\quad + \nabla \cdot \left(\frac{\rho}{\rho_0} \boldsymbol{g} \right) + \frac{1}{\rho_0} \nabla \cdot (\boldsymbol{J} \times \boldsymbol{B}).
\end{aligned}
$$

5.2 Spherical harmonics expansion

In Calypso, fields are expanded into spherical harmonics. A scalar field (for example, temperature $T(r, \theta, \phi)$) is expanded as

$$T(r, \theta, \phi) = \sum_{l=0}^{L} \sum_{m=-l}^{l} T_l^m(r) Y_l^m(\theta, \phi),$$

where Y_l^m are the spherical harmonics. Solenoidal fields (e.g. velocity $\boldsymbol{u}$, vorticity $\boldsymbol{\omega}$, magnetic field $\boldsymbol{B}$, and current density $\boldsymbol{J}$) are decomposed into poloidal and toroidal components. For example, the magnetic field is described as

$$\boldsymbol{B}(r, \theta, \phi) = \sum_{l=1}^{L} \sum_{m=-l}^{l} \left(\boldsymbol{B}_{Sl}^{m} + \boldsymbol{B}_{Tl}^{m} \right),$$

where

$$\begin{aligned} \boldsymbol{B}_{Sl}^{m}(r, \theta, \phi) &= \nabla \times \nabla \times \left(B_{Sl}^{m}(r) Y_l^m(\theta, \phi) \hat{r} \right), \\ \boldsymbol{B}_{Tl}^{m}(r, \theta, \phi) &= \nabla \times \left(B_{Tl}^{m}(r) Y_l^m(\theta, \phi) \hat{r} \right). \end{aligned}$$

The spherical harmonics are defined as real functions. $P_l^m \cos(m\phi)$ is assigned for positive m, $P_l^m \sin(m\phi)$ is assigned for negative m, where P_l^m are Legendre polynomials. Because Schmidt quasi normalization is used for the Legendre polynomials P_l^m, the orthogonality relation for the spherical harmonics is

$$\int Y_l^m Y_{l'}^{m'} \sin\theta d\theta d\phi = 4\pi \frac{1}{2l+1} \delta_{ll'} \delta_{mm'},$$

where, $\delta_{ll'}$ is Kronecker delta.

5.3 Evaluation of Coriolis term

The curl of the Coriolis force $-2\Omega \nabla \times (\hat{z} \times \boldsymbol{u})$ is evaluated in the spectrum space using the triple products of the spherical harmonics. These 3j-symbols (or Gaunt integral $G_{Lll'}^{Mmm'}$ and Elsasser integral $E_{Lll'}^{Mmm'}$) are written as

$$\begin{aligned} G_{Lll'}^{Mmm'} &= \int Y_L^M Y_l^m Y_{l'}^{m'} \sin\theta d\theta d\phi, \\ E_{Lll'}^{Mmm'} &= \int Y_L^M \left(\frac{\partial Y_l^m}{\partial \theta} \frac{\partial Y_{l'}^{m'}}{\partial \phi} - \frac{\partial Y_l^m}{\partial \phi} \frac{\partial Y_{l'}^{m'}}{\partial \theta} \right) d\theta d\phi. \end{aligned}$$

The Gaunt integral $1/(4\pi) G_{Lll'}^{Mmm'}$ and Elsasser integral $1/(4\pi) E_{Lll'}^{Mmm'}$ for the Coriolis terms are evaluated in the simulation program.

5.4 Boundary conditions

Calypso currently supports the following boundary conditions for velocity $\boldsymbol{u}$, magnetic field $\boldsymbol{B}$, temperature T, and composition variation C. These boundary conditions are defined in the control file `control_MHD`.

5.4.1 Non-slip boundary

The velocity $\boldsymbol{u}$ is set to be 0 at the boundary. For poloidal and toroidal coefficients of velocity, $U_{Sl}^{m}(r)$ and $U_{Tl}^{m}(r)$, the boundary condition can be described as

$$U_{Sl}^{m}(r) = \frac{\partial U_{Sl}^{m}}{\partial r} = 0,$$

and

$$U_{Tl}^{m}(r) = 0.$$

5.4.2 Free-slip boundary

For a free slip boundary, shear stress and radial flow vanish at the boundary. The boundary condition for poloidal and toroidal coefficients are described as

$$U_{Sl}^{m}(r) = \frac{\partial^2}{\partial r^2}\left(\frac{1}{r}U_{Sl}^{m}(r)\right) = 0,$$

and

$$\frac{\partial}{\partial r}\left(\frac{1}{r^2}U_{Tl}^{m}(r)\right) = 0.$$

5.4.3 Fixed rotation rate

If the boundary rotates with a rotation vector $\boldsymbol{\Omega}_b = (\Omega_{bx}, \Omega_{by}, \Omega_{bz})$, the boundary conditions for poloidal and toroidal coefficients are described as

$$\begin{aligned}
U_{Sl}^{m}(r) &= \frac{\partial U_{Sl}^{m}}{\partial r} = 0,\\
U_{T1}^{1s}(r) &= r^2\Omega_{by},\\
U_{T1}^{0}(r) &= r^2\Omega_{bz},\\
U_{T1}^{1c}(r) &= r^2\Omega_{bx},
\end{aligned}$$

and

$$U_{Tl}^{m}(r) = 0 \text{ for } l > 2.$$

5.4.4 Fixed homogenous temperature

When a constant temperature T_b is is applied, the spherical harmonic coefficients are

$$T_0^0(r) = T_b,$$

and

$$T_l^m(r) = 0 \text{ for } l > 1.$$

5.4.5 Fixed homogenous heat flux

A constant heat flux is imposed by setting the radial temperature gradient to F_{Tb}. The spherical harmonic coefficients are

$$\frac{\partial T_0^0}{\partial r} = F_{Tb},$$

and

$$\frac{\partial T_l^m}{\partial r} = 0 \text{ for } l > 1.$$

5.4.6 Fixed composition

When a constant composition C_b is applied, the spherical harmonic coefficients are

$$C_0^0(r) = C_b,$$

and

$$C_l^m(r) = 0 \text{ for } l > 1.$$

5.4.7 Fixed composition flux

A constant composition flux is imposed by setting the radial composition gradient to F_{Cb}. The spherical harmonic coefficients are

$$\frac{\partial C_0^0}{\partial r} = F_{Cb},$$

and

$$\frac{\partial C_l^m}{\partial r} = 0 \text{ for } l > 1.$$

5.4.8 Connection to the magnetic potential field

If the regions outside the fluid shell are assumed to be electrical insulators, current density vanishes in the electric insulator

$$\boldsymbol{J}_{ext} \quad = \quad 0,$$

where the suffix $_{ext}$ indicates fields outside of the fluid shell. At the boundaries of the fluid shell, the magnetic field $\boldsymbol{B}_{fluid}$, current density $\boldsymbol{J}_{fluid}$, and electric field $\boldsymbol{E}_{fluid}$ in the conductive fluid satisfy:

$$(\boldsymbol{B}_{fluid} - \boldsymbol{B}_{ext}) = 0,$$
$$(\boldsymbol{J}_{fluid} - \boldsymbol{J}_{ext}) \cdot \hat{r} = 0,$$

and

$$(\boldsymbol{E}_{fluid} - \boldsymbol{E}_{ext}) \times \hat{r} = 0,$$

where, $\hat{r}$ is the radial unit vector (i.e. normal vector for the spherical shell boundaries). Consequently, radial current density $\boldsymbol{J}$ vanishes at the boundary as

$$\boldsymbol{J} \cdot \hat{r} = 0 \text{ at } r = r_i, r_o$$

In an electrical insulator the magnetic field can be described as a potential field

$$\boldsymbol{B}_{ext} = -\nabla W_{ext},$$

where W_{ext} is the magnetic potential. The boundary conditions can be satisfied by connecting the magnetic field in the fluid shell at boundaries to the potential fields. The magnetic field is connected to the potential field in an electrical insulator. At CMB ($r = r_o$), the boundary condition can be described by the poloidal and toroidal coefficients of the magnetic field as

$$\frac{l}{r} B_{Sl}^{m}(r) \quad = \quad -\frac{\partial B_{Sl}^{m}}{\partial r},$$

and

$$B_{Tl}^{m}(r) \quad = \quad 0.$$

If the inner core is also assumed to be an insulator, the magnetic boundary conditions for ICB ($r = r_i$) can be described as

$$\frac{l+1}{r} B_{Sl}^{m}(r) \quad = \quad \frac{\partial B_{Sl}^{m}}{\partial r},$$

and

$$B_{Tl}^{m}(r) \quad = \quad 0.$$

5.4.9 Magnetic boundary condition for center

If the inner core has the same conductivity as the outer core, we solve the induction equation for the inner core as for the outer core with the boundary conditions for the center. The poloidal and toroidal coefficients at center are set to

$$B_{Sl}^{m}(0) = B_{Tl}^{m}(0) = 0.$$

5.4.10 Pseudo-vacuum magnetic boundary condition

Under the pseudo-vacuum boundary condition, the magnetic field has only a radial component at the boundaries. Considering the conservation of the magnetic field, the magnetic boundary condition will be

$$\frac{\partial}{\partial r}\left(r^2 B_r\right) = B_\theta = B_\phi = 0 \text{ at } r = r_i, r_o.$$

The present boundary condition is also described by using the poloidal and toroidal coefficients as

$$\frac{\partial B_{Sl}^{m}}{\partial r} = B_{Tl}^{m}(r) = 0 \text{ at } r = r_i, r_o.$$

6 Installation

6.1 Library Requirements

Calypso requires the following libraries.

- GNU make
- MPI libraries (OpenMPI, MPICH, etc)
- FFTPACK Ver 5.1D (`http://people.sc.fsu.edu/~jburkardt/f_src/fftpack5.1d/fftpack5.1d.html`). The source files for FFTPACK are included in `src/EXTERNAL_libs` directory.

Linux and Max OS X use GNU make as a default 'make' command, but some system (e.g. BSD or SOLARIS) does not use GNU make as default. `configure` command searches and set correct GNU make command.

In addition, the following environment and libraries can be used (optional).

- OpenMP
- BLAS
- FFTW version 3 (`http://www.fftw.org`) including Fortran wrapper
- PARALLEL HDF5 (`http://www.hdfgroup.org/HDF5/PHDF5`) including Fortran wrapper.

Note: Calypso does NOT use MPI and OpenMP features in FFTW3.

In the most of platforms, the Fourier transform by FFTW is faster than that by FFTPACK.

HDF5 is used for field data output with XDMF format instead of VTK format. The comparison of field data format is described in section refsec:VTK.

OpenMP is used for the parallelization under the shared memory. Better choice to use both MPI and OpenMP parallelization (so-called Hybrid parallelization) or only using MPI (so-called flat MPI) is depends on the computational platform and compiler. For example, flat MPI has much better performance on Linux cluster with Intel Xeon processors and with Intel fortran compiler, but Hybrid model has better performance on Hitachi SR16000 with Power 6 processors.

6.2 Known problems

FFTPACK and Intel compiler

FFTPACK fails to compile with Intel fortran using the '-warn all' option. Currently the '-warn all' option is excluded by Makefile when FFTPACK is compiled.

Homebrew's FFTW3 on Mac OS X

Calypso uses Fortran wrappers in FFTW3. If FFTW3 is installed using Homebrew for Mac OS X (http://mxcl.github.com/homebrew/), the required fortran wrappers are not installed. In this case, please install FFTW3 with Fortran wrappers with another package manager (Macports (http://www.macports.org, for example), build FFTW3 by yourself including the Fortran wrapper, or turn off FFTW3 features in Calypso.

XL fortran

In XL fortran, preprocessor options is not specified by `-D...`, but `-Wf, '-D...'`. Pleease edit preprocessor macro opthion `F90CPPFLAGS` in `work/Makefile` by an editor.

Cross compiler support

`configure` command in Calypso does not support cross compilation. If you want to compile with a cross compiler, please set the variables in Makefile manually (see section 6.6)

6.3 Directories

The top directory of Calypso (ex. [CALYPSO_HOME]) contains the following directories.

```
% cd [CALYPSO_HOME]
% ls
CMakeLists.txt Makefile.in configure.in examples
INSTALL bin doc src
LICENSE configure doxygen work
```

`bin`: directory for executable files

`cmake`: directory for cmake configurations `cmake:` directory for document generated by doxygen

`doc:` documentations

`examples:` examples

`src:` source files

`work:` work directory. Compile is done in this directory.

6.4 Doxygen

Doxygen (`http://www.doxygen.org`) is an powerful document generation tool from source files. We only save a configuration file in this directory because thousands of html files generated by doxygen. The documents for source codes are generated by the following command:

```
% cd [CALYPSO_HOME]/doxygen
% doxygen ./Doxyfile_CALYPSO
```

The html documents can see by opening `[CALYPSO_HOME]/doxygen/html/index.html`. Automatically generated documentation is also available on the CIG website at `http://www.geodynamics.org/cig/software/calypso/`.

6.5 Install using `configure` command

6.5.1 Configuration using `configure` command

Calypso uses the configure script for configuration to install. The simplest way to install programs is the following process in the top directory of Calypso.

```
%pwd
[CALYPSO_HOME]
% ./configure
...
% make
...
% make install
```

After the installation, object modules can be deleted by the following command;

```
% make clean
```

./configure generates a Makefile in the current directory. Available options for `configure` can be checked using the `./configure --help` command. The following options are available in the `configure` command.

```
Optional Features:
  --disable-option-checking  ignore unrecognized --enable/--with options
  --disable-FEATURE       do not include FEATURE (same as --enable-FEATURE=no)
  --enable-FEATURE[=ARG]  include FEATURE [ARG=yes]
  --enable-fftw3          Use fftw3 library
 Optional Packages:
  --with-PACKAGE[=ARG]    use PACKAGE [ARG=yes]
  --without-PACKAGE       do not use PACKAGE (same as --with-PACKAGE=no)
  --with-hdf5=yes/no/PATH full path of h5pcc for parallel HDF5 configuration
  --with-blas=<lib>       use BLAS library <lib>

Some influential environment variables:
  CC          C compiler command
  CFLAGS      C compiler flags
  LDFLAGS     linker flags, e.g. -L<lib dir> if you have libraries in a
              nonstandard directory <lib dir>
  LIBS        libraries to pass to the linker, e.g. -l<library>
  CPPFLAGS    (Objective) C/C++ preprocessor flags, e.g. -I<include dir> if
              you have headers in a nonstandard directory <include dir>
  FC          Fortran compiler command
  FCFLAGS     Fortran compiler flags
  MPICC       MPI C compiler command
  MPIFC       MPI Fortran compiler command
  PKG_CONFIG  path to pkg-config utility
  CPP         C preprocessor
  FFTW3_CFLAGS
              C compiler flags for FFTW3, overriding pkg-config
  FFTW3_LIBS  linker flags for FFTW3, overriding pkg-config
```

An example of usage of the configure command is the following;

```
% ./configure --prefix='/Users/matsui/local' \
? CFLAGS='-O -Wall -g' FCFLAGS='-O -Wall -g' \
? PKG_CONFIG_PATH='/Users/matsui/local/lib/pkgconfig' \
? --enable-fftw3 --with-hdf5='/Users/matsui/local/bin/h5pcc'
```

6.5.2 Compile

Compile is performed using the `make` command. The Makefile in the top directory is used to generate another Makefile in the `work` directory, which is automatically used to complete the compilation. The object file and libraries are compiled in the `work` directory. Finally, the executive files are assembled in `bin` directory. You should find the following programs in the `bin` directory.

`gen_sph_grids`: Preprocessing program for data transfer for spherical transform

`sph_mhd`: Simulation program

`sph_initial_field`: Example program to generate initial field

`sph_add_initial_field`: Example program to add initial field in existing spectum data

`sph_snapshot`: Data transfer from spectrum data to field data

`sph_dynamobench`: Data processing for dynamo benchmark test by Christensen *et. al.* (2002)

`sph_zm_snapshot`: Generate zonal mean field

`assemble_sph`: Data transfer program to change number of subdomains.

`t_ave_sph_mean_square`: Time averaging program for the mean square data.

`t_ave_picked_sph_coefs`: Time averaging program for the picked spectrum data.

`t_ave_nusselt`: Time averaging program for the Nusselt number data.

`check_sph_grids`: Check program for tests.

`make_f90depends`: Program to generate dependency of the source code (`make` command uses to generate `work/Makefile`)

The following library files are also made in `work` directory.

`libcalypso.a`: Calypso library

`libfftpack.5d.a`: FFTPACK 5.1 library

6.5.3 Clean

The object and fortran module files in `work` directory is deleted by typing

```
% make clean
```

This command deletes files with the extension `.o`, `.mod`, `.par`, `.diag`, and `.`.

6.5.4 Install

The executive files are copied to the install directory `$(INSTDIR)/bin`. The install directory `$(INSTDIR)` is defined in Makefile, and can also set by `${--prefix}` option for `configure` command. Alternatively, you can use the programs in `${SRCDIR}/bin` directory without running `make install`. If directory `${PREFIX}` does not exist, `make install` creates `${PREFIX}`, `${PREFIX}/lib`, `${PREFIX}/bin`, and `${PREFIX}/include` directories. No files are installed in `${PREFIX}/lib` and `${PREFIX}/include`.

6.6 Install without using configure

It is possible to compile Calypso without using the `configure` command. To do this, you need to edit the `Makefile`. First, copy `Makefile` from template `Makefile.in` as

```
% cp Makefile.in Makefile
```

In Makefile, the following variables should be defined.

`SHELL` Name of shell command.

`SRCDIR` Directory of this Makefile.

`INSTDIR` Install directory.

`MPICHDIR` Directory names for MPI implementation. If you set fortran90 compiler name for MPI programs in `MPIF90`, you do not need to define this valuable.

`MPICHINCDIR` Directory names for include files for MPI implementation. If you set fortran90 compiler name for MPI programs in `MPIF90`, you do not need to define this valuable.

`MPILIBS` Library names for MPI implementation. If you set fortran90 compiler name for MPI programs in `MPIF90`, you do not need to define this valuable.

F90_LOCAL Command name of local Fortran 90 compiler to compile module dependency listing program.

MPIF90 Command name of Fortran90 compiler and linker for MPI programs. If command does not have MPI implementation, you need to define the definition of MPI libraries MPICHDIR, MPICHINCDIR, and MPILIBS.

AR Command name for archive program (ex. ar) to generate libraries. If you need some options for archive command, options are also included in this valuable.

RANLIB Command name for ranlib to generate index to the contents of an archive. If system does not have ranlib, set true in this valuable. true command does not do anything for libraries.

F90OPTFLAGS Optimization flags for Fortran90 compiler (including OpenMP flags)

FFTW3_CFLAGS Option flags for FFTW3 (ex. -I/usr/local/include)

FFTW3_LIBS Library lists for FFTW3 (ex. -L/usr/local/lib -lfftw3 -lm)

HDF5_FFLAGS Option flags to compile with HDF5. This setting can be found by using hfd5 command h5pfc -show.

HDF5_LDFLAGS Option flags to link with HDF5. This setting can be found by using hfd5 command h5pfc -show.

HDF5_FLIBS Library lists for HDF5. This setting can be found by using hfd5 command h5pfc -show.

6.7 Install using cmake

CMake is a cross-platform, open-source build system. CMake can be downloaded from http://www.cmake.org. The following procedure is required to install.

1. Create working directory (you can also use [CALYPSO_HOME]/work).
2. Generate Makefile and working directories by cmake command.
3. Compile programs by make command.

In this section, `[CALYPSO\_HOME]/work` is used as the working directory. Options for CMake can be checked by `cmake -i [CALYPSO_HOME]` command at `[CALYPSO_HOME]/work`. There are a number of options can be found, but the following valuables are important settings for installation:

`CMAKE_INSTALL_PREFIX` Install directory

`CMAKE_Fortran_COMPILER` Fortran90 compiler.

`CMAKE_DISABLE_FIND_PACKAGE_OpenMP_Fortran` OpenMP is not used if 'yes' is set in this valuable.

`CMAKE_DISABLE_FIND_PACKAGE_FFTW` FFTW3 library does not linked if 'yes' is set in this valuable.

`CMAKE_LIBRARY_PATH` CMake library search paths. This directory is used to search FFTW3 library.

`CMAKE_INCLUDE_PATH` CMake include search paths. This directory is used to search include file for FFTW3.

`CMAKE_DISABLE_FIND_PACKAGE_FFTW` FFTW3 library does not linked if 'yes' is set in this valuable.

`HDF5_INCLUDE_DIRS` Include file directories to compile with HDF5. This setting can be found by using hfd5 command `h5pfc -show`.

`HDF5_LIBRARY_DIRS` Location of HDF5 library. This setting can be found by using hfd5 command `h5pfc -show`.

`HDF5_LIBRARIES` Library lists for HDF5. This setting can be found by using hfd5 command `h5pfc -show`.

`CMAKE_DISABLE_FIND_PACKAGE_HDF5` HDF5 library does not linked if 'yes' is set in this valuable.

An example of using CMake on Mac OS X is the following:

```
% cd work
% h5pfc -show
mpif90 -I/home/matsui/local/include -L/home/matsui/local/lib
 /home/matsui/local/lib/libhdf5hl_fortran.a
 /home/matsui/local/lib/libhdf5_hl.a
```

```
 /home/matsui/local/lib/libhdf5_fortran.a
 /home/matsui/local/lib/libhdf5.a
  -L/home/matsui/local/lib -lmpi -lz -ldl -lm
% cmake .. -DCMAKE_LIBRARY_PATH='/home/matsui/local/lib' \
? -DCMAKE_INCLUDE_PATH='/home/matsui/local/include' \
? -DHDF5_INCLUDE_DIRS='/home/matsui/local/include' \
? -DHDF5_LIBRARY_DIRS='/home/matsui/local/lib' \
? -DHDF5_LIBRARIES='/home/matsui/local/lib/libhdf5hl_fortran.a \
? /home/matsui/local/lib/libhdf5_hl.a \
? /home/matsui/local/lib/libhdf5_fortran.a \
? /home/matsui/local/lib/libhdf5.a'
```

After configuration, compile and install are started by

```
% make
...
% make install
```

After running `make` command, execute files are built in `[CALYPSO_HOME]/work/bin` directory.

7 Simulation procedure

Calypso consists of programs shown in Table 1. Because the serial programs do not use MPI, they are simply invoked by

```
% [program]
```

Parallel programs must be invoked using MPI commands. On a Linux cluster using MPICH, parallel programs are invoked with

```
% mpirun -np [# of processes] [program]
```

This command will vary depending on the MPI implementation installed on the machine. Please consult with your sysadmin for details.

To perform simulations by Calypso, the following processes are required.

1. Generate grids and spherical harmonics indexing information by `gen_sph_grids`.
2. Make initial fields by `sph_initial_field` (if necessary).

Table 1: List of program and required control file name

Program	Control file name	Type
gen_sph_grids	control_sph_shell	Parallel
sph_mhd	control_MHD	Parallel
sph_initial_field	control_MHD	Parallel
sph_add_initial_field	control_MHD	Parallel
sph_snapshot	control_snapshot	Parallel
sph_zm_snapshot	control_snapshot	Parallel
sph_dynamobench	control_snapshot	Parallel
assemble_sph	control_sph_assemble	Parallel
t_ave_sph_mean_square	N/A	Serial
t_ave_picked_sph_coefs	N/A	Serial
t_ave_nusselt	N/A	Serial

3. Perform the simulation by `sph_mhd`.

4. Convert the parallel spectra data by `assemble_sph` to continue with changing number of processes (if necessary).

5. Data analysis by `sph_snapshot`, `sph_snapshot`, or `sph_dynamobench`.

6. Update initial fields by `sph_add_initial_field` for more simulations (if necessary).

7. Evaluate time averages by `t_ave_sph_mean_square`, `t_ave_picked_sph_coefs`, or `t_ave_nusselt` if necessary.

The simulation program `sph_mhd` requires an indexing file for spherical transform. `sph_mhd` generates spectrum data and monitoring data, and field data in Cartesian coordinate as outputs. The data transform programs (`sph_snapshot` and `sph_zm_snapshot`) generate outputs data from parallel spectra data. The flow of data is shown in Figure 2.

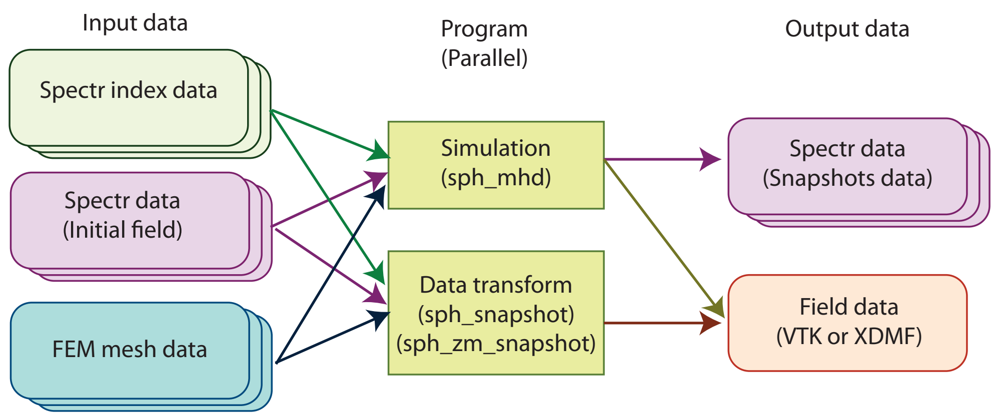


Figure 2: Data flow of the simulation. Simulations require index data for spherical harmonics transform, initial spectra (optional) data, and FEM mesh data. Simulation program also outputs spectra data, monitoring data and field data in Cartesian coordinate. Data transform program generates output data for simulation program from spectra data.

Each program needs one control file, the name of which is defined by the program. (Standard input is not supported by Fortran 90 so Calypso uses control files.) The appropriate control file names are shown in the Table 1. The following rules are used in the control files. An example of a control file is shown in Figure 3.

- Lines starting with '`#`' or '`!`' are treated as a comment lines and ignored.
- All control files consist of blocks which start with '`begin [name]`' and end with '`end [name]`'.
- The item name is shown first and the associated value/data is second.
- The order of items and blocks can be changed.
- If an item consists of multiple data, these should be listed in one line.
- If an item does not belong in the block it is ignored.
- An array block starts with '`begin array [name] [number of components]`' and ends with '`end array [name]`'.
- If `[number of components]` for an array is 0, '`end array [name]`' on the next line is not needed.

- In Fortran program, character '/' is recognized as an end of character valuable if text with '/' (*e.g.* file prefix including file paths) is not enclosed by ' or ".
- Calypso's control file input is limited to 255 characters for each line.

```
begin spherical_shell_ctl
!
  begin data_files_def
    num_subdomain_ctl    4
!
    sph_file_prefix             'sph_shell/in'
  end data_files_def
!
  begin num_grid_sph
    truncation_level_ctl     4
    ngrid_meridonal_ctl     12
    ngrid_zonal_ctl         24
!
    radial_grid_type_ctl   explicit
    array r_layer       4
      r_layer    1  0.5384615384615
      r_layer    2  0.5384615384615
      r_layer    3  1.038461538462
      r_layer    4  1.538461538462
    end array r_layer
!
  end num_grid_sph
end spherical_shell_ctl
```

Figure 3: Example of Control file

8 Examples

Several examples are provided in the `examples` directory. There are three subdirectories as examples. README files are also provided to perform these examples in each subdirectory.

`assemble_sph` Examples for assembling program of spectrum data. (see section 15)

`dynamo_benchmark` Examples for dynamo benchmark by Christensen *et. al.* (2001)

`spherical_shell` Examples for preprocessing program (see Section 9)

8.1 Examples for preprocessing program

Four examples illustrate the use of the preprocessing program. The examples include

`Chebyshev_points` Example to generate indexing data using Chebyshev collocation points

`equidistance` Example to generate indexing data with equi-distance grid

`explicitly_defined` Example to generate indexing data with explicitly defined radial points

`with_inner_core` Example to generate indexing data including inner core and external of the fluid shell.

8.2 Examples of dynamo benchmark

There are four examples for simulations using dynamo benchmark test as following.

`Case_0` Example of dynamo benchmark case 0 (Thermally driven convection without magnetic field)

`Case_1` Example of dynamo benchmark case 1 (Dynamo model with co-rotating and electrically insulated inner core)

`Case_2` Example of dynamo benchmark case 2 (Dynamo model with rotatable and conductive inner core)

`Compositional_case_1` Example of dynamo benchmark case 1 using compositional variation instead of temperature

The process of the simulation is as following:

1. Change to the directory for Benchmark Case 1 (for example)

```
[username]$ cd [CALYPSO_DIR]/examples/dynamo_benchmark/dynamobench_case1
```

2. Create the grid files for the simulation

```
[dynamobench_case_1]$ mpirun -np 4 [CALYPSO_DIR]/bin/gen_sph_grids
```

3. Run simulation program

```
[dynamobench_case_1]$ mpirun -np 4 [CALYPSO_DIR]/bin/sph_mhd
```

4. To continue the simulation, change the parameter `rst_ctl` in `control_MHD` from `dynamo_benchmark_1` to `start_from_rst_file` and continue simulation by repeating step 2.

5. To check the results for dynamo benchmark, run

```
[dynamobench_case_1]$ mpirun -np 4 [CALYPSO_DIR]/bin/sph_dynamobench
```

8.3 Example of data assembling program

An example for spectrum data assembling program is provided in `assemble_sph` directory.

8.4 Example of heat and compositional source

An example to perform a simulation with heat and compositional sources is given in `heat_composition_source` directory. To simplify the problem, only the thermal and compositional fields are evolved with no velocity (i.e. pure diffusion problem). A module to generate initial field data
`const_sph_initial_spectr` is copied to `src/programs/data_utilities/INITIAL_FIELD/` directory. The code must be recompiled after modifying this module. Initial field is generated by the program `sph_initial_field` after generating spherical harmonics information by `gen_sph_grid`. Aftrer the simulation, Y_0^0 component of temperature and composition as a function of radius and time is written in `picked_mode.dat`.

8.5 Example of thermal and compositional boundary conditions by external file

Heterogeneous boundary are input using an external file. An example to set thermal and compositional boundary conditions is given in `heterogineous_temp` directory. As in the heat source example, only the diffusion problem is solved in this example. In file `bc_spectr.btx`, temperature boundary conditions are defined for Y_0^0, Y_1^{1s}, Y_1^{1c}, and ,Y_2^{2c} component, and compositional boundary is defined for Y_0^0, Y_2^{2s}, and Y_2^{2c} components. The radial profile of these spherical harmonics coefficients are written in `picked_mode.dat`.

9 Preprocessing program (gen_sph_grid)

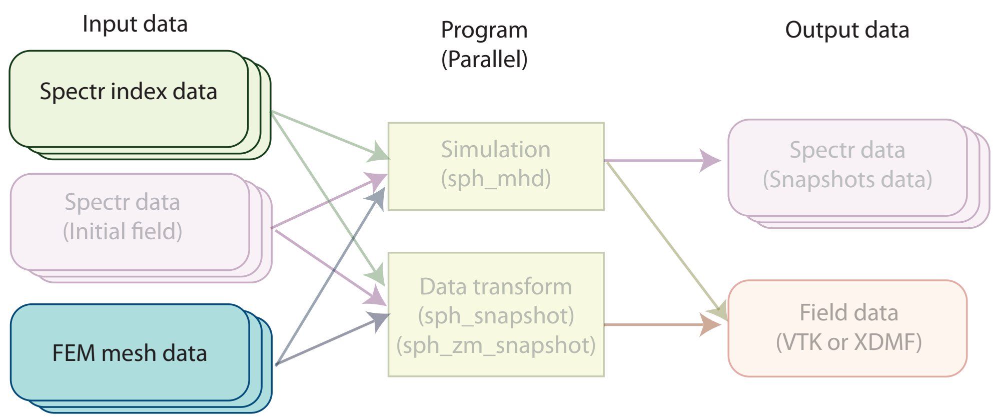


Figure 4: Generated files by preprocessing program in Data flow.

This program generates index table and a communication table for parallel spherical harmonics, table of integrals for Coriolis term, and FEM mesh information to generate visualization data (see Figure 4). This program needs control file for input. This program can perform with **any** number of MPI processes less than the number of subdomains. The output files include the indexing tables.

Table 2: List of files for gen_sph_grid

extension	Parallelization	I/O
control_sph_grid	Single	Input
[sph_prefix].[domain#].rj	Distributed	Output
[sph_prefix].[domain#].rlm	Distributed	Output
[sph_prefix].[domain#].rtm	Distributed	Output
[sph_prefix].[domain#].rtp	Distributed	Output
[sph_prefix].[domain#].gfm	Distributed	Output
radial_info.dat	Single	Output

9.1 Position of radial grid

The preprocessing program sets the radial grid spacing, either by a list in the control file or by setting an equidistant grid or Chebyshev collocation points.

In equidistance grid, radial grids are defined by

$$r(k) = r_i + (r_o - r_i) \frac{k - k_{ICB}}{N},$$

where, k_{ICB} is the grid points number at ICB. The radial grid set from the closest points of minimum radius defined by `[Min_radius_ctl]` in control file to the closest points of the maximum radius defined by `[Max_radius_ctl]` in control file, and radial grid number for the innermost points is set to $k = 1$.

In Chebyshev collocation points, radial grids in the fluid shell are defined by

$$r(k) = r_i + \frac{(r_o - r_i)}{2} \left[\frac{1}{2} - \cos\left(\pi \frac{k - k_{ICB}}{N} \right) \right],$$

For the inner core ($r < r_i$), grid points is defined by

$$r(k) = r_i - \frac{(r_o - r_i)}{2} \left[\frac{1}{2} - \cos\left(\pi \frac{k - k_{ICB}}{N} \right) \right],$$

and, grid points in the external of the shell ($r > r_o$) is defined by

$$r(k) = r_o + \frac{(r_o - r_i)}{2} \left[\frac{1}{2} - \cos\left(\pi \frac{k - k_{CMB}}{N} \right) \right],$$

where, k_{CMB} is the grid point number at CMB.

9.2 Control file (`control_sph_shell`)

Control file (`control_sph_shell`) consists the following items. Detailed description for each item can be checked by clicking "(Detail)" at the end of each item.
`spherical_shell_ctl`

- `data_files_def` (Detail)
  - `num_subdomain_ctl` `[Num_PE]` (Detail)
  - `sph_file_prefix` `[sph_prefix]` (Detail)

- num_domain_ctl (Detail)
  - num_radial_domain_ctl [Ndomain] (Detail)
  - num_horizontal_domain_ctl [Ndomain] (Detail)
  - array num_domain_sph_grid [Direction] [Ndomain] (Detail)
  - array num_domain_legendre [Direction] [Ndomain] (Detail)
  - array num_domain_spectr [Direction] [Ndomain] (Detail)
- num_grid_sph (Detail)
  - truncation_level_ctl [Lmax] (Detail)
  - ngrid_meridonal_ctl [Ntheta] (Detail)
  - ngrid_zonal_ctl [Nphi] (Detail)
  - radial_grid_type_ctl [explicit, Chebyshev, or equi_distance] (Detail)
  - num_fluid_grid_ctl [Nr_shell] (Detail)
  - fluid_core_size_ctl [Length] (Detail)
  - ICB_to_CMB_ratio_ctl [R_ratio] (Detail)
  - Min_radius_ctl [Rmin] (Detail)
  - Max_radius_ctl [Rmax] (Detail)

  - array r_layer [Layer #] [Radius] (Detail)
  - array boundaries_ctl [Boundary_name] [Layer #] (Detail)

If num_radial_domain_ctl and num_horizontal_domain_ctl are defined, the following arrays num_domain_sph_grid, num_domain_legendre, and num_domain_spectr are not necessary.

9.3 Spectrum index data

gen_sph_grid generates indexing table of the spherical transform. To perform spherical harmonics transform with distributed memory computers, data communication table is also included in these files. Calypso needs four indexing data for the spherical transform.

[sph_prefix].[domain#].rj Indexing table for spectrum data $f(r, l, m)$ to calculate linear terms. In program, spherical harmonics modes (l, m) is indexed by $j = l(l+1) + m$. The spectrum data are decomposed by spherical harmonics modes j. Data communication table for Legendre transform is included. The data also have the radial index of the ICB and CMB.

[sph_prefix].[domain#].rlm Indexing table for spectrum data $f(r, l, m)$ for Legendre transform. The spectrum data are decomposed by radial direction r and spherical harmonics order m. Data communication table to caricurate liner terms is included.

[sph_prefix].[domain#].rtm Indexing table for data $f(r, \theta, m)$ for Legendre transform. The data are decomposed by radial direction r and spherical harmonics order m. Data communication table for backward Fourier transform is included.

[sph_prefix].[domain#].rtp Indexing table for data $f(r, \theta, m)$ for Fourier transform and field data $f(r, \theta, \phi)$. The data are decomposed by radial direction r and meridional direction θ. Data communication table for forward Legendre transform is included.

9.4 Finite element mesh data

Calypso generates field data for visualization with XDMF or VTK format. To generate field data file, the preprocessing program generates FEM mesh data for each subdomain of spherical grid (r, θ, ϕ) under the Cartesian coordinate (x, y, z). The mesh data file is written as GeoFEM (http://geofem.tokyo.rist.or.jp) mesh data format, which consists of each subdomain mesh and communication table among overlapped nodes.

9.5 Radial grid data

The preprocessing program generates radius of each layer in radial_info.dat if radial_grid_type_ctl is set to Chebyshev or equi_distance. This file consists of blocks array r_layer and array boundaries_ctl for control file. This data may be useful if you want to modify radial grid spacing by yourself.

10 Simulation program (sph_mhd)

The name of the simulation program is sph_mhd. This program requires control_MHD as a Control file. This program performs with the indexing file for spherical harmonics and Coriolis term integration file generated by the preprocessing program gen_sph_grid. Data files for this program are listed in Table 3. Indexing data for spherical harmonics

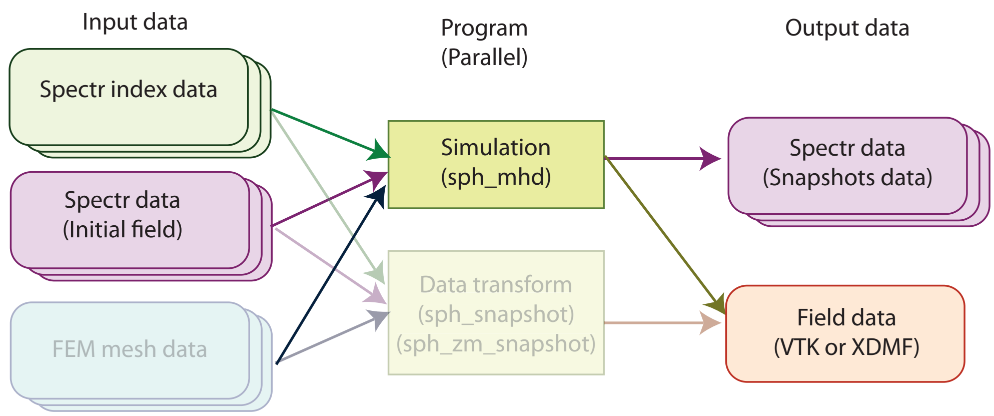


Figure 5: Data flow for the simulation program.

which starting with [sph_prefix] are obtained by the preprocessing program gen_sph_grid. The boundary condition data file [boundary_data_name] is optionally required if boundary conditions for temperature and composition are not homogenious.

Table 3: List of files for simulation sph_mhd

name	Parallelization	I/O
control_MHD	Serial	Input
[sph_prefix].[domain#].rj	Distributed	Input
[sph_prefix].[domain#].rlm	Distributed	Input
[sph_prefix].[domain#].rtm	Distributed	Input
[sph_prefix].[domain#].rtp	Distributed	Input
[sph_prefix].[domain#].gfm	Distributed	Input
[boundary_data_name]	Single	Input
[rst_prefix].[step#].[domain#].fst	Distributed	Input/Output
[vol_pwr_prefix].dat	Single	Output
[vol_pwr_prefix]_l.dat	Single	Output
[vol_pwr_prefix]_m.dat	Single	Output
[vol_pwr_prefix]_lm.dat	Single	Output
[vol_ave_prefix].dat	Single	Output
[layer_pwr_prefix]_l.dat	Single	Output
[layer_pwr_prefix]_m.dat	Single	Output
[layer_pwr_prefix]_lm.dat	Single	Output
[gauss_coef_prefix].dat	Single	Output
[picked_sph_prefix].dat	Single	Output
[nusselt_number_prefix].dat	Single	Output
[fld_prefix].[step#].[domain#].[extension]	-	Output
[section_prefix].[step#].[extension]	Single	Output
[isosurface_prefix].[step#].[extension]	Single	Output

10.1 Control file

The format of the control file `control_MHD` is described below. The detail of each block is described in section A. You can jump to detailed description by clicking "(Detail)".

`MHD_control` (Header of the control file)

- `data_files_def` (Detail)
  - `num_subdomain_ctl` `[Num_PE]` (Detail)
  - `num_smp_ctl` `[Num_Threads]` (Detail)
  - `sph_file_prefix` `[sph_prefix]` (Detail)
  - `boundary_data_file_name` `[boundary_data_name]` (Detail)
  - `restart_file_prefix` `[rst_prefix]` (Detail)
  - `field_file_prefix` `[fld_prefix]` (Detail)
  - `field_file_fmt_ctl` `[fld_format]` (Detail)
- `model`
  - `phys_values_ctl` (Detail)
    - `array nod_value_ctl` `[Field]` `[Viz_flag]` `[Monitor_flag]` (Detail)
  - `time_evolution_ctl` (Detail)
    - `array time_evo_ctl` `[Field]` (Detail)
  - `boundary_condition` (Detail)
    - `array bc_temperature` `[Group]` `[Type]` `[Value]` (Detail)
    - `array bc_velocity` `[Group]` `[Type]` `[Value]` (Detail)
    - `array bc_composition` `[Group]` `[Type]` `[Value]` (Detail)
    - `array bc_magnetic_field` `[Group]` `[Type]` `[Value]` (Detail)
  - `forces_define` (Detail)

  * array force_ctl [Force] (Detail)
- dimensionless_ctl (Detail)
  * array dimless_ctl [Name] [Value] (Detail)
- coefficients_ctl (Detail)
  * thermal (Detail)
    · array coef_4_termal_ctl [Name] [Power] (Detail)
    · array coef_4_t_diffuse_ctl [Name] [Power] (Detail)
    · array coef_4_heat_source_ctl [Name] [Power] (Detail)
  * momentum (Detail)
    · array coef_4_velocity_ctl [Name] [Power] (Detail)
    · array coef_4_press_ctl [Name] [Power] (Detail)
    · array coef_4_v_diffuse_ctl [Name] [Power] (Detail)
    · array coef_4_buoyancy_ctl [Name] [Power] (Detail)
    · array coef_4_Coriolis_ctl [Name] [Power] (Detail)
    · array coef_4_Lorentz_ctl [Name] [Power] (Detail)
    · array coef_4_composit_buoyancy_ctl [Name] [Power] (Detail)
  * induction (Detail)
    · array coef_4_magnetic_ctl [Name] [Power] (Detail)
    · array coef_4_m_diffuse_ctl [Name] [Power] (Detail)
    · array coef_4_induction_ctl [Name] [Power] (Detail)
  * composition (Detail)
    · array coef_4_composition_ctl [Name] [Power] (Detail)
    · array coef_4_c_diffuse_ctl [Name] [Power] (Detail)
    · array coef_4_composition_source_ctl [Name] [Power] (Detail)
- temperature_define (Detail)
  * ref_temp_ctl [REFERENCE_TEMP] (Detail)
  * low_temp_ctl (Detail)
    · depth [RADIUS] (Detail)
    · temperature [TEMPERATURE] (Detail)

- * high_temp_ctl (Detail)
    - · depth [RADIUS] (Detail)
    - · temperature [TEMPERATURE] (Detail)

- control
  - time_step_ctl (Detail)
    - * elapsed_time_ctl [ELAPSED_TIME] (Detail)
    - * i_step_init_ctl [ISTEP_START] (Detail)
    - * i_step_finish_ctl [ISTEP_FINISH] (Detail)
    - * i_step_check_ctl [ISTEP_MONITOR] (Detail)
    - * i_step_rst_ctl [ISTEP_RESTART] (Detail)
    - * i_step_field_ctl [ISTEP_FIELD] (Detail)
    - * dt_ctl [DELTA_TIME] (Detail)
    - * time_init_ctl [INITIAL_TIME] (Detail)
  - restart_file_ctl (Detail)
    - * rst_ctl [INITIAL_TYPE] (Detail)
  - time_loop_ctl (Detail)
    - * scheme_ctl [EVOLUTION_SCHEME] (Detail)
    - * coef_imp_v_ctl [COEF_INP_U] (Detail)
    - * coef_imp_t_ctl [COEF_INP_T] (Detail)
    - * coef_imp_b_ctl [COEF_INP_B] (Detail)
    - * coef_imp_c_ctl [COEF_INP_C] (Detail)
    - * FFT_library_ctl [FFT_Name] (Detail)
    - * Legendre_trans_loop_ctl [Leg_Loop] (Detail)

- sph_monitor_ctl (Detail)
  - volume_average_prefix [vol_ave_prefix] (Detail)
  - volume_pwr_spectr_prefix [vol_pwr_prefix] (Detail)
  - layered_pwr_spectr_prefix [layer_pwr_prefix] (Detail)

- picked_sph_prefix [picked_sph_prefix] (Detail)
- gauss_coefs_prefix [gauss_coef_prefix] (Detail)
- gauss_coefs_radius_ctl [gauss_coef_radius] (Detail)
- nusselt_number_prefix [nusselt_number_prefix] (Detail)
- array pick_layer_ctl [Layer #] (Detail)
- array pick_sph_spectr_ctl [Degree] [Order] (Detail)
- array pick_sph_degree_ctl [Degree] (Detail)
- array pick_sph_order_ctl [Order] (Detail)
- array pick_gauss_coefs_ctl [Degree] [Order] (Detail)
- array pick_gauss_coef_degree_ctl [Degree] (Detail)
- array pick_gauss_coef_order_ctl [Order] (Detail)
- nphi_mid_eq_ctl [Nphi_mid_equator] (Detail)

10.2 Spectrum data for restarting

Spectrum data is used for restarting data and generating field data by Data transform program sph_snapshot, sph_zm_snapshot, or sph_dynamobench. This file is saved for each subdomain (MPI processes), then [step #] and [domain #] are added in the file name. The [step #] is calculated by time step / [ISTEP_RESTART].

10.3 Thermal and compositional boundary condition data file

Thermal and compositional heterogeneity at boundaries are defined by a external file named [boundary_data_name]. In this file, temperature, composition, heat flux, or compositional flux at ICB or CMB can be defined by spherical harmonics coefficients. To use boundary conditions in [boundary_data_name], file name is defined by boundary_data_file_name column in control file, and boundary condition type [type] is set to fixed_file or fixed_flux_file in bc_temperature or bc_composition column. By setting fixed_file or fixed_flux_file in control file, boundary conditions are copied from the file [boundary_data_name].

An example of the boundary condition file is shown in Figure 6. As for the control file, a line starting from '#' or '!' is recognized as a comment line. In [boundary_data_name], boundary condition data is defined as following:

1. Number of total boundary conditions to be defined in this file.
2. Field name to define the first boundary condition
3. Place to define the first boundary condition (ICB or CMB)
4. Number of spherical harmonics modes for each boundary condition
5. Spectrum data for the boundary conditions (degree l, order m, and harmonics coefficients)
6. After finishing the list of spectrum data return to Step 2 for the next boundary condition

If harmonics coefficients of the boundary conditions are not listed in item 5, 0.0 is automatically applied for the harmonics coefficients of the boundary conditions. So, only non-zero components need to be listed in the boundary condition file.

10.4 Field data for visualization

Field data is used for the visualization processes. Field data are written with XDMF format (http://www.xdmf.org/index.php/Main_Page), merged VTK, or distributed VTK format (http://www.vtk.org/VTK/img/file-formats.pdf). The output data format is defined by fld_format. Visualization applications which we checked are listed in Table 4. Because the field data is written by using Cartesian coordinate (x, y, z) system, coordinate conversion is required to plot vector field in spherical coordinate (r, θ, ϕ) or cylindrical coordinate (s, ϕ, z). We will introduce a example of visualization process using ParaView in Section 18.

10.4.1 Distributed VTK data

Distributed VTK data have the following advantage and disadvantages to use:

- Advantage
    - Faster output

```
#
#  number of boundary conditions
      4
#
#   boundary condition data list
#
#    Fixed temperature at ICB
temperature
ICB
    3
  0  0   1.0E+00
  1  1   2.0E-01
  2  2   3.0E-01
#
#    Fixed heat flux at CMB
heat_flux
CMB
   2
  0  0    -0.9E+0
  1  -1    5.0E-1
#
#    Fixed composition flux at ICB
composite_flux
ICB
   2
  0  0    0.0E+00
  2  0   -2.5E-01
#
#    Fixed composition at CMB
composition
CMB
   2
  0   0   1.0E+00
  2  -2   5.0E-01
```

Figure 6: An example of boundary condition file.

Table 4: Checked visualization application

Format	Application
Distributed VTK	ParaView (`http://www.paraview.org`)
Merged VTK	ParaView, VisIt (`https://wci.llnl.gov/codes/visit/`) Mayavi (`http://mayavi.sourceforge.net/`)
XDMF	ParaView, VisIt

  - No external library is required

- Disadvantage
  - Many data files are generated
  - Total data file size is large
  - Only ParaView supports this format

Distributed VTK data consist files listed in Table 5. For ParaView, all subdomain data is read by choosing `[fld_prefix].[step#].pvtk` in file menu.

Table 5: List of written files for distributed VTK format

name	
`[fld_prefix].[step#].[domain#].vtk`	VTK data for each subdomain
`[fld_prefix].[step#].pvtk`	Subdomain file list for Paraview

10.4.2 Merged VTK data

Merged VTK data have the following advantage and disadvantages to use:

- Advantage
  - Merged field data is generated
  - No external library is required
  - Many applications support VTK format

- Disadvantage
    - Very slow to output
    - Total data file size is large

Merged VTK data generate files listed in Table 6.

Table 6: List of written files for merged VTK format

name	
`[fld_prefix].[step#].vtk`	Merged VTK data

10.4.3 Merged XDMF data

Merged XDMF data have the following advantage and disadvantages to use:

- Advantage
    - Fastest output
    - Merged field data is generated
    - File size is smaller than the VTK formats
- Disadvantage
    - Parallel HDF5 library should be required to use

Merged XDMF data generate files listed in Table 7. For ParaView, all subdomain data is read by choosing `[fld_prefix].solution.xdmf` in file menu.

Table 7: List of written files for XDMF format

name	
`[fld_prefix].mesh.h5`	HDF5 file for geometry data
`[fld_prefix].[step#].h5`	HDF5 file for field data
`[fld_prefix].solution.xdmf`	HDF5 file lists to be read

10.5 Mean square amplitude data

This program output mean square amplitude of the fields which is marked as `Monitor_ON` over the fluid shell at every `[increment_monitor]` steps. For vector fields, For the velocity $\boldsymbol{u}$ and magnetic field $\boldsymbol{B}$, the kinetic energy $1/2u^2$ and magnetic energy $1/2B^2$ are calculated instead of mean square amplitude. Labels on the first lines indicate following data. The data file have the following headers in the first 7 lines, and headers of the data and data are stored in the following lines. The header in the first 7 lines is the following. If these mean square amplitude data files exist before starting the simulation, programs append results at the end of files without checking constancy of the number of data and order of the field. If you change the configuration of data output structure, please move the existed data files to another directory before starting the programs.

`line 2:` Number of radial grid and truncation level

`line 4:` radial layer ID for ICB and CMB

`line 6:` Number of field of data, total number of components

`line 7:` Number of components for each field

Labels for data indicates as

`t_step` Time setp number

`time` Time

`K_ene_pol` Amplitude of poloidal kinetic energy

`K_ene_tor` Amplitude of toroidal kinetic energy

`K_ene` Amplitude of total kinetic energy

`M_ene_pol` Amplitude of poloidal magnetic energy

`M_ene_tor` Amplitude of toroidal magnetic energy

`M_ene` Amplitude of total magnetic energy

`[Field]_pol` Mean square amplitude of poloidal component of `[Field]`

`[Field]_tor` Mean square amplitude of toroidal component of `[Field]`

`[Field]` Mean square amplitude of `[Field]`

10.5.1 Volume average data

Volume average data are written by defining volume_average_prefix in control file. Volume average data are written in [vol_ave_prefix].dat with same format as RMS amplitude data. If you need the sphere average data for specific radial point, you can use picked spectrum data for $l = m = 0$ at specific radius.

10.5.2 Volume spectrum data

Volume spectrum data are written by defining volume_pwr_spectr_prefix in control file. By defining volume_pwr_spectr_prefix, following spectrum data averaged over the fluid shell is written. Data format is the same as the volume mean square data, but degree l, order m, or meridional wave number $l - m$ is added in the list of data.

[vol_pwr_prefix_l.dat Volume average of mean square amplitude of the fields as a function of spherical harmonic degree l. For scalar field, the spectrum is

$$f_{sq}(l) = \frac{1}{V} \sum_{m=-l}^{m=l} \int (f_l^m)^2 \, dV.$$

For vector field, spectrum for the poloidal and toroidal components are written by

$$B_{Ssq}(l) = \frac{1}{V} \sum_{m=-l}^{m=l} \int (\boldsymbol{B}_{Sl}^m)^2 \, dV,$$
$$B_{Tsq}(l) = \frac{1}{V} \sum_{m=-l}^{m=l} \int (\boldsymbol{B}_{Tl}^m)^2 \, dV.$$

If the vector field $\boldsymbol{F}$ is not solenoidal (i.e. $\nabla \cdot \boldsymbol{F} \neq 0$), The poloidal component of mean square data are included mean square field of the potential components as

$$F_{Ssq}(l) = \frac{1}{V} \sum_{m=-l}^{m=l} \int \left[(\boldsymbol{B}_{Sl}^m)^2 + (-\nabla \phi_{Fl}^m)^2 \right] dV.$$

[vol_pwr_prefix]_m.dat Volume average of mean square amplitude of the fields as a function of spherical harmonic order m. The zonal wave number is referred in this spectrum data. For scalar field, the spectrum is

$$f_{sq}(m) = \frac{1}{V} \sum_{l=0}^{l=m} \int \left[(f_l^m)^2 + \left(f_l^{-m}\right)^2 \right] dV.$$

For vector field, spectrum for the poloidal and toroidal components are written by

$$
\begin{aligned}
B_{Ssq}(m) &= \frac{1}{V}\sum_{l=0}^{l=m}\int\left[\left(\boldsymbol{B}_{Sl}^{m}\right)^2+\left(\boldsymbol{B}_{Sl}^{-m}\right)^2\right]dV,\\
B_{Tsq}(m) &= \frac{1}{V}\sum_{l=0}^{l=m}\int\left[\left(\boldsymbol{B}_{Tl}^{m}\right)^2+\left(\boldsymbol{B}_{Tl}^{-m}\right)^2\right]dV.
\end{aligned}
$$

`[vol_pwr_prefix]_lm.dat` Volume average of mean square amplitude of the fields as a function of spherical harmonic order $n = l - m$. The wave number in the latitude direction is referred in this spectrum data. For scalar field, the spectrum is

$$
f_{sq}(n) = \frac{1}{V}\sum_{l=n}^{l=l-n}\int\left[\left(f_l^{l-n}\right)^2+\left(f_l^{-l+n}\right)^2\right]dV.
$$

For vector field, spectrum for the poloidal and toroidal components are written by

$$
\begin{aligned}
B_{Ssq}(n) &= \frac{1}{V}\sum_{l=n}^{l=l-n}\int\left[\left(\boldsymbol{B}_{Sl}^{l-n}\right)^2+\left(\boldsymbol{B}_{Sl}^{-l+n}\right)^2\right]dV,\\
B_{Tsq}(n) &= \frac{1}{V}\sum_{l=n}^{l=l-n}\int\left[\left(\boldsymbol{B}_{Tl}^{l-n}\right)^2+\left(\boldsymbol{B}_{Tl}^{-l+n}\right)^2\right]dV.
\end{aligned}
$$

10.5.3 layered spectrum data

Spectrum data for the each radial position are written by defining `volume_pwr_spectr_prefix` in control file. By defining `layered_pwr_spectr_prefix`, following spectrum data averaged over the fluid shell is written. Data format is the same as the volume spectrum data, but radial grid point and radius of the layer is added in the list. The following files are generated.

`[layer_pwr_prefix]_l.dat` Surface average of mean square amplitude of the fields as a function of spherical harmonic degree l and radial grid id k. For scalar field, the spectrum is

$$
f_{sq}(k,l) = \frac{1}{S}\sum_{m=-l}^{m=l}\int\left(f_l^m\right)^2 dS.
$$

For vector field, spectrum for the poloidal and toroidal components are written by

$$
\begin{aligned}
B_{Ssq}(k,l) &= \frac{1}{S}\sum_{m=-l}^{m=l}\int (\boldsymbol{B}_{Sl}^{m})^2\, dS, \\
B_{Tsq}(k,l) &= \frac{1}{S}\sum_{m=-l}^{m=l}\int (\boldsymbol{B}_{Tl}^{m})^2\, dS.
\end{aligned}
$$

`[layer_pwr_prefix]_m.dat` Surace average of mean square amplitude of the fields as a function of spherical harmonic order m and radial grid id k. The zonal wave number is referred in this spectrum data. For scalar field, the spectrum is

$$
f_{sq}(k,m) = \frac{1}{S}\sum_{l=m}^{l=L}\int \left[(f_l^m)^2 + \left(f_l^{-m}\right)^2\right] dS.
$$

For vector field, spectrum for the poloidal and toroidal components are written by

$$
\begin{aligned}
B_{Ssq}(k,m) &= \frac{1}{S}\sum_{l=m}^{l=L}\int \left[(\boldsymbol{B}_{Sl}^{m})^2 + \left(\boldsymbol{B}_{Sl}^{-m}\right)^2\right] dS, \\
B_{Tsq}(k,m) &= \frac{1}{S}\sum_{l=m}^{l=L}\int \left[(\boldsymbol{B}_{Tl}^{m})^2 + \left(\boldsymbol{B}_{Tl}^{-m}\right)^2\right] dS.
\end{aligned}
$$

`[layer_pwr_prefix]_lm.dat` Surface average of mean square amplitude of the fields as a function of spherical harmonic order $n = l - m$ and radial grid id k. The wave number in the latitude direction is referred in this spectrum data. For scalar field, the spectrum is

$$
f_{sq}(k,n) = \frac{1}{S}\sum_{l=n}^{l=L}\int \left[\left(f_l^{l-n}\right)^2 + \left(f_l^{-l+n}\right)^2\right] dS.
$$

For vector field, spectrum for the poloidal and toroidal components are written by

$$
\begin{aligned}
B_{Ssq}(k,n) &= \frac{1}{S}\sum_{l=n}^{l=L}\int \left[\left(\boldsymbol{B}_{Sl}^{l-n}\right)^2 + \left(\boldsymbol{B}_{Sl}^{-l+n}\right)^2\right] dS, \\
B_{Tsq}(k,n) &= \frac{1}{S}\sum_{l=n}^{l=L}\int \left[\left(\boldsymbol{B}_{Tl}^{l-n}\right)^2 + \left(\boldsymbol{B}_{Tl}^{-l+n}\right)^2\right] dS.
\end{aligned}
$$

10.6 Gauss coefficient data [gauss_coef_prefix].dat

This program output selected Gauss coefficients of the magnetic field. Gauss coefficients is evaluated for radius defined by [gauss_coef_radius] every [increment_monitor] steps. Gauss coefficients are evaluated by using poloidal magnetic field at CMB $B_{Sl}^{m}(r_o)$ and radius defined by [gauss_coef_radius] r_e as

$$
\begin{aligned}
g_l^m &= \frac{l}{r_e^2}\left(\frac{r_o}{r_e}\right)^l B_{Sl}^{m}(r_o), \\
h_l^m &= \frac{l}{r_e^2}\left(\frac{r_o}{r_e}\right)^l B_{Sl}^{-m}(r_o).
\end{aligned}
$$

The data file has the following headers in the first three lines,

`line 2:` Number of saved Gauss coefficients and reference radius.

`line 3:` Labels of Gauss coefficients data.

The data consists of time step, time, and Gauss coefficients for each step in one line. If the Gauss coefficients data file exist before starting the simulation, programs append Gauss coefficients at the end of files without checking constancy of the number of data and order of the field. If you change the configuration of data output structure, please move the old Gauss coefficients file to another directory before starting the programs.

10.7 Spectrum monitor data [picked_sph_prefix].dat

This program outputs spherical harmonics coefficients at specified spherical harmonics modes and radial points in single text file. Spectrum data marked [Monitor_On] are written in our line for each spherical harmonics mode and radial point every [increment_monitor] steps. If the spectrum monitor data file exist before starting the simulation, programs append spectrum data at the end of files without checking constancy of the number of data and order of the field. If you change the configuration of data output structure, please move the old spectrum monitor file to another directory before starting the programs.

If a vector field $\boldsymbol{F}$ is not a solenoidal field, $\boldsymbol{F}$ is described by the spherical harmonics coefficients of the poloidal F_{Sl}^{m}, toroidal F_{Tl}^{m}, and potential φ_l^m components as

$$
\boldsymbol{F}(r,\theta,\phi) = -\frac{1}{r^2}\frac{\partial \varphi_0^0}{\partial r}\hat{r} + \sum_{l=1}^{L}\sum_{m=-l}^{l}\left[\nabla\times\nabla\times\left(F_{Sl}^{m}\hat{r}\right) + \nabla\times\left(F_{Tl}^{m}\right) - \nabla\left(\varphi_l^m Y_l^m\right)\right].
$$

In Calypso, the following coefficients are written for the non-solenoidal vector.

$$\texttt{[field\_name]\_pol} : \quad \begin{cases} F_{Sl}^{m} - \dfrac{r^2}{l\,(l+1)}\dfrac{\partial \varphi_l^m}{\partial r} & \text{for} \quad (l \neq 0) \\ -r^2 \dfrac{\partial \varphi_0^0}{\partial r} & \text{for} \quad (l = 0) \end{cases}$$

$$\texttt{[field\_name]\_dpdr} : \quad \begin{cases} \dfrac{\partial F_{Sl}^{m}}{\partial r} - \varphi_l^m & \text{for} \quad (l \neq 0) \\ 0 & \text{for} \quad (l = 0) \end{cases}$$

$$\texttt{[field\_name]\_tor} : \quad F_{Tl}^{m}$$

10.8 Nusselt number data `[nusselt_number_prefix].dat`

CAUTION: Nusselt number is not evaluated if heat source is exsist. The Nusselt number Nu at CMB and ICB is written for each step in one line. The Nusselt number is evaluated by

$$Nu = \frac{< \partial T / \partial r >}{\partial T_{diff} / \partial r},$$

where, $< \partial T / \partial r >$ and T_{diff} are the horizontal average of the temperature gradient at ICB and CMB and diffusive temperature profile, respectively. T_{diff} is evaluated without heat source, as

$$T_{diff} = \frac{r_o T_o - r_i T_i}{r_o - r_i} + \frac{r_o r_i \left(T_i - T_o\right)}{r_o - r_i} \frac{1}{r}.$$

This diffusive temperature profile is for the case without heat source in the fluid. If simulation is performed including the heat source, this data file does not written. If the Nusselt number data file exist before starting the simulation, programs append spectrum data at the end of files without checking constancy. If you change the configuration of data output structure, please move the old spectrum monitor file to another directory before starting the programs.

11 Data transform program (sph_snapshot and sph_zm_snapshot)

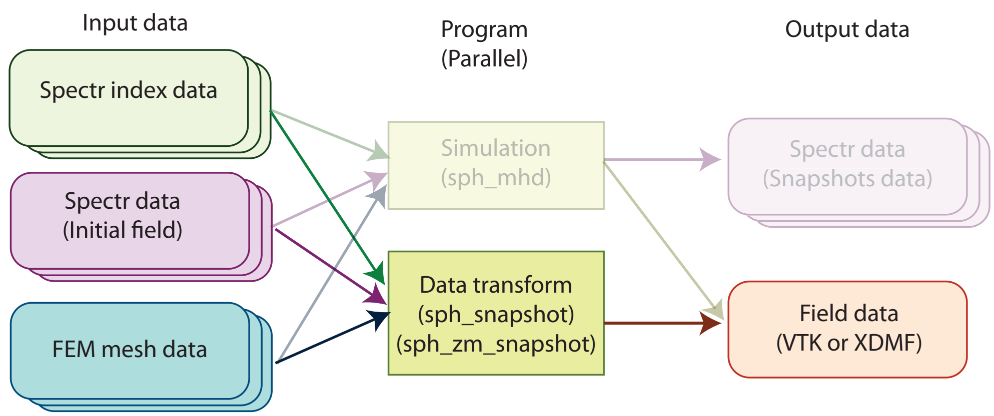


Figure 7: Data flow for data transform program.

Simulation program outputs spectrum data as a whole field data. This program generates field data from spectrum data for visualization. This program also can pick Gauss coefficients, mean square data over sphere or each surface from spectrum data as the simulation program.

This program requires control file `control_snapshot`. File format of the control file is same as the control field for simulation `control_MHD`.

The same files as the simulation program are read in this program, and field data are generated from the snapshots of spectrum data. The monitoring data for snapshots can also be generated. `[step #]` is added in the file name, and the `[step #]` is calculated by `time step` / `[ISTEP_FIELD]`.

Table 8: List of files for simulation `sph_snap` and `sph_zm_snap`

name	Parallelization	I/O
`control_snapshot`	Serial	Input
`[sph_prefix].[domain#].rj`	Distributed	Input
`[sph_prefix].[domain#].rlm`	Distributed	Input
`[sph_prefix].[domain#].rtm`	Distributed	Input
`[sph_prefix].[domain#].rtp`	Distributed	Input
`[sph_prefix].[domain#].gfm`	Distributed	Input
`[boundary_data_name]`	Single	Input
`[rst_prefix].[step#].[domain#].fst`	Distributed	Input
`[vol_pwr_prefix].dat`	Single	Output
`[vol_pwr_prefix]_l.dat`	Single	Output
`[vol_pwr_prefix]_m.dat`	Single	Output
`[vol_pwr_prefix]_lm.dat`	Single	Output
`[vol_ave_prefix].dat`	Single	Output
`[layer_pwr_prefix]_l.dat`	Single	Output
`[layer_pwr_prefix]_m.dat`	Single	Output
`[layer_pwr_prefix]_lm.dat`	Single	Output
`[gauss_coef_prefix].dat`	Single	Output
`[picked_sph_prefix].dat`	Single	Output
`[nusselt_number_prefix].dat`	Single	Output
`[fld_prefix].[step#].[domain#].[extension]`	-	Output

12 Initial field generation program (sph_initial_field)

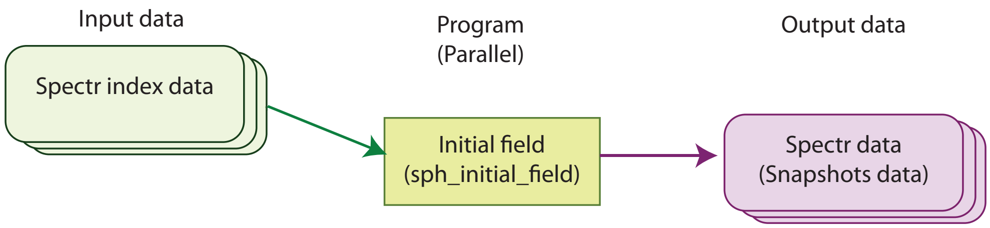


Figure 8: Data flow for initial field generation program.

The initial fields for dynamo benchmark can set in the simulation program by setting [INITIAL_TYPE] flag. This program is used to generate initial field by user. The heat source q_T and light element source q_C are also defined by this program because q_T and q_C are defined as scalar fields. The Fortran source file to define initial field const_sph_initial_spectr.f90 is saved in src/programs/data_utilities /INITIAL_FIELD/ directory, and please compile again after modifying this module. This program also needs the files listed in Table 9. This program generates the spectrum

Table 9: List of files for simulation sph_initial_field

name	Parallelization	I/O
control_MHD	Serial	Input
[sph_prefix].[domain#].rj	Distributed	Input
[sph_prefix].[domain#].rlm	Distributed	Input
[sph_prefix].[domain#].rtm	Distributed	Input
[sph_prefix].[domain#].rtp	Distributed	Input
[rst_prefix].0.[domain#].fst	Distributed	Input/Output

data files [rst_prefix].0.[domain#].fst. To use generated initial data file, please set [ISTEP_START] to be 0 and [INITIAL_TYPE] to be start_from_rst_file.

12.1 Definition of the initial field

To construct Initial field data, you need to edit the source code const_sph_initial_spectr.f90 in src/programs/data_utilities/INITIAL_FIELD/ directory. The module const_sph_initial_spectr consists of the following subroutines:

sph_initial_spectrum: Top subroutine to construct initial field.

set_initial_velocity: Routine to construct initial velocity.

set_initial_temperature: Routine to construct initial temperature.

set_initial_composition: Routine to construct initial composition.

set_initial_magne_sph: Routine to construct initial magnetic field.

set_initial_heat_source_sph: Routine to construct heat source.

set_initial_light_source_sph: Routine to construct composition source.

The construction routine for each field are called from the top routine const_sph_initial_spectr.f90. If lines to call subroutines are commented out, corresponding initial fields are set to 0. In addition, the initial fields to be constructed need to be defined by nod_value_ctl array in the control_MHD.

Table 10: Field name and corresponding field id in Calypso

field name	scalar	poloidal	toroidal
Velocity	-	ipol%i_velo	itor%i_velo
Magnetic field	-	ipol%i_magne	itor%i_magne
Current density	-	ipol%i_current	itor%i_current
Temperature	ipol%i_temp	-	-
Composition	ipol%i_light	-	-
Heat source	ipol%i_heat_source	-	-
Composition source	ipol%i_light_source	-	-

Initial fields need to be defined by the spherical harmonics coefficients at each radial points as array `d_rj(i,i_field)`, where `i` and `i_field` are the local address of the spectrum data and field id, respectively. The address of the fields are listed in Table 10.

In Calypso, local data address for each MPI process is used for the spectrum data address `i`. To find the local address `i`, two functions are required.
First, `j = find_local_sph_mode_address(l,m)` returns the local spherical harmonics address `j` from aa spherical harmonics mode Y_l^m. If process does not have the data for Y_l^m, `j` is set to 0. Second, `i = local_sph_data_address(k,j)` returns the local data address `i` from radial grid number `k` and local spherical harmonics id `j`. For do loops in the radial direction, the total number of radial grid points, radial address for ICB, and radial address for CMB are defined as `nidx_rj(1)`, `nlayer_ICB`, and `nlayer_CMB`, respectively. The radius for the `k`-th grid points can be obtained by `r = radius_1d_rj_r(k)`. The subroutines to define initial temperature for the dynamo benchmark Case 1 is shown below as an example.

```
!
      subroutine set_initial_temperature
!
      use m_sph_spectr_data
!
      integer ( kind = kint) :: inod, k, jj
      real (kind = kreal) :: pi, rr, xr, shell
      real(kind = kreal), parameter :: A_temp = 0.1d0
!
!
!$omp parallel do
      do inod = 1, nnod_rj
        d_rj(inod,ipol%i_temp) = zero
      end do
!$omp end parallel do
!
      pi = four * atan(one)
      shell = r_CMB - r_ICB
!
!   search address for (l = m = 0)
      jj = find_local_sph_mode_address(0, 0)
!
!   set reference temperature if (l = m = 0) mode is there
      if (jj .gt. 0) then
```

```
          do k = 1, nlayer_ICB-1
            inod = local_sph_data_address(k,jj)
            d_rj(inod,ipol%i_temp) = 1.0d0
          end do
          do k = nlayer_ICB, nlayer_CMB
            inod = local_sph_data_address(k,jj)
            d_rj(inod,ipol%i_temp) = (ar_1d_rj(k,1) * 20.d0/13.0d0        &
     &                               - 1.0d0 ) * 7.0d0 / 13.0d0
          end do
        end if
!
!
!    Find local addrtess for (l,m) = (4,4)
        jj =  find_local_sph_mode_address(4, 4)
!         jj =  find_local_sph_mode_address(5, 5)
!
!    If data for (l,m) = (4,4) is there, set initial temperature
        if (jj .gt. 0) then
!    Set initial field from ICB to CMB
          do k = nlayer_ICB, nlayer_CMB
!
!    Set radius data
            rr = radius_1d_rj_r(k)
!    Set 1d address to substitute at (Nr, j)
            inod = local_sph_data_address(k,jj)
!
!    set initial temperature
            xr = two * rr - one * (r_CMB+r_ICB) / shell
            d_rj(inod,ipol%i_temp) = (one-three*xr**2+three*xr**4-xr**6)  &
     &                            * A_temp * three / (sqrt(two*pi))
          end do
        end if
!
!    Center
        if(inod_rj_center .gt. 0) then
          jj = find_local_sph_mode_address(0, 0)
          inod = local_sph_data_address(1,jj)
          d_rj(inod_rj_center,ipol%i_temp) = d_rj(inod,ipol%i_temp)
```

```
      end if
!
      end subroutine set_initial_temperature
!
```

13 Initial field modification program (sph_add_initial_field)

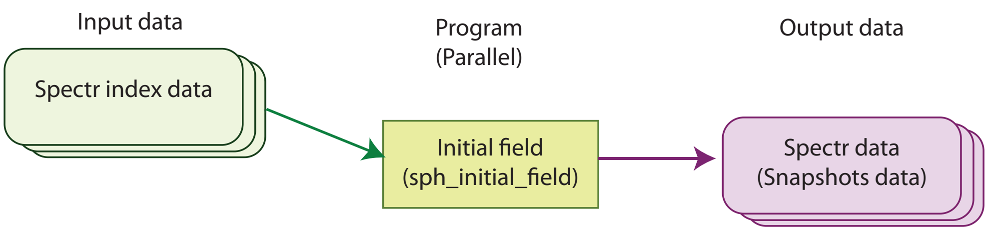


Figure 9: Data flow for initial field modification program.

Caution: This program overwrites existing initial field data. Please run it after taking a backup.

This program modifies or adds new data to an initial field file. It could be used to start a new geodynamo simulation by adding seed magnetic field or source terms to a non-magnetic convection simulation. The initial fields to be added are also defined in `const_sph_initial_spectr.f90`. `data_utilities/INITIAL_FIELD/` directory. This program also needs the files listed in Table 11. This program generates the spectrum data files `[rst_prefix].[step#].[domain#].fst`. To use generated initial data file, set `[ISTEP_START]` and `[ISTEP_RESTART]` to be appropriate time step and increment, respectively. To read the original initial field data, `[INITIAL_TYPE]` is set to be `start_from_rst_file` in `control_MHD`. In other words, the `[step #]` in the file name, `[ISTEP_START]`, and `[ISTEP_RESTART]` in the control file should be the consistent.

Table 11: List of files for simulation sph_add_initial_field

name	Parallelization	I/O
control_MHD	Serial	Input
[sph_prefix].[domain#].rj	Distributed	Input
[sph_prefix].[domain#].rlm	Distributed	Input
[sph_prefix].[domain#].rtm	Distributed	Input
[sph_prefix].[domain#].rtp	Distributed	Input
[rst_prefix].[step #].[domain#].fst	Distributed	Input/Output

14 Check program for dynamo benchmark (sph_dynamobench)

This program is only used to check solution for dynamo benchmark by Christensen *et. al*. The following files are used for this program.

Table 12: List of files for dynamo benchmark check sph_dynamobench

name	Parallelization	I/O
control_snapshot	Serial	Input
[sph_prefix].[domain#].rj	Distributed	Input
[sph_prefix].[domain#].rlm	Distributed	Input
[sph_prefix].[domain#].rtm	Distributed	Input
[sph_prefix].[domain#].rtp	Distributed	Input
[rst_prefix].[step#].[domain#].fst	Distributed	Input
dynamobench.dat	Single	Output

14.1 Dynamo benchmark data dynamobench.dat

In benchmark test by Christensen *et. al.*, both global values and local values are checked. As global results, Kinetic energy $\frac{1}{V}\int\frac{1}{2}u^2dV$ in the fluid shell, magnetic energy in the fluid shell $\frac{1}{V}\frac{1}{EPm}\int\frac{1}{2}B^2dV$ (for case 1 and 2), and magnetic energy in the solid inner sphere $\frac{1}{V_i}\frac{1}{EPm}\int\frac{1}{2}B^2 dV_i$ (for case 2 only). Benchmark also requests By increasing number of grid point at mid-dpeth of the fluid shell in the equatorial plane by `nphi_mid_eq_ctl`, program can find accurate solution for the point where $u_r = 0$ and $\partial u_r/\partial\phi > 0$. Angular frequency of the field pattern with respect to the ϕ direction is also required. The benchmark test also requires temperature and θ component of velocity. In the text file `dynamobench.dat`, the following data are written in one line for every `[i_step_rst_ctl]` step.

`t_step:` Time step number

`time:` Time

`KE_pol:` Poloidal kinetic energy

`KE_tor:` Toroidal kinetic energy

`KE_total:` Total kinetic energy

`ME_pol:` Poloidal magnetic energy (Case 1 and 2)

`ME_tor:` Toroidal magnetic energy (Case 1 and 2)

`ME_total:` Total magnetic energy (Case 1 and 2)

`ME_pol_ic:` Poloidal magnetic energy in inner core (Case 2)

`ME_tor_icore:` Toroidal magnetic energy in inner core (Case 2)

`ME_total_icore:` Total magnetic energy in inner core (Case 2)

`omega_ic_z:` Angular velocity of inner core rotation (Case 2)

`MAG_torque_ic_z:` Magnetic torque integrated over the inner core (Case 2)

`phi_1...4:` Longitude where $u_r = 0$ and $\partial u_r/\partial\phi > 0$ at mid-depth in equatorial plane.

`omega_vp44:` Drift frequency evaluated by V_{S4}^4 component

`omega_vt54:` Drift frequency evaluated by V_{T5}^4 component

`B_theta:` Θ component of magnetic field at requested point.

`v_phi:` ϕ component of velocity at requested point.

`temp:` Temperature at requested point.

```
t_step    time    KE_pol    KE_tor    KE_total    ME_pol    ME_t
or    ME_total    ME_pol_icore    ME_tor_icore    ME_total_icore
    omega_ic_z    MAG_torque_ic_z    phi_1    phi_2    phi_3
phi_4    omega_vp44    omega_vt54    B_theta    v_phi    temp
     20000   9.999999999998981E-001   1.534059732073072E+001   2
.431439471284618E+001   3.965499203357688E+001   2.405694011955O
09E+000   1.648662987055900E+000   4.054356999010911E+000   3.90
8687924452961E+001   4.812865754441352E-001   3.956816581997376E
+001   5.220517005592486E+000  -2.321885847438682E+002   3.59417
5626663308E-001   1.930213889461227E+000   3.501010216256124E+00
0   5.071806543051021E+000   7.808553595635292E-001  -1.64958344
1437563E-001  -5.136522824340612E+000  -8.047915942925034E+000
 3.752181234262930E-001
...
```

15 Data assemble program (`assemble_sph`)

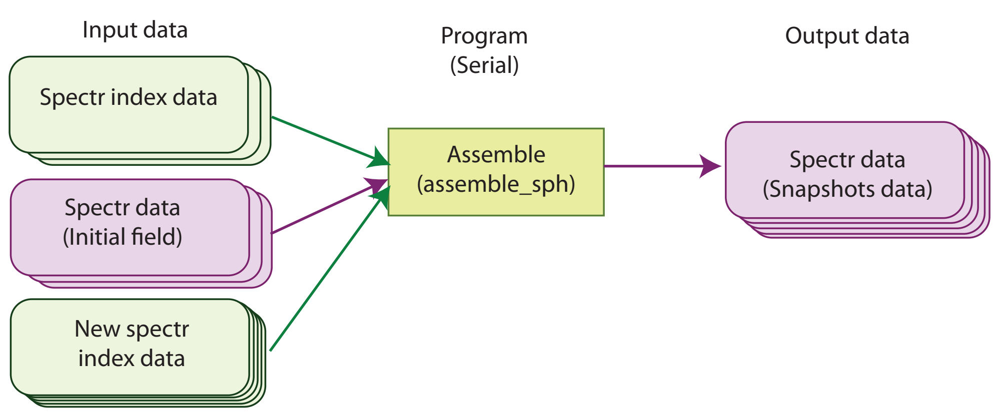


Figure 10: Data flow for spectrum data assemble program

Calypso uses distributed data files for simulations. This program is to generate new spectrum data for restarting with different spatial resolution or parallel configuration. This program organizes new spectral data by using specter indexing data using different domain decomposition. The following files used for data IO. If radial resolution is changed from the original data, the program makes new spectrum data by linear interpolation. If new data have smaller or larger truncation degree, the program fills zero to the new spectrum data or truncates the data to fit the new spatial resolution, respectively. This program can perform with **any** number of MPI processes. Data files for the program are shown In Table 13, and definition of `control_assemble_sph` is

Table 13: List of files for `assemble_sph`

extension	Distributed?	I/O
`control_sph_assemble`	Serial	Input
`[sph_prefix].[domain#].rj`	Distributed	Input
`[new_sph_prefix].[domain#].rj`	Distributed	Input
`[rst_prefix].[step#].[domain#].fst`	Distributed	Input
`[new_rst_prefix].[step#].[domain#].fst`	Distributed	Output

15.1 Format of control file

Control file consists the following groups.
`assemble_control`

- `data_files_def` (Detail)
  - `num_subdomain_ctl` `[Num_PE]` (Detail)
  - `sph_file_prefix` `[sph_prefix]` (Detail)
  - `restart_file_prefix` `[rst_prefix]` (Detail)
- `new_data_files_def` (Detail)
  - `num_new_domain_ctl` `[new_num_domain]` (Detail)
  - `new_sph_mode_prefix` `[new_sph_prefix]` (Detail)
  - `new_restart_prefix` `[new_rst_prefix]` (Detail)
  - `delete_original_data_flag` `[YES or NO]` (Detail)
- `control`
  - `time_step_ctl` (Detail)
    - `i_step_init_ctl` `[integer]` (Detail)

- `i_step_finish_ctl` `[integer]` (Detail)
- `i_step_rst_ctl` `[integer]` (Detail)

- `newrst_magne_ctl` (Detail)
  - `magnetic_field_ratio_ctl [ratio]` (Detail)

16 Module dependency program (`module_dependency`)

This program is only used to generate Makefile in `work` directory. Most of case, Fortran 90 modules have to compiled prior to be referred by another fortran90 routines. This program is generates dependency lists in Makefile. To use this program, the following limitation is required.

- One source code has to consist of one module.
- The module name should be the same as the file name.

17 Time averaging programs (`t_ave_picked_sph_coefs`, `t_ave_sph_mean_square`, and `t_ave_nusselt`)

These small programs are used to evaluate time average and standard deviation of the time sequencial data.

18 Visualization using field data

The field data is written by XDMF or VTK data format using Cartesian coordinate. In this section we briefly introduce how to display the radial magnetic field using ParaView as an example.

After the starting Paraview, the file to be read is chosen in the file menu, and press ”apply”, button. Then, Paraview load the data from files (see Figure 11). Because the magnetic field is saved by the Cartesian coordinate, the radial magnetic field is obtained by the calculator tool. The procedure is as following (see Figure 12)

1. Push calculator button.
2. Choose ”Point Data” in Attribute menu

Table 14: List of programs to take time average

name	Program to use
[vol_pwr_prefix].dat	t_ave_sph_mean_square
[vol_pwr_prefix]_l.dat	t_ave_sph_mean_square
[vol_pwr_prefix]_m.dat	t_ave_sph_mean_square
[vol_pwr_prefix]_lm.dat	t_ave_sph_mean_square
[layer_pwr_prefix]_l.dat	t_ave_sph_mean_square
[layer_pwr_prefix]_m.dat	t_ave_sph_mean_square
[layer_pwr_prefix]_lm.dat	t_ave_sph_mean_square
[picked_sph_prefix].dat	t_ave_picked_sph_coefs
[nusselt_number_prefix].dat	t_ave_sph_mean_square

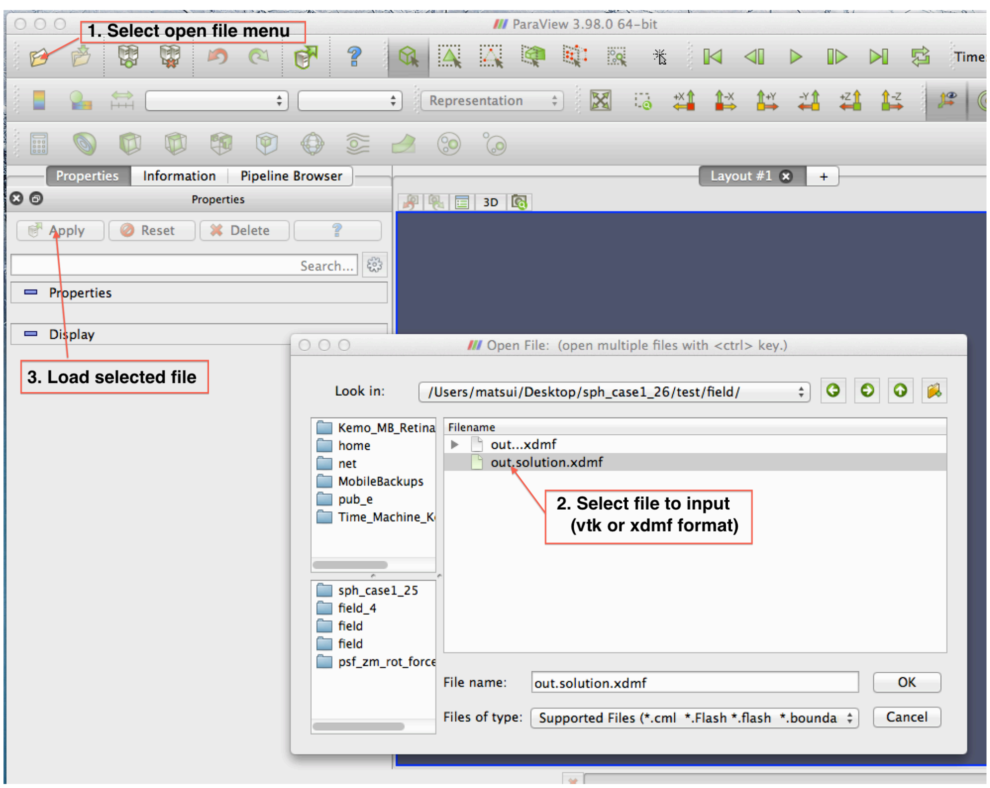


Figure 11: File open window for ParaView

3. Input data name for radial magnetic field (”B_r” in Figure 12)

4. Enter the equation to evaluate radial mantic field $B_r = \boldsymbol{B} \cdot \boldsymbol{r}/|r|$.

5. Finally, push ”Apply” button.

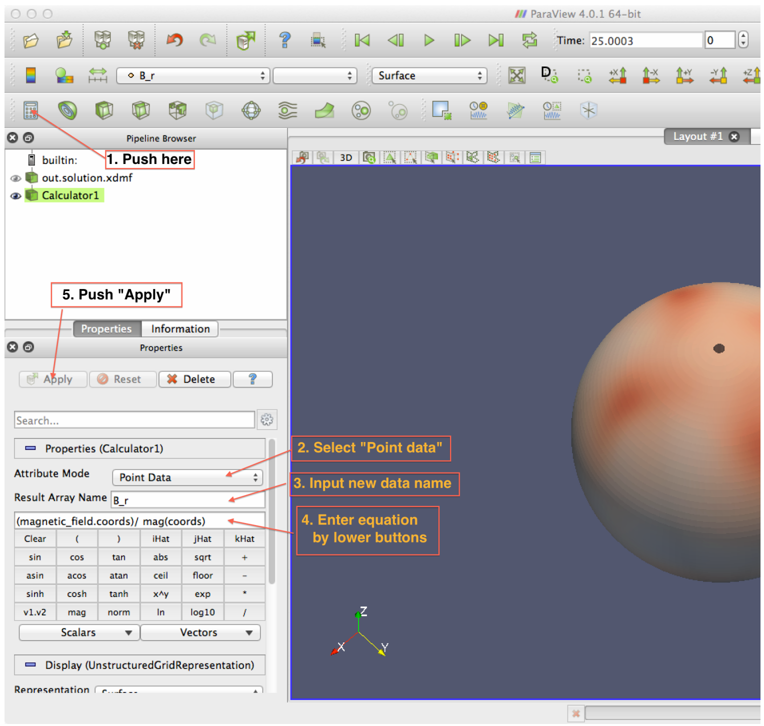


Figure 12: File open window for ParaView

After obtaining the radial mantric field, the image in figure 13 is obtained by using ”slice” and ”Contour” tools with appropriate color mapping.

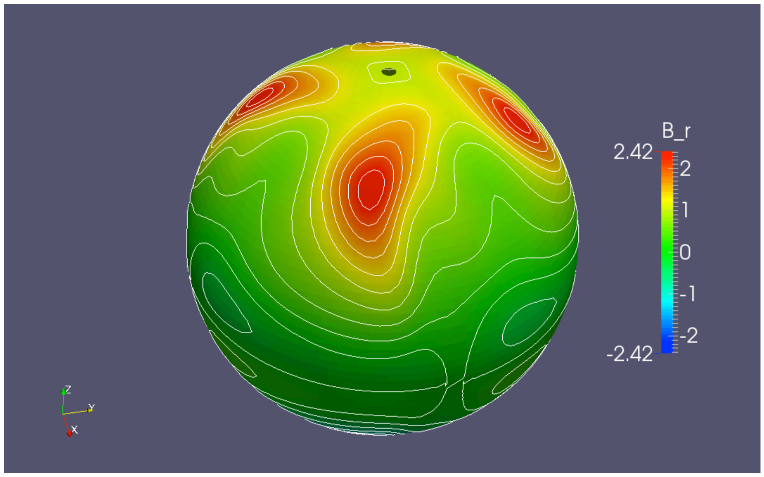


Figure 13: Visualization of radial magnetic field by Paraview.

References

[1] Bullard, E. C. and Gellman, H., Homogeneous dynamos and terrestrial magnetism, *Proc. of the Roy. Soc. of London*, **A247**, 213–278, 1954.

[2] Christensen, U.R., Aubert, J., Cardin, P., Dormy, E., Gibbons, S., Glatzmaier, G. A., Grote, E., Honkura, H., Jones, C., Kono, M., Matsushima, M., Sakuraba, A., Takahashi, F., Tilgner, A., Wicht, J. and Zhang, K., A numerical dynamo benchmark, *Physics of the Earth and Planetary Interiors*, **128**, 25–34, 2001.

Appendix A Definition of parameters for control files

A.1 data_files_def

File names and number of processes and threads are defined in this block.
(Back to control_MHD)
(Back to control_sph_shell)
(Back to control_assemble_sph)

num_subdomain_ctl [Num_PE]
Number of subdomain for the MPI program [Num_PE] is defined by integer. If number of processes in mpirun -np is different from number of subdomains, program will be stopped with message.

num_smp_ctl [Num_Threads]
Number of SMP threads for OpenMP [Num_Threads] is defined by integer. You can set larger number than the actual umber of thread to be used. If actual number of thread is less than this number, number of threads is set to the number which is defined in this field.

sph_file_prefix [sph_prefix]
File prefix of spherical harmonics indexing and FEM mesh file [sph_prefix] is defined by text. Process ID and extension are added after this file prefix.

boundary_data_file_name [boundary_data_name]
File name of boundary condition data file [boundary_data_name] is defined by text.

restart_file_prefix [rst_prefix]
File prefix of spectrum data for restarting and snapshots [rst_prefix] is defined by text. Step number, process ID, and extension are added after this file prefix.

field_file_prefix [fld_prefix]
File prefix of field data for visualize snapshots [fld_prefix] is defined by text. Step number and file extension are added after this file prefix.

field_file_fmt_ctl [fld_format]
Field data field format for visualize snapshots [fld_format] is defined by text. The following formats are currently supported.

single_HDF5 Merged HDF5 file (Available if HDF5 library is linked)

single_VTK Merged VTK file (Default)

VTK Distributed VTK file

A.2 phys_values_ctl

Fields for the simulation are defined in this block.
(Back to control_MHD)

`array nod_value_ctl  [Field] [Viz_flag] [Monitor_flag]`
Fields name [Field] for the simulation are listed in this array. If required fields for simulation are not in the list, simulation program adds required field in the list, but does not output any field data and monitoring data. [Viz_flag] is set to output of the field data for visualization by

VIz_On Write field data to VTK file

VIz_Off Do not write field data to VTK file.

In the [Monitor_flag], output in the monitoring data is defined by

Monitor_On Write spectrum into monitoring data

Monitor_Off Do not write spectrum into monitoring data

Supported field in the present version is listed in Table 15

A.3 time_evolution_ctl

Fields for time evolution are defined in this block.
(Back to control_MHD)

`array time_evo_ctl  [Field]`
Fields name for time evolution are listed in this array in [Field] by text. Available fields are listed in Table 16.

A.4 boundary_condition

Boundary condition are defined in this block.
(Back to control_MHD)

Table 15: List of field name

[Name]	field name	Description
velocity	Velocity	$\boldsymbol{u}$
vorticity	Vorticity	$\boldsymbol{\omega} = \nabla \times \boldsymbol{u}$
pressure	Pressure	P
temperature	Temperature	T
perturbation_temp	Perturbation of temperature	$\Theta = T - T_0$
heat_source	Heat source	q_T
composition	Composition variation	C
composition_source	Composition source	q_C
magnetic_field	Magnetic field	$\boldsymbol{B}$
current_density	Current density	$\boldsymbol{J} = \nabla \times \boldsymbol{B}$
electric_field	Electric field	$\boldsymbol{E} = \sigma \left(\boldsymbol{J} - \boldsymbol{u} \times \boldsymbol{B} \right)$
viscous_diffusion	Viscous diffusion	$-\nu \nabla \times \nabla \times \boldsymbol{u}$
buoyancy	Thermal buoyancy	$-\alpha_T T \boldsymbol{g}$
composite_buoyancy	Compositional buoyancy	$-\alpha_C C \boldsymbol{g}$
Lorentz_force	Lorentz force	$\boldsymbol{J} \times \boldsymbol{B}$
Coriolis_force	Coriolis force	$-2\Omega \hat{z} \times \boldsymbol{u}$
thermal_diffusion	Termal diffusion	$\kappa_T \nabla^2 T$
grad_temp	Temperature gradient	∇T
heat_flux	Advective heat flux	$\boldsymbol{u} T$
composition_diffusion	Compositional diffusion	$\kappa_C \nabla^2 C$
grad_composition	Composition gradient	∇C
composite_flux	Advective composition flux	$\boldsymbol{u} C$
magnetic_diffusion	Magnetic diffusion	$-\eta \nabla \times \nabla \times \boldsymbol{B}$
poynting_flux	Poynting flux	$\boldsymbol{E} \times \boldsymbol{B}$
rot_Lorentz_force	Curl of Lorentz force	$\nabla \times (\boldsymbol{J} \times \boldsymbol{B})$
rot_Coriolis_force	Curl of Coriolis force	$-2\Omega \nabla \times (\hat{z} \times \boldsymbol{u})$
rot_buoyancy	Curl of thermal buoyancy	$-\nabla \times (\alpha_T T \boldsymbol{g})$
rot_composite_buoyancy	Curl of compositional buoyancy	$-\nabla \times (\alpha_C C \boldsymbol{g})$
buoyancy_flux	Buoyancy flux	$-\alpha_T T \boldsymbol{g} \cdot \boldsymbol{u}$
Lorentz_work	Work of Lorentz force	$\boldsymbol{u} \cdot (\boldsymbol{J} \times \boldsymbol{B})$

Table 16: List of field name for time evolution

label	field name	Description
`velocity`	Velocity	$\boldsymbol{u}$
`temperature`	Temperature	T
`composition`	Composition variation	C
`magnetic_field`	Magnetic field	$\boldsymbol{B}$

`array bc_temperature [Group] [Type] [Value]`
Boundary conditions for temperature are defined by this array. Position of boundary is defined in `[Group]` column by `ICB` or `CMB`. The following type of boundary conditions are available for temperature in `[Type]` column.

`fixed` Fixed homogeneous temperature on the boundary. The fixed value is defined in `[Value]` by real.

`fixed_file` Fixed temperature defined by external file. `[Value]` in this line is ignored. See section 10.3.

`fixed_flux` Fixed homogeneous heat flux on the boundary. The value is defined in `[Value]` by real. Positive value indicates outward flux from fluid shell. (*e.g.* Flux to center at ICB and Flux to mantle at CMB are positive.)

`fixed_flux_file` Fixed heat flux defined by external file. `[Value]` in this line is ignored. See section 10.3.

`array bc_velocity [Group] [Type] [Value]`
Boundary conditions for velocity are defined by this array. Position of boundary is defined in `[Group]` by `ICB` or `CMB`. The following boundary conditions are available for velocity in `[Type]` column.

`non_slip_sph` Non-slip boundary is applied to the boundary defined in `[Group]`. Real value is required in `[Value]`, but they value is not used in the program.

`free_slip_sph` Free-slip boundary is applied to the boundary defined in `[Group]`. Real value is required in `[Value]`, but they value is not used in the program.

`rot_inner_core` If this condition is set, inner core ($r < r_i$) rotation is solved by using viscous torque and Lorentz torque. This boundary condition can be used for `ICB`, and grid is filled to center. Real value is required in `[Value]`, but they value is not used in the program.

`rot_x` Set constant rotation around x-axis in `[Value]` by real. Rotation vector can be defined with `rot_y` and `rot_z`.

`rot_y` Set constant rotation around y-axis in `[Value]` by real. Rotation vector can be defined with `rot_z` and `rot_x`.

`rot_z` Set constant rotation around z-axis in `[Value]` by real. Rotation vector can be defined with `rot_x` and `rot_y`.

`array bc_magnetic_field  [Group]  [Type]  [Value]`
Boundary conditions for magnetic field are defined by this array. Position of boundary is defined in `[Group]` by `to_Center`, `ICB`, or `CMB`. The following boundary conditions are available for magnetic field in `[Type]` column.

`insulator` Magnetic field is connected to potential field at boundary defined in [Group]. real value is required at `[Value]`, but they value is not used in the program.

`sph_to_center` If this condition is set, magnetic field in conductive inner core ($r < r_i$) is solved. This boundary condition can be used for `ICB`, and grid is filled to center. The value at `[Value]` does not used.

`array bc_composition  [Group]  [Type]  [Value]`
Boundary conditions for composition variation are defined by this array. Position of boundary is defined in `[Group]` by `ICB` or `CMB`. The following boundary conditions are available for composition variation in `[Type]` column.

`fixed` Fixed homogeneous composition on the boundary. The fixed value is defined in `[Value]` by real.

`fixed_file` Fixed composition defined by external file. `[Value]` in this line is ignored. See section 10.3.

`fixed_flux` Fixed homogeneous compositional flux on the boundary. The value is defined in `[Value]` by real. Positive value indicates outward flux from fluid shell. (*e.g.* Flux to center at ICB and Flux to mantle at CMB are positive.)

`fixed_flux_file` Fixed compositional flux defined by external file. `[Value]` in this line is ignored. See section 10.3.

A.5 `forces_define`

Forces for the momentum equation are defined in this block.
(Back to `control_MHD`)

`array force_ctl [Force]`
Name of forces for momentum equation are listed in `[Force]` by text. The following fields are available.

Table 17: List of force

Label	Field name	Equation
`Coriolis`	Coriolis force	$-2\Omega\hat{z} \times \boldsymbol{u}$
`Lorentz`	Lorentz force	$\boldsymbol{J} \times \boldsymbol{B}$
`gravity`	Thermal buoyancy	$-\alpha_T T \boldsymbol{g}$
`Composite_gravity`	Compositional buoyancy	$-\alpha_C C \boldsymbol{g}$

A.6 `dimensionless_ctl`

Dimensionless numbers are defined in this block.
(Back to `control_MHD`)

`array dimless_ctl [Name] [Value]`
Dimensionless are listed in this array. The name is defined in `[Name]` by text, and value is defined in `[Value]` by real. These name of the dimensionless numbers are used to construct coefficients for each terms in governing equations. The following names can not be used because of reserved name in the program.

Table 18: List of reserved name of dimensionless numbers

label	field name	value
`Zero`	zero	0.0
`One`	one	1.0
`Two`	two	2.0
`Radial_35`	Ratio of outer core thickness to whole core	$0.65 = 1 - 0.35$

A.7 coefficients_ctl

Coefficients of each term in governing equations are defined in this block. Each coefficients are defined by list of name of dimensionless number [Name] and its power [Power]. For example, coefficient for Coriolis term for the dynamo benchmark $2E^{-1}$ is defined as

```
array coef_4_Coriolis_ctl   2
  coef_4_Coriolis_ctl       Two             1.0
  coef_4_Coriolis_ctl       Ekman_number  -1.0
end array coef_4_Coriolis_ctl
```

(Back to control_MHD)

A.7.1 thermal

Coefficients of each term in heat equation are defined in this block.
(Back to control_MHD)

coef_4_termal_ctl [Name] [Power]
Coefficient for evolution of temperature $\dfrac{\partial T}{\partial t}$ and advection of heat $(\boldsymbol{u} \cdot \nabla) T$ is defined by this array.

coef_4_t_diffuse_ctl [Name] [Power]
Coefficient for thermal diffusion $\kappa_T \nabla^2 T$ is defined by this array.

coef_4_heat_source_ctll [Name] [Power]
Coefficient for heat source q_T is defined by this array.

A.7.2 momentum

Coefficients of each term in momentum equation are defined in this block.
(Back to control_MHD)

coef_4_velocity_ctl [Name] [Power]
Coefficient for evolution of velocity $\dfrac{\partial \boldsymbol{u}}{\partial t}$ (or $\dfrac{\partial \boldsymbol{\omega}}{\partial t}$ for the vorticity equation) and advection $-\boldsymbol{\omega} \times \boldsymbol{u}$ (or $-\nabla \times (\boldsymbol{\omega} \times \boldsymbol{u})$ for the vorticity equation) is defined by this array.

`coef_4_press_ctl [Name] [Power]`
Coefficient for pressure gradient $-\nabla P$ is defined by this array. Pressure does not appear the vorticity equation which is used for the time integration. But this coefficient is used to evaluate pressure field.

`coef_4_v_diffuse_ctl [Name] [Power]`
Coefficient for viscous diffusion $-\nu \nabla \times \nabla \times \boldsymbol{u}$ is defined by this array.

`coef_4_buoyancy_ctl [Name] [Power]`
Coefficient for buoyancy $-\alpha_T T \boldsymbol{g}$ is defined by this array.

`coef_4_Coriolis_ctl [Name] [Power]`
Coefficient for Coriolis force $-2\Omega \hat{z} \times \boldsymbol{u}$ is defined by this array.

`coef_4_Lorentz_ctl [Name] [Power]`
Coefficient for Lorentz force $\rho_0^{-1} \boldsymbol{J} \times \boldsymbol{B}$ is defined by this array.

`coef_4_composit_buoyancy_ctl [Name] [Power]`
Coefficient for compositional buoyancy $-\alpha_C C \boldsymbol{g}$ is defined by this array.

A.7.3 `induction`

Coefficients of each term in magnetic induction equation are defined in this block.
(Back to `control_MHD`)

`coef_4_magnetic_ctl [Name] [Power]`
Coefficient for evolution of temperature $\dfrac{\partial \boldsymbol{B}}{\partial t}$ is defined by this array.

`coef_4_m_diffuse_ctl [Name] [Power]`
Coefficient for magnetic diffusion $-\eta \nabla \times \nabla \times \boldsymbol{B}$ is defined by this array.

`coef_4_induction_ctl [Name] [Power]`
Coefficient for magnetic induction $\nabla \times (\boldsymbol{u} \times \boldsymbol{B})$ is defined by this array.

A.7.4 `composition`

Coefficients of each term in composition equation are defined in this block.
(Back to `control_MHD`)

`coef_4_composition_ctl [Name] [Power]`
Coefficient for evolution of composition variation $\frac{\partial C}{\partial t}$ and advection of heat $(\boldsymbol{u} \cdot \nabla) C$ is defined by this array.

`coef_4_c_diffuse_ctl [Name] [Power]`
Coefficient for compositional diffusion $\kappa_C \nabla^2 C$ is defined by this array.

`coef_4_composition_source_ctll [Name] [Power]`
Coefficient for composition source q_C is defined by this array.

A.8 `temperature_define`

Reference of temperature T_0 is defined in this block. If reference of temperature is defined, perturbation of temperature $\Theta = T - T_0$ is used for time evolution and buoyancy.
(Back to `control_MHD`)

`ref_temp_ctl [REFERENCE_TEMP]`
Type of reference temperature is defined by text. The following options are available for `[REFERENCE_TEMP]`.

`none` Reference of temperature is not defined. Temperature T is used to time evolution and thermal buoyancy.

`spherical_shell` Reference of temperature is set by

$$T_0 = \frac{1}{(r_h - r_l)} \left[r_l T_l - r_h T_h + \frac{r_l r_h}{r} (T_h - T_l) \right].$$

`low_temp_ctl` Amplitude of low reference temperature T_l and its radius r_l (Generally $r_l = r_o$) are defined in this block.

`high_temp_ctl` Amplitude of high reference temperature T_h and its radius r_h (Generally $r_h = r_i$) are defined in this block.

`depth [RADIUS]`
Radius for reference temperature is defined by real.

`temperature [TEMPERATURE]`
Temperature for reference temperature is defined by real.

A.9 `time_step_ctl`

Time stepping parameters are defined in this block.
(Back to `control_MHD`)
(Back to `control_assemble_sph`)

`elapsed_time_ctl [ELAPSED_TIME]`
Elapsed (wall clock) time (second) for simulation `[ELAPSED_TIME]` is defined by real. This parameter varies if end step `[ISTEP_FINISH]` is defined to `-1`. If simulation runs for given time, program output spectrum data `[rst_prefix].elaps.[process #].fst` immediately, and finish the simulation.

`i_step_init_ctl [ISTEP_START]`
Start step of simulation `[ISTEP_START]` is defined by integer. if `[ISTEP_START]` is set to `-1` and `[INITIAL_TYPE]` is set to `start_from_rst_file`, program read spectrum data file `[rst_prefix].elaps.[process #].fst` and start the simulation.

`i_step_finish_ctl [ISTEP_FINISH]`
End step of simulation `[ISTEP_FINISH]` is defined by integer. If this value is set to `-1`, simulation stops when elapsed time reaches to `[ELAPSED_TIME]`.

`i_step_check_ctl [ISTEP_MONITOR]`
Increment of time step for monitoring data `[ISTEP_MONITOR]` is defined by integer.

`i_step_rst_ctl [ISTEP_RESTART]`
Increment of time step to output spectrum data for restarting `[ISTEP_RESTART]` is defined by integer.

i_step_field_ctl [ISTEP_FIELD]
Increment of time step to output field data for visualization [ISTEP_FIELD] is defined by integer. If [ISTEP_FIELD] is set to be 0, no field data are written.

dt_ctl [DELTA_TIME]
Length of time step Δt is defined by real value.

time_init_ctl [INITIAL_TIME]
Initial time t_0 is defined by real value. This value is ignored if simulation starts from restart data.

A.10 restart_file_ctl

Initial field for simulation is defined in this block.
(Back to control_MHD)

rst_ctl [INITIAL_TYPE]
Type of Initial field is defined by text. The following parameters are available for [INITIAL_TYPE].

- No_data No initial data file. Small temperature perturbation and seed magnetic field are set as an initial field.
- start_from_rst_file Initial field is read from spectrum data file. File prefix is defined by restart_file_prefix.
- Dynamo_benchmark_0 Generate initial field for dynamo benchmark case 0
- Dynamo_benchmark_1 Generate initial field for dynamo benchmark case 1
- Dynamo_benchmark_2 Generate initial field for dynamo benchmark case 2
- Pseudo_vacuum_benchmark Generate initial field for pseudo vacuum dynamo benchmark

A.11 time_loop_ctl

Time evolution scheme is defined in this block.
(Back to control_MHD)

scheme_ctl [EVOLUTION_SCHEME]
Time evolution scheme is defined by text. Currently, Crank-Nicolson scheme is only available for diffusion terms.

Crank_Nicolson Crank-Nicolson scheme for diffusion terms and second order Adams-Bashforth scheme the other terms.

coef_imp_v_ctl [COEF_INP_U]
Coefficients for the implicit parts of the Crank-Nicolson scheme for viscous diffusion [COEF_INP_U] is defined by real.

coef_imp_t_ctl [COEF_INP_T]
Coefficients for the implicit parts of the Crank-Nicolson scheme for thermal diffusion [COEF_INP_T] is defined by real.

coef_imp_b_ctl [COEF_INP_B]
Coefficients for the implicit parts of the Crank-Nicolson scheme for magnetic diffusion [COEF_INP_B] is defined by real.

coef_imp_c_ctl [COEF_INP_C]
Coefficients for the implicit parts of the Crank-Nicolson scheme for compositional diffusion [COEF_INP_C] is defined by real.

FFT_library_ctl [FFT_Name]
FFT library name for Fourier transform is defined by text. The following libraries are available for [FFT_Name]. If this flag is not defined, program searches the fastest library in the initialization process.

FFTW Use FFTW

FFTPACK Use FFTPACK

FFT_library_ctl [FFT_Name]
Loop configuration for Legendre transform is defined by text. The following settings are available for [Leg_Loop]. If this flag is not defined, program searches the fastest approarch in the initialization process.

Inner_radial_loop Loop for the radial grids is set as the innermost loop

Outer_radial_loop Loop for the radial grids is set as the outermost loop

Long_loop Long one-dimentional loop is used

A.12 sph_monitor_ctl

Monitoring data is defined in this block. Monitoring data output root mean square, average, Gauss coefficients, or specific components of spectrum data which are flagged by Monitor_On in nod_value_ctl array.
(Back to control_MHD)

volume_average_prefix [vol_ave_prefix]
File prefix for volume average data [vol_ave_prefix] is defined by Text. Program add .dat extension after this file prefix. If this file prefix is not defined, volume average data are not generated.

volume_pwr_spectr_prefix [vol_pwr_prefix]
File prefix for mean square spectrum data averaged over the fluid shell [vol_pwr_prefix] is defined by Text.

Spectrum as a function of degree l is written in [vol_pwr_prefix])_l.dat, spectrum as a function of order m is written in [vol_pwr_prefix]_m.dat, and spectrum as a function of $(l-m)$ is written in [vol_pwr_prefix]_lm.dat. This prefix is also used for the file name of the volume mean square data as [vol_pwr_prefix].dat. If this file prefix is not defined, volume spectrum data are not generated and volume mean square data is written as sph_pwr_volume.dat.

layered_pwr_spectr_prefix [layer_pwr_prefix]
File prefix for mean square spectrum data averaged over each sphere surface [layer_pwr_prefix] is defined by Text.

Spectrum as a function of degree l is written in [layer_pwr_prefix]_l.dat, spectrum as a function of order m is written in [layer_pwr_prefix]_m.dat, and spectrum as a function of $(l-m)$ is written in [layer_pwr_prefix]_lm.dat. If this file prefix is not defined, sphere averaged spectrum data are not generated.

picked_sph_prefix [picked_sph_prefix]
File prefix for picked spectrum data [picked_sph_prefix] is defined by Text. Program add .dat extension after this file prefix. If this file prefix is not defined, picked spectrum data are not generated.

`gauss_coefs_prefix [gauss_coef_prefix]`
File prefix for Gauss coefficients `[gauss_coef_prefix]` is defined by Text. Program add `.dat` extension after this file prefix. If this file prefix is not defined, Gauss coefficients data are not generated.

`gauss_coefs_radius_ctl [gauss_coef_radius]`
Normalized radius to obtain Gauss coefficients `[gauss_coef_radius]` is defined by real. Gauss coefficients are evaluated from the poloidal magnetic field at CMB by assuming electrically insulated mantle. Do not set `[gauss_coef_radius]` less than the outer core radius r_o.

`nusselt_number_prefix [nusselt_number_prefix]`
File prefix for Nusselt number data at ICB and CMB `[nusselt_number_prefix]` is defined by Text. Program add `.dat` extension after this file prefix. If this file prefix is not defined, Nusselt number data are not generated.
CAUTION: Nusselt number is not evaluated if heat source is exsist.

`array pick_layer_ctl [Layer #]` List of radial grid point number `[Layer #]` to output spectrum data by integer. If this array is not defined, picked spectrum data are written for all radial grid points.

`array pick_sph_spectr_ctl [Degree] [Order]`
List of spherical harmonics mode l and m of spectrum data to output. `[Degree]` and `[Order]` are defined by integer.

`array pick_sph_degree_ctl [Degree]`
Degrees l to output spectrum data are listed in `[Degree]` by integer. All spectrum data with listed degree l is output in file.

`array pick_sph_order_ctl [Order]`
Order m to output spectrum data are listed in `[Order]` by integer. All spectrum data with listed order m is output in file.

`array pick_gauss_coefs_ctl [Degree] [Order]`
List of spherical harmonics mode l and m of Gauss coefficients to output. `[Degree]` and `[Order]` are defined by integer.

`array pick_gauss_coef_degree_ctl  [Degree]`
Degrees l to output Gauss coefficients are listed in `[Degree]` by integer. All Gauss coefficients with listed l is output in file.

`array pick_gauss_coef_order_ctl  [Order]`
Orders m to output Gauss coefficients are listed in `[Order]` by integer. All Gauss coefficients with listed order m is output in file.

`nphi_mid_eq_ctl  [Nphi_mid_equator]`
Number of grid points `[Nphi_mid_equator]` in longitudinal direction to evaluate mid-depth of the shell in the equatorial plane for dynamo benchmark is defined as integer. If `[Nphi_mid_equator]` is not defined or less than zero, `[Nphi_mid_equator]` is set set number grid as the input spherical transform data.

A.13 `num_domain_ctl`

Parallelization is defined in this block. Domain decomposition is defined for spectrum data, field data, and Legendre transform.
(Back to `control_sph_shell`)

`num_radial_domain_ctl  [Ndomain]`
Number of subdomains in the radial direction for the spherical grid (r, θ, ϕ) and spherical transforms (r, θ, m) and (r, l, m).

`num_horiaontal_domain_ctl  [Ndomain]`
Number of subdomains in the horizontal direction. The number will be the number of subdomains for the meridional directios for the spherical grid (r, θ, ϕ) and Fourier transform (r, θ, m). For Legendre transform (r, θ, m) and (r, l, m), the number will be the number of subdomains for the h.armonics ordedr m.

`num_domain_sph_grid  [Direction] [Ndomain]`
Definition of number of subdomains for physical data in spherical coordinate (r, θ, ϕ). Direction `radial` or `meridional` is set in `[Direction]`, and number of subdomains `[Ndomain]` are defined in the integer field.

`num_domain_legendre  [Direction] [Ndomain]`
Definition of number of subdomains for Legendre transform between (r, θ, m) and (r, l, m). Direction `radial` or `zonal` is set in `[Direction]`, and number of subdomains `[Ndomain]` are defined in the integer field.

`num_domain_spectr  [Direction] [Ndomain]`
Definition of number of subdomains for spectrum data in (r, l, m). Direction `modes` is set in the `[Direction]` field, and number of subdomains `[Ndomain]` are defined in the integer field.

A.14 `num_grid_sph`

Spatial resolution of the spherical shell is defined in this block.
(Back to `control_sph_shell`)

`truncation_level_ctl  [Lmax]`
Truncation level L is defined by integer. Spherical harmonics is truncated by triangular $0 \leq l \leq L$ and $0 < m < l$.

`ngrid_meridonal_ctl  [Ntheta]`
Number of grid in the meridional direction `[Ntheta]` is defined by integer.

`ngrid_zonal_ctl  [Nphi]`
Number of grid in the zonal direction `[Nphi]` is defined by integer.

`raidal_grid_type_ctl  [explicit, Chebyshev, or equi_distance]`
Type of the radial grid spacing is defined by text. The following types are supported in Calypso.

`explicit` Equi-distance grid

`Chebyshev` Chebyshev collocation points

`equi_distance` Set explicitly by `r_layer` array

`num_fluid_grid_ctl  [Nr_shell]`
(This option works with `radial_grid_type_ctl` is `explicit` or `Chebyshev`.) Number of layer in the fluid shell `[Nr_shell]` is defined by integer. Number of grids including CMB and ICB will be (`[Nr_shell]` + 1).

`fluid_core_size_ctl [Length]`
(This option works with `radial_grid_type_ctl` is `explicit` or `Chebyshev`.) Size of the outer core `[Length]` ($= r_o - r_i$) is defined by real.

`ICB_to_CMB_ratio_ctl [R_ratio]`
(This option works with `radial_grid_type_ctl` is `explicit` or `Chebyshev`.) Ratio of the inner core radius to outer core `[R_ratio]` ($= r_i/r_o$) is defined by real.

`Min_radius_ctl [Rmin]`
(This option works with `radial_grid_type_ctl` is `explicit` or `Chebyshev`.) Minimum radius of the domains `[Rmin]` is defined by real. If this value is not defined, ICB becomes inner boundary of the domain.

`Max_radius_ctl [Rmax]`
(This option works with `radial_grid_type_ctl` is `explicit` or `Chebyshev`.) Maximum radius of the domains `[Rmax]` is defined by real. If this value is not defined, CMB becomes outer boundary of the domain.

`r_layer [Layer #] [Radius]`
(This option works with `[radial_grid_type_ctl]` is `explicit`.) List of the radial grid points in the simulation domain. Index of the radial point `[Layer #]` is defined by integer, and radius `[Radius]` is defined by real.

`array boundaries_ctl [Boundary_name] [Layer #]`
(This option works with `[radial_grid_type_ctl]` is `explicit`.) Boundaries of the simulation domain is defined by `[Layer #]` in `[r_layer]` array. The following boundary name can be defined for `[Boundary_name]`.

`to_Center` Inner boundary of the domain to fill the center.

`ICB` ICB

`CMB` CMB

A.15 `new_data_files_def`

File names and number of processes for new domain decomposed data are defined in this block.
(Back to `control_assemble_sph`)

num_new_domain_ctl [new_num_domain]
Number of subdomain for new new decomposed data [new_num_domain] is defined by integer.

new_sph_mode_prefix [new_sph_prefix]
File prefix of new spherical harmonics indexing [new_sph_prefix] is defined by text.

new_restart_prefix [new_rst_prefix]
File prefix of new spectrum data [new_rst_prefix] is defined by text.

delete_original_data_flag [delete_original_data_flag]
If this flag set to YES, original specter data is deleted at the end of program.

A.16 newrst_magne_ctl

Parameters to modify magnetic field are defined in this block.
(Back to control_assemble_sph)

magnetic_field_ratio_ctl [ratio]
Ratio of new magnetic field data to original magnetic field [ratio] is defined by real.

Appendix B GNU GENERAL PUBLIC LICENSE

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

(b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

(c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

(c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> one line to give the program's name and a brief idea of what it does.
> Copyright (C) yyyy name of author
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) yyyy name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
> This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w` and `show c`; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program
> ‘Gnomovision’ (which makes passes at compilers) written by James Hacker.
>
> signature of Ty Coon, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.