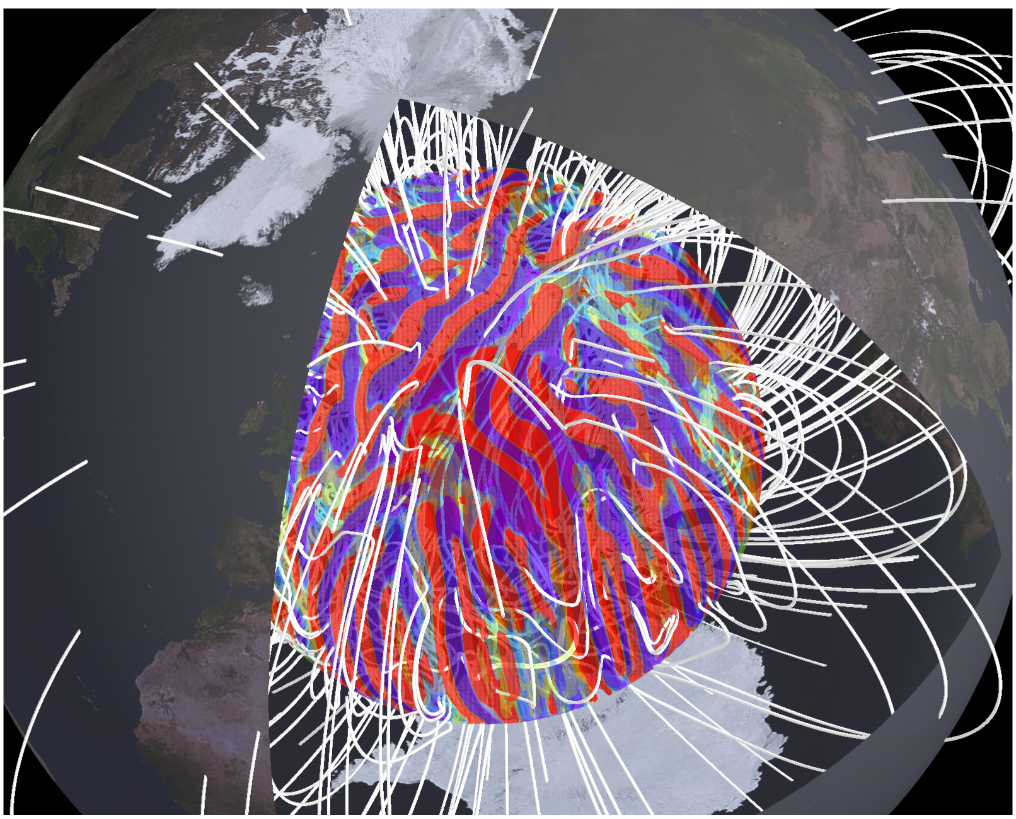
COMPUTATIONAL INFRASTRUCTURE FOR GEODYNAMICS (CIG)

Calypso

User Manual
Version 2.0



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. Under GeoFEM Project, Lee Chen developed cross sectioning, iso-surfacing, and volume rendering modules for data visualization for parallel computations..

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 6.2)
- 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 8.3.)

- 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 8.2 and 8.3.)

- 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 7.1.5)

2.2 Updates for Ver 1.2

In Version 1.2, the following features are implemented:

- To reduce the number of calculation, Legendre transform is calculated with taking account to the symmetry with respect to the equator. Time for Legendre transform is approximately half of that in Ver 1.1.

- BLAS library can be used for the Legendre transform optionally.

- Cross sectioning and isosurfacing module are newly implemented. These modules are re-written by Fortran90 from the parallel sectioning modules in GeoFEM by Lee Chen in C, and some features are added for visualizations of geodynamo simulations. See section 7.4 and 7.5.

- Initial data assemble program `assemble_mhd` is parallelized. This program can perform with any number of MPI processes, but we recommend to run the program with **one** process or the same number of processes as the number of subdomains for the target configuration which is defined by `num_new_domain_ctl`. See section 8.3.

- The time and time step information in the restart data can be modifield by `assemble_mhd`. See section 8.3

2.3 Updates for Ver 2.0

In Version 2.0, there are a number of changes as;

- Start using Fortran structures to reduce global instances.
- Using MPI-IO for data IO to generate a single date file from MPI processes
- Include interface to zlib library (`https://www.zlib.net`) for data IO with data compression. zlib is pre-installed in MacOS and most of Linux distributions.
- Calypso now support various format of data IO (ascii, binary, gzipped ascii, and gzipped binary) through MPI-IO.
- Include spherical harmonics index generator into simulation program. Consequently, we can start program without preprocessing.
- Modules to generate longitudinal average data is included into the simulation program.
- Due to the performance drops on massively parallel environment, spectrum data `[picked_sph_prefix].dat` is splitted into the data files for each spherical harmonics mode `[picked_sph_prefix]_l[degree]_m[order][c/s].dat`. Consequently, the file prefix [picked_sph_prefix] is recommends to includer subdirectory and that files are saved undere a subdirectory. (See Section 7.6.6 and tests under `tests/Dynamobench_case2`.)
- Number of array information is not required to define "array" block in control files.
- Block layout of the control file is changed for spatial resolution and spherical harmonic data IO.

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 Earth's 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

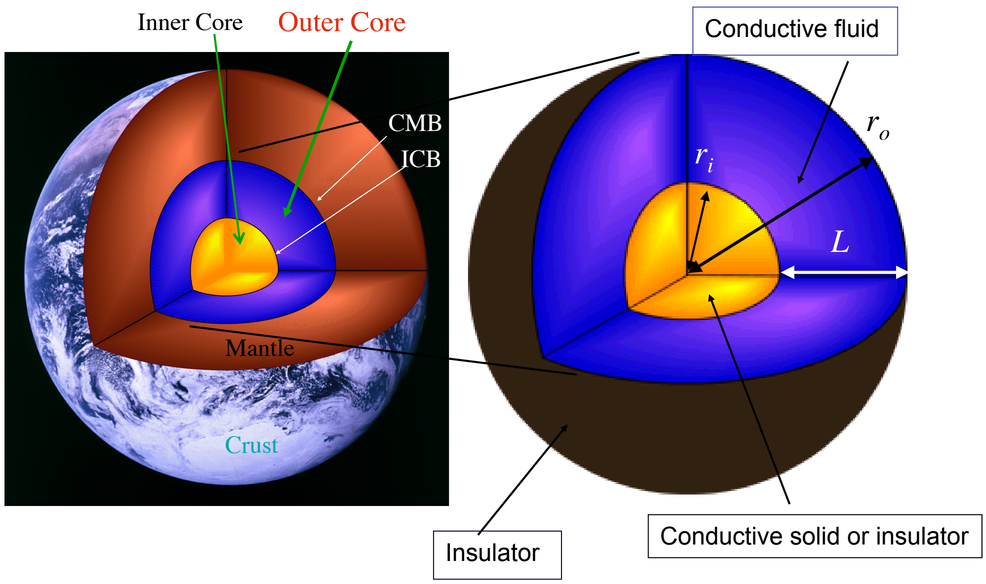


Figure 1: Rotating spherical shell modeled on the Earth's outer core.

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 $\mathbf{\Omega} = \Omega \hat{z}$. 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,

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

$$\begin{aligned}
\frac{\partial \boldsymbol{B}}{\partial t} &= -\eta \nabla \times \nabla \times \boldsymbol{B} + \nabla \times (\boldsymbol{u} \times \boldsymbol{B}), &(2)\\
\frac{\partial T}{\partial t} + (\boldsymbol{u} \cdot \nabla) T &= \kappa_T \nabla^2 T + q_T, &(3)\\
\frac{\partial C}{\partial t} + (\boldsymbol{u} \cdot \nabla) C &= \kappa_C \nabla^2 C + q_C, &(4)\\
\nabla \cdot \boldsymbol{u} &= \nabla \cdot \boldsymbol{B} = 0, &(5)\\
\boldsymbol{\omega} &= \nabla \times \boldsymbol{u}, &(6)
\end{aligned}$$

and

$$\boldsymbol{J} = \frac{1}{\mu_0} \nabla \times \boldsymbol{B}, \tag{7}$$

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,

$$\rho = \rho_0 \left[1 - \alpha_T (T - T_0) - \alpha_C (C - C_0)\right] \tag{8}$$

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

$$\begin{aligned}
\frac{\partial \boldsymbol{\omega}}{\partial t} + \nabla \times (\boldsymbol{\omega} \times \boldsymbol{u}) &= -\nu \nabla \times \nabla \times \boldsymbol{\omega} - 2\Omega \nabla \times (\hat{z} \times \boldsymbol{u}) \\
&\quad + \nabla \times \left(\frac{\rho}{\rho_0} \boldsymbol{g} \right) + \frac{1}{\rho_0} \nabla \times (\boldsymbol{J} \times \boldsymbol{B}), &(9)
\end{aligned}$$

and

$$\begin{aligned}
\nabla \cdot (\boldsymbol{\omega} \times \boldsymbol{u}) &= -\nabla^2 \left(P + \frac{1}{2} u^2 \right) - 2\Omega \nabla \cdot (\hat{z} \times \boldsymbol{u}) \\
&\quad + \nabla \cdot \left(\frac{\rho}{\rho_0} \boldsymbol{g} \right) + \frac{1}{\rho_0} \nabla \cdot (\boldsymbol{J} \times \boldsymbol{B}) &(10)
\end{aligned}$$

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

$$T(r,\theta,\phi) = \sum_{l=0}^{L}\sum_{m=-l}^{l} T_l^m(r) Y_l^m(\theta,\phi), \tag{11}$$

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

$$\boldsymbol{B}(r,\theta,\phi) = \sum_{l=1}^{L}\sum_{m=-l}^{l} \left(\boldsymbol{B}_{Sl}^{m} + \boldsymbol{B}_{Tl}^{m}\right), \tag{12}$$

where

$$\boldsymbol{B}_{Sl}^{m}(r,\theta,\phi) = \nabla\times\nabla\times\left(B_{Sl}^{m}(r)Y_l^m(\theta,\phi)\hat{r}\right), \tag{13}$$

$$\boldsymbol{B}_{Tl}^{m}(r,\theta,\phi) = \nabla\times\left(B_{Tl}^{m}(r)Y_l^m(\theta,\phi)\hat{r}\right). \tag{14}$$

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

$$\int Y_l^m Y_{l'}^{m'} \sin\theta d\theta d\phi = 4\pi\frac{1}{2l+1}\delta_{ll'}\delta_{mm'}, \tag{15}$$

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

5.2.1 Forward spherical harmonics transform

Calypso uses spherical harmonics expansion method (or so-called pseudo-spectrum method). In the present method, linear terms are solved by using the coefficients of the spherical harmonics expansion, and nonlinear terms are calculated in the physical grid space. Consequently, fields are transformed to grids space by the backward spherical transform using equations (11) to (14)., and nonlinear terms are transferred into the spherical harmonics coefficients by forward spherical harmonics transform.

The nonlinear terms (advection, Lorentz force, and induction, and heat flux terms) are evaluated in the physical space (r,θ,ϕ) and are not solenoidal fields (i.e. the divergence of the nonlinear terms are not to be 0). The forward spherical transform for a non-solenoidal vector field $\boldsymbol{A}$ can be expressed by the radial, horizontal divergence, and toroidal components as

$$
\begin{aligned}
A_{Rl}^{m}(r,t) &= N_l^{-1} \int \frac{l(l+1)}{r^2} Y_l^m A_r(r,\theta,\phi) d\sigma, && (16)\\
A_{Hl}^{m}(r,t) &= N_l^{-1} \int \frac{1}{r} \left[\frac{\partial Y_l^m}{\partial \theta} A_\theta(r,\theta,\phi) + \frac{1}{\sin\theta} \frac{\partial Y_l^m}{\partial \phi} A_\phi(r,\theta,\phi) \right] d\sigma, && (17)
\end{aligned}
$$

and,

$$
\begin{aligned}
A_{Tl}^{m}(r,t) &= N_l^{-1} \int \frac{1}{r} \left[\frac{1}{\sin\theta} \frac{\partial Y_l^m}{\partial \phi} A_\theta(r,\theta,\phi) - \frac{\partial Y_l^m}{\partial \theta} A_\phi(r,\theta,\phi) \right] d\sigma, && (18)\\
& && (19)
\end{aligned}
$$

where, $d\sigma = \sin\theta d\theta d\phi$ is the integration over a sphere. The non-solenoidal vector $\boldsymbol{A}$ can also describes the spherical harmonics coefficients for the poloidal A_{Sl}^{m}, toroidal A_{Tl}^{m}, and scalar potential φ_l^m as

$$
\boldsymbol{A} = \sum \left[-\nabla \varphi_l^m + \nabla \times \nabla \times (A_{Sl}^{m} \hat{r}) + \nabla \times (A_{Tl}^{m} \hat{r}) \right]. \quad (20)
$$

Consequently, the poloidal and toroidal components of the rotation of $\mathbf{A}$ can be described by

$$
\begin{aligned}
(\nabla \times \mathbf{A})_{Sl}^{m} &= A_{Tl}^{m} && (21)\\
(\nabla \times \mathbf{A})_{Tl}^{m} &= \frac{\partial^2 A_{Sl}^{m}}{\partial r^2} - \frac{l(l+1)}{r^2} A_{Sl}^{m} \\
&= -A_{Vl}^{m} + \frac{\partial A_{Hl}^{m}}{\partial r}, && (22)
\end{aligned}
$$

and, the divergence of $\mathbf{A}$ can be obtained by

$$
\begin{aligned}
(\nabla \cdot \mathbf{A})_{Tl}^{m} &= -\frac{\partial^2 \varphi_l^m}{\partial r^2} - \frac{2}{r} \varphi_l^m + \frac{l(l+1)}{r^2} \varphi_l^m \\
&= \frac{\partial A_{Vl}^{m}}{\partial r} + \frac{2}{r} A_{Vl}^{m} - \frac{l(l+1)}{r^2} A_{Hl}^{m}. && (23)
\end{aligned}
$$

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

$$G_{Lll'}^{Mmm'} = \int Y_L^M Y_l^m Y_{l'}^{m'} \sin\theta d\theta d\phi, \tag{24}$$

$$E_{Lll'}^{Mmm'} = \int Y_L^M \left(\frac{\partial Y_l^m}{\partial \theta} \frac{\partial Y_{l'}^{m'}}{\partial \phi} - \frac{\partial Y_l^m}{\partial \phi} \frac{\partial Y_{l'}^{m'}}{\partial \theta} \right) d\theta d\phi. \tag{25}$$

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 Radial discretization

In Calypso, spherical harmonic coefficients $f_l^m(r,t)$ are discretized by the second order finite difference method. A non-equidistant grids is widely used in geodynamo simulations to resolve boundary layers. Now, considering the n-th grid point at $r = r_n$ and neighboring grid points. The Taylor expansion of the spherical harmonic coefficients at $n+1$-th and $n-1$-th points $f_l^m(r_{n+1},t)$ and $f_l^m(r_{n-1},t)$ can be described by

$$\left(\begin{array}{c} f_l^m(r_n) \\ f_l^m(r_{n-1}) \\ f_l^m(r_{n+1}) \end{array} \right) = A_n \left(\begin{array}{c} f_l^m(r_n) \\ \partial_r f_l^m(r_n) \\ \partial_r r f_l^m(r_n) \end{array} \right), \tag{26}$$

where, A_n is

$$A_n = \left(\begin{array}{ccc} 1 & 0 & 0 \\ 1 & r_n - r_{n-1} & \left(r_n - r_{n-1}\right)^2/2 \\ 1 & r_{n+1} - r_n & \left(r_{n+1} - r_n\right)^2/2 \end{array} \right), \tag{27}$$

The first and second derivatives are expressed by

$$\left(\begin{array}{c} f_l^m(r_n) \\ \partial_r f_l^m(r_n) \\ \partial_r r f_l^m(r_n) \end{array} \right) = A_n^{-1} \left(\begin{array}{c} f_l^m(r_{n-1}) \\ f_l^m(r_n) \\ f_l^m(r_{n+1}) \end{array} \right). \tag{28}$$

At the boundaries, the first radial derivative of the coefficients $\partial_r f_l^m(r_1)$ is used instead of the grid outside of the boundaries. For example, A_n at the inner boundary $n = 1$ can be written by using the first differenciation as

$$\left(\begin{array}{c} f_l^m(r_1) \\ \partial_r f_l^m(r_1) \\ f_l^m(r_2) \end{array} \right) = A_1 \left(\begin{array}{c} f_l^m(r_1) \\ \partial_r f_l^m(r_1) \\ \partial_{rr} f_l^m(r_1) \end{array} \right). \tag{29}$$

where, A_1 is

$$A_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & r_{N+1} - r_N & \left(r_{N+1} - r_N\right)^2/2 \end{pmatrix}, \tag{30}$$

When $f_l^m(r_1)$ is fixed at the boundary, the second derivative $\partial_{rr} f_l^m(r_1)$ is not used. When the boundary condition for $\partial_r f_l^m(r_1)$ is given, the second derivative $\partial_{rr} f_l^m(r_1)$ is obtained by the boundary condition and $f_l^m(r_1)$.

5.5 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`.

5.5.1 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

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

and

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

5.5.2 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

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

and

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

5.5.3 Fixed rotation rate

If the boundary rotates with a rotation vector $\mathbf{\Omega}_b = (\Omega_{bx}, \Omega_{by}, \Omega_{bz})$, 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

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

5.5.4 Fixed homogenous temperature

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

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

and

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

5.5.5 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

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

5.5.6 Fixed composition

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

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

and

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

5.5.7 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

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

5.5.8 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

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

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

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

and

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

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

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

and

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

5.5.9 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

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

5.5.10 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 Compiler Requirements

Most of source code of Calypso are written in Fortran2003. Consequently, Fortran compiler with supporting fortran 2003 is required. We can obtain a number of information about Fortran from `http://fortranwiki.org/`, and you can also find a table of the supported features of Fortran 2003 standard at `http://fortranwiki.org/fortran/show/Fortran+2003+status`. In addition, C compiler is optionally required to us zlib support for compressed data IO.

6.2 Library Requirements

Calypso requires the following libraries.

- GNU make
- MPI libraries (OpenMPI, MPICH, etc)
- FFTPACK Ver 5.1D (`https://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. MPI library such as OpenMPI (`https://www.open-mpi.org`) or MPICH (`https://www.mpich.org`) can be installed by the most of package manager.

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

- OpenMP
- BLAS
- zlib (https://www.zlib.net)
- FFTW version 3 (`http://www.fftw.org`) including Fortran wrapper
- PARALLEL HDF5 (`https://support.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 FFT-PACK.

Zlib is used for compressed data IO. Zlib is installed in most of UNIX platforms.

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 SR24000 with Power 8 processors.

6.2.1 Installation of required softwares for Linux

GCC, the GNU Compiler Collection (https://gcc.gnu.org) is already installed in the most of Linux distributions. However, packages for development are need to be installed. For Ubuntu 20, for example, the required compilers and libraries can be installed by using `apt` command as following::

```
% sudo apt install build-essential
% sudo apt install pkg-config
% sudo apt install git
% sudo apt install gfortran
% sudo apt install libopenmpi-dev
% sudo apt install zlib1g
% sudo apt install zlib1g-dev
% sudo apt install libblas-dev
% sudo apt install libfftw3-dev
% sudo apt install libhdf5-openmpi-dev
```

6.2.2 Installation of required softwares for Mac OS

For MacOS, any fortran compiler needs to be installed because Xcode does not have fortran compiler. The easiest way is installing GCC by using a package manager such as macports (https://www.macports.org) or homebrew (https://brew.sh/index). By using the Macports,

Macports The required compiler and packages can be installed as followings as an example using GCC11. GCC in Macports includes gfortran compiler.

```
% sudo port install gcc11
% sudo port install openmpi-gcc11
% sudo port install fftw-3 +gcc11 +openmpi
% sudo port install hdf5 +fortran +gcc11 +openmpi +threadsafe
```

Homebrew The required compiler and packages can be installed as followings: GCC in Homebrew includes gfortran compiler.

```
% brew install gcc
% brew install open-mpi
% brew install fftw
% brew install lhdf5-mpi
```

6.3 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.

XL fortran

In XL fortran, preprocessor options is not specified by `-D...`, but `-Wf, '-D...'`. Please edit preprocessor macro option `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 6.7)

6.4 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.

6.5 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/`.

6.6 Install using `configure` command

6.6.1 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>
  --with-zlib=DIR root directory path of zlib installation defaults to
                    /usr/local or /usr if not found in /usr/local
  --without-zlib to disable zlib usage completely

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' \
? PKG_CONFIG_PATH='/Users/matsui/local/lib/pkgconfig'
```

At the end of the configuration, The following message can use to check if libraries can be refered correctly:

```
-----   Configuration summary   -------------------------------

host:         "x86_64-apple-darwin16.7.0"
host_alias:   ""
XL_FORTRAN:     ""
Use OpenMP ...         yes

Use BLAS ...           yes

Use FFTW3 ...          yes
Use parallel HDF5 ... yes

Use zlib at ...        yes

---------------------------------------------------------------
```

6.6.2 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 harmonics transform

`check_sph_grids`:
: Check program for data communication for spherical harmonics 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)

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

`sectioning`:
: Generate cross section and isosurface from field data and FEM mesh data.

`field_to_VTK`:
: Data transfer program from field and FEM mesh data to VTK format.

`psf_to_vtk`:
: Data transfer program from section and isosurface data to VTK format.

`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

`libcalypso_c.a:` Calypso library from C sources

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

6.6.3 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 `.`

6.6.4 Distclean

To revert the files and directory to the original package, use make distclean as

```
% make distclean
```

6.6.5 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`.

6.6.6 Construct dependecies (only for developper)

Fortran90 routines need to be build after modules which are used in the routines. C source files also need dependency among include files. Consequently, list of dependency of source files are saved in the file `Makefile.depends` in each directory. When you modify the source files with changing the module usage, `Makefile.depends` files need to be updated. To update the `Makefile.depends`files, use the `make` command at the `[CALYPSO_HOME]` directory as

```
% make depends
```

This process generate dependencies of the Fortran modules by program `make_f90depends`. For C source files, the dependency is generated by the gcc with `-MM -w -DDEPENDENCY_CHECK` option. Consequently, the dependencies need to be generated by the environment with gcc or compatible compiler. After generating the dependency, you can transfer the modified package and build without using gcc.

6.7 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)

BLAS_LIBS Library lists for BLAS (ex. -lblas)

ZLIB_CFLAGS Option flags for zlib (ex. -I/usr/include)

ZLIB_LIB Library lists for zlib (ex. -L/usr/lib -lz)

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.

6.8 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:

- Install directory

  `CMAKE_INSTALL_PREFIX`
  : Install directory

- Compiler settings

  `CMAKE_Fortran_COMPILER`
  : Fortran90 compiler.

  `CMAKE_c_COMPILER` C compiler.

  `CMAKE_Fortran_FLAGS`
  : Optimization flags for Fortran90 compiler.

  `CMAKE_c_FLAGS`
  : Optimization flags for C compiler.

- Option settings

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

  `CMAKE_DISABLE_FIND_PACKAGE_BLAS`
  : BLAS library is not linked if 'yes' is set in this valuable.

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

  `CMAKE_DISABLE_FIND_PACKAGE_ZLIB`
  : Zlib library is not linked if 'yes' is set in this valuable.

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

- Manual settings for optional features

  `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.

`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`.

The easiest example of using CMake on Mac OS X with gcc9 is the following:

```
% cd build
% cmake ~/CALYPSO/ -DCMAKE_Fortran_COMPILER=/opt/local/bin/gfortran-mp-9
? -DCMAKE_c_COMPILER=/opt/local/bin/gcc-mp-9 \
? -DCMAKE_Fortran_FLAGS="-O3 -g" -DCMAKE_c_FLAGS="-O3"
```

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. Calypso consists of the following programs:

- Preprocessing programs
  - `sph_initial_field` (See Section 8.2),
  - `sph_add_initial_field` (See Section 8.3),
  - `gen_sph_grid` (See Section 9.1) (Deprecated),
- Simulation program `sph_mhd` (See Section 7),
- Data analysis programs,
  - `sph_snapshot` (See Section 8.1),

- sph_dynamobench (See Section 8.4),
- sectioning (See Section 8.5),
- field_to_VTK (See Section 8.6),
- psf_to_VTK (See Section 8.7),
- assemble_sph (See Section 8.8),
- t_ave_sph_mean_square (See Section 8.9.1),

- Utility program to bulid.
  - module_dependency (See Section 8.10),

7 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 paramer file control_MHD, boundary condition data file [boundary_data_name] (optional), and indexing file for spherical harmonics generated by the preprocessing program gen_sph_grid (optional). Data files

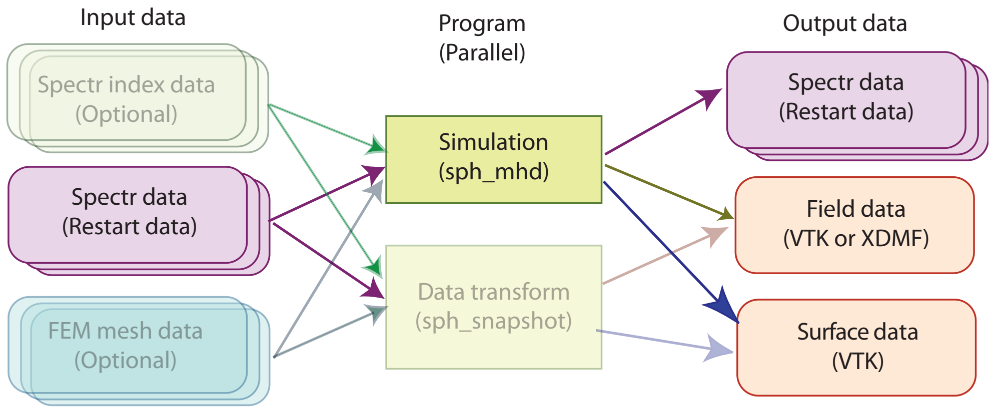


Figure 2: Data flow for the simulation program.

for this program are listed in Table 1. Indexing data for spherical harmonics which starting with [sph_prefix] are obtained by the preprocessing program gen_sph_grid. If these indexing data files do not exist, the spherical harmonics indexing data files are also generated by using information in spherical_shell_ctl block. The boundary condition data file [boundary_data_name] is optionally required if boundary conditions for temperature and composition are not homogenous.

Caution: Calypso can save data files into subdirectories where is defined in control files. However, These directories have to prepare before the simulations because Fortran does not have a feature to make a new directory.

Table 1: List of files for simulation sph_mhd

name	Parallelization	I/O
control_MHD	Serial	Input
[rst_prefix].[step #].[rst_extension]	-	Input/Output
[vol_pwr_prefix]_s.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]_s.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

(Output): Marked files are generated if files do not exist.

Table 2: List of optional files sph_mhd

name	Parallelization	I/O
[boundary_data_name]	Single	Input
[sph_prefix].[rj_extension]	-	Input / (Output)
[sph_prefix].[rlm_extension]	-	Input / (Output)
[sph_prefix].[rtm_extension]	-	Input / (Output)
[sph_prefix].[rtp_extension]	-	Input / (Output)
[sph_prefix].[fem_extension]	-	(Input / Output)

7.1 Control file

Control files for Calypso consists of blocks starting and ending with `begin` and `end`, respectively. Entities with more than one components are defined between `begin array` and `end array` flags. The number of components of an array must be defined at `begin array` line. If blocks to be defined in an external file, the external file name is defined by `file` flag.

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 each item.

Block `MHD_control` (Top block of the control file)

- Block `data_files_def`
  - `num_subdomain_ctl [Num_PE]`
  - `num_smp_ctl [Num_Threads]`
  - `sph_file_prefix [sph_prefix]`
  - `boundary_data_file_name [boundary_data_name]`
  - `restart_file_prefix [rst_prefix])`
  - `field_file_prefix [fld_prefix]`
  - `sph_file_fmt_ctl [sph_format]`
  - `restart_file_fmt_ctl [rst_format]`
  - `field_file_fmt_ctl [fld_format]`
- (File or Block `spherical_shell_ctl [resolution_control]`)
  - (Block `FEM_mesh_ctl` See Section 7.1.1)
  - (Block `num_domain_ctl` See Section 7.1.1)
  - (Block `num_grid_sph` See Section 7.1.1)
- Block `model`
  - Block `phys_values_ctl`
    - Array `nod_value_ctl [Field] [Viz_flag] [Monitor_flag]`
  - Block `time_evolution_ctl`
    - Array `time_evo_ctl [Field]`

- Block boundary_condition
  - Array bc_temperature [Group] [Type] [Value]
  - Array bc_velocity [Group] [Type] [Value]
  - Array bc_composition [Group] [Type] [Value]
  - Array bc_magnetic_field [Group] [Type] [Value]
- Block forces_define
  - Array force_ctl [Force]
- Block dimensionless_ctl
  - Array dimless_ctl [Name] [Value]
- Block coefficients_ctl
  - Block thermal
    - Array coef_4_termal_ctl [Name] [Power]
    - Array coef_4_t_diffuse_ctl [Name] [Power]
    - Array coef_4_heat_source_ctl [Name] [Power]
  - Block momentum
    - Array coef_4_velocity_ctl [Name] [Power]
    - Array coef_4_press_ctl [Name] [Power]
    - Array coef_4_v_diffuse_ctl [Name] [Power]
    - Array coef_4_buoyancy_ctl [Name] [Power]
    - Array coef_4_Coriolis_ctl [Name] [Power]
    - Array coef_4_Lorentz_ctl [Name] [Power]
    - Array coef_4_composit_buoyancy_ctl [Name] [Power]
  - Block induction
    - Array coef_4_magnetic_ctl [Name] [Power]
    - Array coef_4_m_diffuse_ctl [Name] [Power]
    - Array coef_4_induction_ctl [Name] [Power]
  - Block composition
    - Array coef_4_composition_ctl [Name] [Power]
    - Array coef_4_c_diffuse_ctl [Name] [Power]
    - Array coef_4_composition_source_ctl [Name] [Power]
- Block temperature_define
  - ref_temp_ctl [REFERENCE_TEMP]

* Block low_temp_ctl
  * depth [RADIUS]
  * temperature [TEMPERATURE]
* Block high_temp_ctl
  * depth [RADIUS]
  * temperature [TEMPERATURE]

- Block control
  - Block time_step_ctl
    * elapsed_time_ctl [ELAPSED_TIME]
    * i_step_init_ctl [ISTEP_START]
    * i_step_finish_ctl [ISTEP_FINISH]
    * i_step_check_ctl [ISTEP_MONITOR]
    * i_step_rst_ctl [ISTEP_RESTART]
    * i_step_field_ctl [ISTEP_FIELD]
    * i_step_sectioning_ctl [ISTEP_SECTION]
    * i_step_isosurface_ctl [ISTEP_ISOSURFACE]
    * dt_ctl [DELTA_TIME]
    * time_init_ctl [INITIAL_TIME]
  - Block restart_file_ctl
    * rst_ctl [INITIAL_TYPE]
  - Block time_loop_ctl
    * scheme_ctl [EVOLUTION_SCHEME]
    * coef_imp_v_ctl [COEF_INP_U]
    * coef_imp_t_ctl [COEF_INP_T]
    * coef_imp_b_ctl [COEF_INP_B]
    * coef_imp_c_ctl [COEF_INP_C]
    * FFT_library_ctl [FFT_Name]
    * Legendre_trans_loop_ctl [Leg_Loop]
- Block sph_monitor_ctl
  - volume_average_prefix [vol_ave_prefix]

- volume_pwr_spectr_prefix [vol_pwr_prefix]
- volume_pwr_spectr_format [file_format]
- nusselt_number_prefix [nusselt_number_prefix]
- nusselt_number_format [file_format]
- Array volume_spectrum_ctl
  - Block volume_spectrum_ctl
    - volume_average_prefix [vol_ave_prefix]
    - volume_pwr_spectr_prefix [vol_pwr_prefix]
    - volume_pwr_spectr_format [file_format]
    - inner_radius_ctl [radius]
    - outer_radius_ctl [radius]
- Block layered_spectrum_ctl
  - layered_pwr_spectr_prefix [layer_pwr_prefix]
  - volume_pwr_spectr_format [file_format]
  - Array spectr_layer_ctl [Layer #]
- Block gauss_coefficient_ctl
  - gauss_coefs_prefix [gauss_coef_prefix]
  - gauss_coefs_format [file_format]
  - gauss_coefs_radius_ctl [gauss_coef_radius]
  - Array pick_gauss_coefs_ctl [Degree] [Order]
  - Array pick_gauss_coef_degree_ctl [Degree]
  - Array pick_gauss_coef_order_ctl [Order]
- Block pickup_spectr_ctl
  - picked_sph_prefix [picked_sph_prefix]|
  - picked_sph_format [file_format]
  - Array pick_layer_ctl [Layer #]
  - Array pick_sph_spectr_ctl [Degree] [Order]
  - Array pick_sph_degree_ctl [Degree]
  - Array pick_sph_order_ctl [Order]
- Block sph_dipolarity_ctl
  - dipolarity_file_prefix [dipolarity_file_prefix]|

    * dipolarity_file_format [file_format]
    * Array dipolarity_truncation_ctl [Degree]
  - Block mid_equator_monitor_ctl
    * nphi_mid_eq_ctl [Nphi_mid_equator]

- Block visual_control
  - i_step_sectioning_ctl [ISTEP_SECTION]
  - Array cross_section_ctl
    * File or Block cross_section_ctl
      [section_control_file]
      (See section 7.4)
  - i_step_isosurface_ctl [ISTEP_ISOSURFACE]
  - Array isosurface_ctl
    * File or Block isosurface_ctl
      [isosurface_control_file]
      (See section 7.5)

- Block dynamo_vizs_control
  - File or Block zonal_mean_section_ctl
    [zonal_mean_section_control_file]
    (See section 7.4)
  - File or Block zonal_RMS_section_ctl
    [zonal_RMS_section_control_file]
    (See section 7.4)
  - Block crustal_filtering_ctl
    * truncation_degree_ctl [Degree]

7.1.1 Spatial resolution definition block

Geometry of spherical shell, spatial resolution, and parallelization is defined in the block named spherical_shell_ctl. The spherical_shell_ctl block can be included into the file control_MHD or saved into another file. if the spherical_shell_ctl block in the independent file [resolution_control], the file name is defined by

- file spherical_shell_ctl [resolution_control])

The `spherical_shell_ctl` block consists of the following parameters `spherical_shell_ctl`

Block `spherical_shell_ctl`

- (Block `FEM_mesh_ctl`)
  - (`FEM_mesh_output_switch [ON or OFF]`)
- Block `num_domain_ctl`
  - `ordering_set_ctl [ORDERING_SET]`
  - `num_radial_domain_ctl [Ndomain]`
  - `num_horizontal_domain_ctl [Ndomain]`
  - Array `num_domain_sph_grid [Direction] [Ndomain]` (Depricated)
  - Array `num_domain_legendre [Direction] [Ndomain]` (Depricated)
  - Array `num_domain_spectr [Direction] [Ndomain]` (Depricated)
- Block `num_grid_sph`
  - `truncation_level_ctl [Lmax]`
  - `ngrid_meridonal_ctl [Ntheta]`
  - `ngrid_zonal_ctl [Nphi]`
  - `radial_grid_type_ctl` `[explicit, Chebyshev, or equi_distance]`
  - `num_fluid_grid_ctl [Nr_shell]`
  - `fluid_core_size_ctl [Length]`
  - `ICB_to_CMB_ratio_ctl [R_ratio]`
  - `Min_radius_ctl [Rmin]`
  - `Max_radius_ctl [Rmax]`

  - Array `r_layer [Layer #] [Radius]`

– Array boundaries_ctl [Boundary_name] [Layer #]

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.
(see example spherical_shell/with_inner_core)

The external file for resoultion and parallelization information [resolution_control] needs the following contorl blocks:

- Block spherical_shell_ctl
  - Block FEM_mesh_ctl
  - Block num_domain_ctl
  - Block num_grid_sph

Calypso obtains resolution and parallelization information in the following order:

Step 1: If spherical harmonics indexing files [sph_prefix] defined by sph_file_prefix [sph_prefix] are exist, read these files and go to simulation.

Step 2: If files [sph_prefix] does not exist, construct resolution and parallelization information from the parameters in spherical_shell_ctl block.

Step 3: If the parameter sph_file_prefix [sph_prefix] is defined, the spherical harmonics indexing files [sph_prefix] are written (not necessary).

Various data format can be chosen for the spherical harmonic indexing files [sph_prefix] (see Section 9.1).

7.1.2 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

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

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.

7.1.3 How to define spatial resolution and parallelization?

Calypso uses spherical harmonics expansion method and in horizontal discretization and finite difference methods in the radial direction. In the spherical harmonics expansion methods, nonlinear terms are solved in the grid space while time integration and diffusion terms are solved in the spectrum space. We need to set truncation degree l_{max} of the spherical harmonics and number of grids in the three direction (N_r, N_θ, N_ϕ) in the preprocessing program. The following condition is required (or recommended) for l_{max} and (N_r, N_θ, N_ϕ). l_{max} is defined by `truncation_level_ctl`, and N_r for the fluid shell (outer core) is defined by `num_fluid_grid_ctl`. N_θ and N_ϕ is defined by `ngrid_meridonal_ctl` and `ngrid_zonal_ctl`, respectively.

- $N_\phi = 2N_\theta$.
- N_θ must be more than $l_{max} + 1$, but
- To eliminate aliasing in the spherical transform, $N_\theta \geq 1.5\,(l_{max} + 1)$ is highly recommended.
- N_ϕ should consists of products among power of 2, power of 3, and power of 5.

Calypso is parallelized 2-dimensionally and direction of the parallelization is changed in the operations in the spherical transform (See Figure 3). Two dimensional parallelization delivers many parallelize configuration. Here is the approach how to find the best configuration:

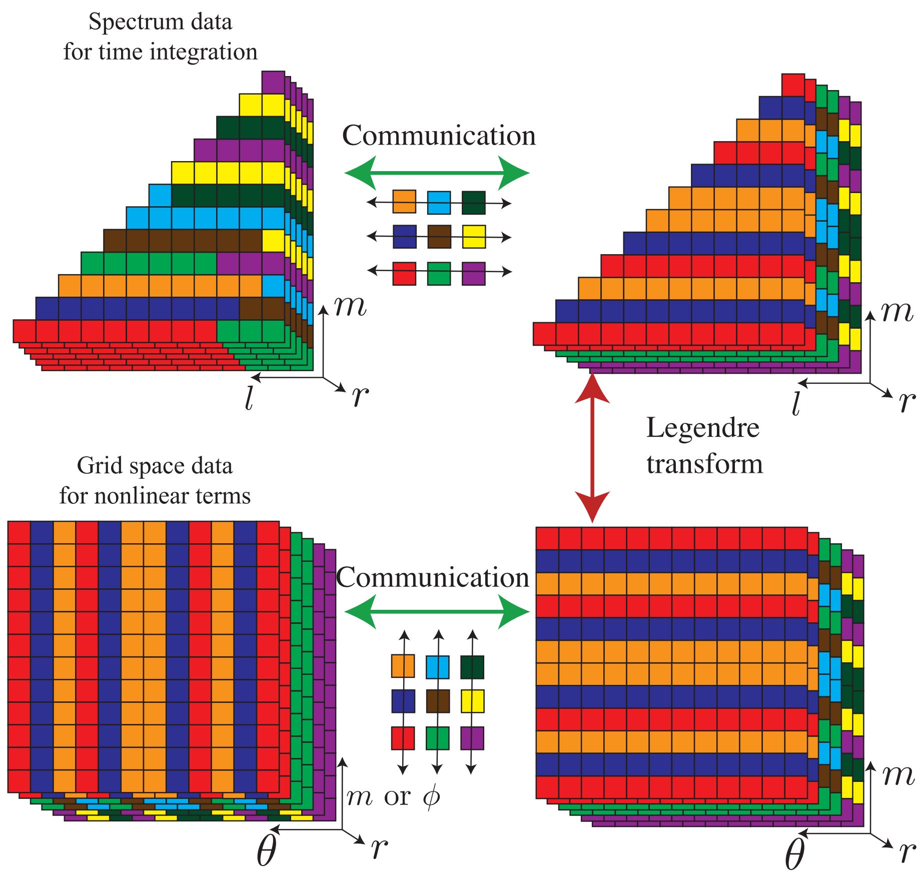


Figure 3: Parallelization and data communication in Calypso in the case using 9 (3x3) processors. Data are decomposed in radial and meridional direction for nonlinear term evaluations, decomposed in radial and harmonic order for Legendre transform, and decomposed in spherical harmonics for linear calculations.

- Maximum parallelization level in horizontal direction is $(l_{max}+1)/2$, and N_r+1 is the maximum level in radial direction.
- Decompose number of radial points N_r+1 and truncation degree $(l_{max}+1)/2$ into prime numbers.
- Decide number of MPI processes from the prime numbers.
- Choose the number of decomposition in the radial and horizontal direction as close as possible.

Here is an example for the case with $(N_r, l_{max}) = (89, 95)$. The maximum number of parallelization is $90 \times 48 = 4320$ processes. N_r+1 and $(l_{max}+1)/2$ can be decomposed into $90 = 2 \times 3^2 \times 5$ and $48 = 2^4 \times 3$. Now, if 160 processes run is intended, $160 = 10 \times 16$ is the closest number of decompositions. Comparing with the prime numbers of the spatial resolution, radial and horizontal decomposition will be 10 and 16, respectively.

7.1.4 Radial grid data

The 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.

7.1.5 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 4. 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.

7.2 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]. Data format is defined by [restart_file_fmt_ctl] as shown in Table 3.

Table 3: data format flag [restart_file_fmt_ctl] and extensions for the restart file. An example of data size and output time is also listed.

File format	[sph_file_fmt_ctl]	extension	size (MByte)	time (sec)
Parallel	ascii	[#].fld	1,286	7.10
	binary	[#].flb	410	2.36
	gzip	[#].fld.gz	418	4.65
	bin_gz	[#].flb.gz	336	2.84
Merged	merged	.fld	1,312	50.2
	merged_bin	.flb	413	2.34
	merged_gz	.fld.gz	455	8.60
	merged_bin_gz	.flb.gz	340	3.57

```
#
#  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 4: An example of boundary condition file.

7.3 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 8.11. Field data also output merged ASCII or binary format including compression using zlib. These original formats have smaller file size than VTK format because of excluding grid information. Program `field_to_VTK` generates VTK file from FEM mesh data and field data.

Table 4: Checked visualization application

Control flag	`fld_format`	Application
`VTK`	Distributed VTK	ParaView
`single_VTK`	Merged VTK	ParaView, VisIt, or Mayavi
`VTK_gzip`	Compressed Distributed VTK	ParaView after expanding by `gzip`
`single_VTK_gz`	Compressed Merged VTK	ParaView, VisIt or Mayavi after expanding by `gzip`
`single_HDF5`	XDMF	ParaView, VisIt
`ascii`	Distributed ASCII	-
`binary`	Distributed binary	-
`gzip`	Distributed compressed ASCII	-
`bin_gz`	Distributed compressed binary	-
`merged`	Merged ASCII	-
`merged_bin`	Merged binary	-
`merged_gzip`	Merged compressed ASCII	-
`merged_bin_gz`	Merged compressed binary	-

More informations about ParaView is in https://www.paraview.org.
More informations about VisIt is in https://wci.llnl.gov/codes/visit/.
More informations about Mayavi is in http://mayavi.sourceforge.net/.

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

- Advantage
  - Faster output
  - 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

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

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

Calypso field data Calypso field data is based on the spectr data for restarting. The data is simply replaced from spherical harmonics coefficients to each component of field data in the cartesian coordinate. The file format flag `[field_file_fmt_ctl]` and corresponding extensiton are showw in Table 8.

Table 8: Data format flag [field_file_fmt_ctl] and extensions for the field file. An example of data size and output time is also listed.

File format	[field_file_fmt_ctl]	extension	size (GByte)	time (sec)
Parallel	ascii	[#].fld	9.31	61.1
	binary	[#].flb	2.93	39.4
	gzip	[#].fld.gz	3.25	40.6
	bin_gz	[#].flb.gz	2.78	39.6
	VTK	[#].vtk	12.4	1030.5
	VTK_gz	[#].vtk.gz	3.36	40.8
Merged	merged	.fld	9.53	39.1
	merged_bin	.flb	3.58	41.7
	merged_gz	.fld.gz	2.99	39.4
	merged_bin_gz	.flb.gz	2.84	39.2
	merged_VTK	.vtk	11.8	39.3
	merged_VTK_gz	.vtk.gz	3.13	39.0

7.4 Cross section data (Parallel Surfacing module)

Calypso can output cross section data for visualization with finer time increment than the whole domain data. The cross section data consist of triangle patches with VTK format, then data can be visualized by Paraview like as the whole field data. This cross sectioning module can output arbitrary quadrature surface, but plane, sphere, and cylindrical section would be useful for the geodynamo simulations.

To output cross sectioning, increment of the surface output data should be defined by i_step_sectioning_ctl in time_step_ctl block. And, array block cross_section_ctl in visual_control section is required to define cross sections. Each cross_section_ctl block defines one cross section. Each cross section can also define by an external file by specifying external file name with file label. The sections shown in Table 9 are supported in the sectioning module. These surfaces are defined in the Cartesian coordinate. The easiest approrach is using sections defined by quadrature function with ten coefficients from a to k in the control array coefs_ctl.

A plane surface is defined by a normal vector (a, b, c) and one point including the surface (x_0, y_0, z_0) in arrays normal_vector and center_position, respectively.

A sphere surface is defined by the position of the center (x_0, y_0, z_0) and radius r in array center_position and radius, respectively.

Table 9: Supported cross sections

Surface type	equation
Quadrature surface	$ax^2 + by^2 + cz^2 + dyz + ezx + fxy + gx + hy + jz + k = 0$
Plane surface	$a(x - x_0) + b(y - y_0) + c(z - z_0) = 0$
Sphere	$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2$
Ellipsoid	$\left(\frac{x - x_0}{a}\right)^2 + \left(\frac{y - y_0}{b}\right)^2 + \left(\frac{z - z_0}{c}\right)^2 = 1$

An Ellipsoid surface is defined the position of the center (x_0, y_0, z_0) and length of the each axis (a, b, c) in arrays `center_position` and `axial_length`, respectively. If one component of the `axial_length` is set to 0, surfacing module generate a Ellipsoidal tube along with the axis where `axial_length` is set to 0.

Area for visualization can be defined by array `chosen_ele_grp_ctl` by choosing `outer_core`, `inner_core`, and `all`. Fields to display is defined in array `output_field`. In array `output_field`, field type in Table 10 needs to defined. The same field can be defined more than once in array `output_field` to output vector field in Cartesian coordinate and radial component, for example.

Table 10: List of field type for cross sectioning and isosurface module

Definition	Field type
`scalar`	scalar field
`vector`	Cartesian vector field
`x`	x-component
`y`	y-component
`z`	z-component
`radial`	radial (r-) component
`theta`	θ-component
`phi`	ϕ-component
`cylinder_r`	cylindrical radial (s-) component
`magnitude`	magnitude of vector

Control data The format of the control file or block for cross sections is described below. The detail of each block is described in section A. cross_section_ctl block can be read from an external file.

To define the external file name, as file cross_section_ctl [file name] in control_MHD or control_snapshot.

Block cross_section_ctl (Top level for sectioning)

- section_file_prefix [section_prefix]
- psf_output_type [file_format]
- Block surface_define
  - section_method [METHOD]
  - Array coefs_ctl [TERM] [COEFFICIENT]
  - radius [SIZE]
  - Array normal_vector [DIRECTION] [COMPONENT]
  - Array axial_length [DIRECTION] [COMPONENT]
  - Array center_position [DIRECTION] [COMPONENT]
  - Array section_area_ctl [AREA_NAME]
- output_field_define
  - Array output_field [FIELD] [COMPONENT]

Output data format of sectioning module Sectioning data are written with VTK format and VTK data compressed by zlib. Field data also output by binary format and binary compressed by zlib. The list of data format and control flag for psf_output_type are listed in Table 11. In the binary data format, position data and field data are saved independently not to write the grid data for each output step. Program psf_to_VTK generates VTK file from the binary section data. The output data format is defined by psf_output_type. Because the field data is written by using Cartesian coordinate (x, y, z) system, (x, y, z) components in ParaView corresponds to the spherical components (r, θ, ϕ) or cylindrical componennts (s, ϕ, z) if sectioning data is writtein the spherical or cylindrical components. Consequently, ParaView can not draw griph or field lines for these spherical or cylindrical vectors.

Table 11: Data format and an example of data size and output time for sectioning data

File format	[fld_format]	extension	size (MByte)	time (sec)
VTK	VTK	.vtk	29.1	2.88
Compressed VTK	VTK_gzip	.vtk.gz	6.04	1.29
Binary	PSF	0.sgd (grid) .sdt (field)	5.30 4.13	 2.59
Compressed binary	PSF_gzip	.0.sgd.gz (grid) .sdt.gz (field)	1.17 3.94	 1.21

7.5 Isosurface data

Calypso can also output isosurface data for visualization. Generally, data size of the isosurface is much larger than the sectioning data. The isosurface data is also written as a unstructured grid data with VTK format. The isosurface also consists of triangle patches.

To output cross sectioning, increment of the surface output data should be defined by i_step_isosurface_ctl in time_step_ctl block. And, array block isosurface_ctl in visual_control section is required to define cross sections. Each isosurface_ctl block defines one cross section. Each cross section can also define by an external file by specifying external file name with file label.

Control data The format of the control file or block for isosurfaces is described below. The detail of each block is described in section A. isosurface_ctl block can be read from an external file. To define the external file name, as file isosurface_ctl [file name] in control_MHD or control_snapshot.

Block isosurface_ctl (Top lebel of the control data)

- isosurface_file_prefix [file_prefix]
- iso_output_type [file_format]
- Block isosurf_define
  - isosurf_field [FIELD]
  - isosurf_component [COMPONENT]
  - isosurf_value [VALUE]

    - Array isosurf_area_ctl [AREA_NAME]

- Block field_on_isosurf
    - result_type [TYPE]
    - result_value [VALUE]
    - Array output_field [FIELD] [COMPONENT]

Output data format of isosurface module Isosurface data are written with VTK format and VTK data compressed by zlib. Field data also output by binary format and binary compressed by zlib. The list of data format and control flag for iso_output_type are listed in Table 12. Like as sectioning data, program psf_to_VTK generates VTK file from the binary section data. The output data format is defined by iso_output_type. Because the field data is written by using Cartesian coordinate (x, y, z) system, (x, y, z) components in ParaView corresponds to the spherical components (r, θ, ϕ) or cylindrical componennts (s, ϕ, z) if sectioning data is writtein the spherical or cylindrical components. Consequently, ParaView can not draw griph or field lines for these spherical or cylindrical vectors.

Table 12: Data format and an example of data size and output time for isosurface data

File format	[fld_format]	extension	Size (MByte)	Time (sec)
VTK	VTK	.vtk	172	1.88
Compressed VTK	VTK_gzip	.vtk.gz	38.8	2.92
Binary	ISO	.sfm	55.7	2.22
Compressed binary	ISO_gzip	.sfm.gz	35.0	1.73

7.6 Time history data for monitoring

Calypso also generate various time history outputs data.These files are written by the SCII format or compressed format by using zlib library. These compressed files can be expanded by gzip commnd. If the time history 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.

7.6.1 Layered spectrum data

Spectrum data for the each radial position are written by defining `layered_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. The radial points for output is listed in the array `spectr_layer_ctl`. If `spectr_layer_ctl` is not defined, mean square data at **all** radial levels will be written. See example of dynamo benchmark case 2. Header of the layered spectrum data file consists of

`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 8:` Number of components for each field

Field labels indicates as following:

`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]`

In vector fields, the kinetic energy $u^2/2$ and magnetic energy $B^2/2$ are calculated instead of mean square amplitude for the velocity $\boldsymbol{u}$ and magnetic field $\boldsymbol{B}$. Headers on the first lines indicate following data. The following mean square data is generated:

`[layer_pwr_prefix]_s.dat[.gz]` Surface average of mean square amplitude of the fields. the mean square for the scalar field f is evluated from its spherical harmonic coefficients by

$$< f^2(r) >= \sum_l \sum_m \frac{1}{2l+1} \left(f_l^m(r) \right)^2 . \tag{31}$$

And, the mean square of the vector field $\boldsymbol{A}$ evaluated by

$$\begin{aligned} < \boldsymbol{A}_S^2(r) > &= \sum_l \sum_m \frac{1}{2l+1} \left[\left(\frac{l\,(l+1)}{r^2} A_{Sl}^{\,m}(r) - \frac{\partial \phi_l^m}{\partial r} \right)^2 \right. \\ & \left. + \frac{l\,(l+1)}{r^2} \left(\frac{\partial A_{Sl}^{\,m}}{\partial r} - \phi_l^m \right)^2 \right], \end{aligned} \tag{32}$$

and

$$< \boldsymbol{A}_T^2(r) > \; = \; \sum_l \sum_m \frac{l\,(l+1)}{2l+1} \frac{1}{r^2} \left(A_{Tl}^{\,m}(r) \right)^2 . \tag{33}$$

Finally, the total mean square amplitude $< \boldsymbol{A}^2 >$ is obtained by

$$< \boldsymbol{A}^2(r) > \; = \; < \boldsymbol{A}_S^2(r) > + < \boldsymbol{A}_T^2(r) > . \tag{34}$$

`[layer_pwr_prefix]_l.dat[.gz]` 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

$$< {f_l}^2(r,l) >= \sum_{m=-l}^{l} \frac{1}{2l+1} \left(f_l^m(r) \right)^2 . \tag{35}$$

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

$$\begin{aligned} < \boldsymbol{A}_{Sl}^2(r,l) > &= \sum_{m=-l}^{l} \frac{1}{2l+1} \left[\left(\frac{l\,(l+1)}{r^2} A_{Sl}^{\,m}(r) - \frac{\partial \phi_l^m}{\partial r} \right)^2 \right. \\ & \left. + \frac{l\,(l+1)}{r^2} \left(\frac{\partial A_{Sl}^{\,m}}{\partial r} - \phi_l^m \right)^2 \right], \end{aligned} \tag{36}$$

and

$$< \boldsymbol{A}_{Tl}^2(r,l) > \;=\; \sum_{m=-l}^{l} \frac{l\,(l+1)}{2l+1}\frac{1}{r^2}\left(A_{Tl}^{m}(r)\right)^2 . \qquad (37)$$

Finally, the total mean square amplitude $< \boldsymbol{A}^2 >$ is obtained by

$$< \boldsymbol{A}_{l}^2(r,l) > \;=\; < \boldsymbol{A}_{Sl}^2(r,l) > + < \boldsymbol{A}_{Tl}^2(r,l) > . \qquad (38)$$

`[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

$$< f_m^2(r,m) >= \sum_{l=m}^{L_{max}} \frac{1}{2l+1}\left[\left(f_l^{m}(r)\right)^2 + \left(f_l^{-m}(r)\right)^2\right] \qquad (39)$$

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

$$\begin{aligned} < \boldsymbol{A}_{Sm}^2(r,m) > \;=\; \sum_{l=m}^{L_{max}} \frac{1}{2l+1} \Bigg[& \left(\frac{l\,(l+1)}{r^2}A_{Sl}^{m}(r) - \frac{\partial \phi_l^m}{\partial r}\right)^2 \\ & + \left(\frac{l\,(l+1)}{r^2}A_{Sl}^{-m}(r) - \frac{\partial \phi_l^{-m}}{\partial r}\right)^2 \\ & + \frac{l\,(l+1)}{r^2}\left(\frac{\partial A_{Sl}^{m}}{\partial r} - \phi_l^{m}\right)^2 \\ & + \frac{l\,(l+1)}{r^2}\left(\frac{\partial A_{Sl}^{-m}}{\partial r} - \phi_l^{-m}\right)^2 \Bigg], \end{aligned} \qquad (40)$$

and

$$< \boldsymbol{A}_{Tm}^2(r,m) > \;=\; \sum_{l=m}^{L_{max}} \frac{l\,(l+1)}{2l+1}\frac{1}{r^2}\left[\left(A_{Tl}^{m}(r)\right)^2 + \left(A_{Tl}^{-m}(r)\right)^2\right] . \qquad (41)$$

Finally, the total mean square amplitude $< \boldsymbol{A}^2 >$ is obtained by

$$< \boldsymbol{A}_{m}^2(r,m) > \;=\; < \boldsymbol{A}_{Sm}^2(r,m) > + < \boldsymbol{A}_{Tm}^2(r,m) > . \qquad (42)$$

[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

$$< f_n^2(r,n) >= \sum_{m=n}^{L_{max}} \frac{1}{2l+1} \left[\left(f_l^{l-n}(r) \right)^2 + \left(f_l^{-l+n}(r) \right)^2 \right] \tag{43}$$

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

$$\begin{aligned} < \boldsymbol{A}_{Sn}^2(r,n) > \;=\; & \sum_{m=n}^{L_{max}} \frac{1}{2l+1} \left[\left(\frac{l\,(l+1)}{r^2} A_{Sl}^{\;l-n}(r) - \frac{\partial \phi_l^{l-n}}{\partial r} \right)^2 \right. \\ & + \left(\frac{l\,(l+1)}{r^2} A_{Sl}^{-l+n}(r) - \frac{\partial \phi_l^{-l+n}}{\partial r} \right)^2 \\ & + \frac{l\,(l+1)}{r^2} \left(\frac{\partial A_{Sl}^{\;l-n}}{\partial r} - \phi_l^{l-n} \right)^2 \\ & \left. + \frac{l\,(l+1)}{r^2} \left(\frac{\partial A_{Sl}^{-l+n}}{\partial r} - \phi_l^{-l+n} \right)^2 \right], \end{aligned} \tag{44}$$

and

$$< \boldsymbol{A}_{Tn}^2(r,n) > \;=\; \sum_{m=n}^{L_{max}} \frac{l\,(l+1)}{2l+1} \frac{1}{r^2} \left[\left(A_{Tl}^{\;l-n}(r) \right)^2 + \left(A_{Tl}^{-l+n}(r) \right)^2 \right]. \tag{45}$$

Finally, the total mean square amplitude $< \boldsymbol{A}^2 >$ is obtained by

$$< \boldsymbol{A}_n^{\,2}(r,n) > \;=\; < \boldsymbol{A}_{Sn}^2(r,n) > + < \boldsymbol{A}_{Tn}^2(r,n) > \tag{46}$$

[layer_pwr_prefix]_m0.dat Surace average of mean square amplitude of the axisymmetric components of fields as a function of radial grid id k. The zonal wave number is referred in this spectrum data. For scalar field, the spectrum is

$$< f^2(r) >_{axis} = \sum_{l=0}^{L_{max}} \frac{1}{2l+1} \left(f_l^0(r) \right)^2 \tag{47}$$

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

$$< \boldsymbol{A}_S^2(r) >_{axis} \;=\; \sum_{l=0}^{L_{max}} \frac{1}{2l+1} \left[\left(\frac{l\,(l+1)}{r^2} A_{Sl}^{\,0}(r) - \frac{\partial \phi_l^0}{\partial r} \right)^2 \right.$$

$$+\frac{l\left(l+1\right)}{r^{2}}\left(\frac{\partial A_{Sl}^{0}}{\partial r}-\phi_{l}^{0}\right)^{2}\Bigg], \tag{48}$$

and

$$< \boldsymbol{A}_T^2(r) >_{axis} \;=\; \sum_{l=0}^{L_{max}} \frac{l\left(l+1\right)}{2l+1}\frac{1}{r^2}\left(A_{Sl}^{0}(r)\right)^2. \tag{49}$$

Finally, the total mean square amplitude $< \boldsymbol{A}^2 >$ is obtained by

$$< \boldsymbol{A}^2(r) >_{axis} \;=\; < \boldsymbol{A}_S^2(r) >_{axis} + < \boldsymbol{A}_T^2(r) >_{axis}. \tag{50}$$

7.6.2 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. The data is written in the file `[vol_pwr_prefix]_s.dat[.gz]` or `sph_pwr_volume_s.dat` if `[vol_pwr_prefix]` is not defined in the control file. The mean square amplitude ifor a scalar field f is defined by

$$< f^2 >= \frac{4\pi}{V}\int < f^2(r) > r^2 dr. \tag{51}$$

For the vector field $\boldsymbol{A}$, the mean square of the potential component $\boldsymbol{A}_S$ is included in the mean square of the poloidal component in the data if the vector field is not a solenoidal field. Consequently, $\boldsymbol{A}_S$ and mean square of the toroidal component $\boldsymbol{A}_T$ are evaluated by

$$\begin{aligned} < \boldsymbol{A}_S^2 > &= \frac{4\pi}{V}\int < \boldsymbol{A}_S^2(r) > r^2 dr, \\ < \boldsymbol{A}_T^2 > &= \frac{4\pi}{V}\int < \boldsymbol{A}_T^2(r) > r^2 dr, \end{aligned} \tag{52}$$

and

$$< \boldsymbol{A}^2 > \;=\; \frac{4\pi}{V}\int < \boldsymbol{A}^2(r) > r^2 dr. \tag{53}$$

The header in the first 12 lines is the following.

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

`line 4:` Radial layer ID for ICB and CMB `line 6:` Radial layer ID and radius for the inner boundary of integration

`line 8:` Radial layer ID and radius for the outer boundary of integration

`line 10:` Number of field of data, total number of components

`line 11:` Number of components for each field

The following is an example of the beginning of the data file:

```
Radial_layers, Truncation
             112             159
ICB_id, CMB_id
               1             112
Lower_boundary_ID, Lower_boundary_radius
               1   5.384615384615384E-001
Upper_boundary_ID, Upper_boundary_radius
             112   1.538461538461538E+000
Number_of_fields, Number_of_components
               8              16
    3    1    3    1    3    3    1    1
t_step    time    K_ene_pol    K_ene_tor    K_ene    temperature    vorticity_pol
    vorticity_tor    vorticity    pressure    M_ene_pol    M_ene_tor    M_ene
    current_density_pol    current_density_tor    current_density    buoyancy_flux
    Lorentz_work
             100  2.000000000000000E-004  3.08485175580558E+001  2.97052514317492E-001
  3.11455700723732E+001  8.37549401638792E-002  3.83545285358558E+001
  4.45268846884004E+003  4.49104299737589E+003  2.20356357802801E+001
  2.88810032145648E-006  1.76927464999397E-006  4.65737497145044E-006
  4.45474362010442E-005  3.76310585686982E-005  8.21784947697424E-005
  1.47655914519649E+013  1.52720673275983E-003
             200  3.99999999999999E-004  1.21246529508559E+002  4.60310202818333E+000
  1.25849631536743E+002  8.38032562478558E-002  5.26153720206686E+002
  1.51087784591356E+004  1.56349321793422E+004  4.32047730989212E+001
  2.88721662147718E-006  1.76772446160287E-006  4.65494108308005E-006
  4.49318945325754E-005  3.83578033623782E-005  8.32896978949536E-005
  5.86224313684072E+013  6.13459599464992E-003
```

7.6.3 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.[gz]` with the same format as the men square amplitude data in Subsection 7.6.2. The volume average of scalar field f is evaluated by the 0-th degree of the spherical harmonics coefficients f_0^0 as

$$< f >= \frac{4\pi}{V} \int f_0^0(r) r^2 dr. \tag{54}$$

The average of the toroidal component of the vector field $< \boldsymbol{A}_T >$ is always zero, and the average of poloidal component of the solenoidal vector $< \boldsymbol{A}_S >$ is also zero. If the vector is non-soleinoidal, 0-th degree of the potential field contributes the average radial vector as

$$< \boldsymbol{A} > \;=\; < \boldsymbol{A}_S >= -\frac{4\pi}{V}\int \frac{\partial \phi_0^0}{\partial r} r^2 dr. \tag{55}$$

If you need the sphere average data for specific radial point, you can use picked spectrum data output in pickup_spectr_ctl using $l = m = 0$ at specific radius.

7.6.4 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 header 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 in each step.

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

$$< f_l^2(l) >= \frac{4\pi}{V}\int < f_l^2(r,l) > r^2 dr. \tag{56}$$

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

$$\begin{aligned} < A_{Sl}^2 > &= \frac{4\pi}{V}\int < A_{Sl}^2(r,l) > r^2 dr, \\ < A_{Tl}^2 > &= \frac{4\pi}{V}\int < A_{Tl}^2(r,l) > r^2 dr, \end{aligned} \tag{57}$$

and

$$< {A_l}^2 > \;=\; \frac{4\pi}{V}\int < {A_l}^2(r,l) > r^2 dr, \tag{58}$$

`[vol_pwr_prefix]_m.dat[.gz]` 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

$$< f_m^2(m) >= \frac{4\pi}{V}\int < f_m^2(r,m) > r^2 dr. \tag{59}$$

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

$$
\begin{aligned}
< A_{Sm}^2 > &= \frac{4\pi}{V} \int < A_{Sm}^2(r,m) > r^2 dr, \\
< A_{Tm}^2 > &= \frac{4\pi}{V} \int < A_{Tm}^2(r,m) > r^2 dr,
\end{aligned} \tag{60}
$$

and

$$
< {A_m}^2 > = \frac{4\pi}{V} \int < {A_m}^2(r,m) > r^2 dr, \tag{61}
$$

`[vol_pwr_prefix]_lm.dat[.gz]` 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

$$
< f_n^2(n) >= \frac{4\pi}{V} \int < f_n^2(r,n) > r^2 dr. \tag{62}
$$

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

$$
\begin{aligned}
< A_{Sn}^2 > &= \frac{4\pi}{V} \int < A_{Sn}^2(r,n) > r^2 dr, \\
< A_{Tn}^2 > &= \frac{4\pi}{V} \int < A_{Tn}^2(r,n) > r^2 dr,
\end{aligned} \tag{63}
$$

and

$$
< {A_n}^2 > = \frac{4\pi}{V} \int < {A_n}^2(r,n) > r^2 dr, \tag{64}
$$

`[vol_pwr_prefix]_m0.dat[.gz]` Volume average of mean square amplitude of the axisymmetric components is stored. The mean square of the axisymmetric scalar is defined by

$$
< f_{m0}^2 > = \frac{1}{V} \sum_{l=0}^{L_{max}} \left[N_l \int \left(f_l^0(r) \right)^2 r^2 dr \right]. \tag{65}
$$

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

$$
< A_{Sm0}^2 > = \frac{1}{V} \sum_{l=0}^{L_{max}} N_l \int \left[\left(\frac{l\,(l+1)}{r^2} A_{Sl}^0(r) - \frac{\partial \phi_l^0}{\partial r} \right)^2 \right.
$$

$$+\frac{l\,(l+1)}{r^2}\left(\frac{\partial A_{Sl}^{0}}{\partial r}-\phi_l^0\right)^2\Bigg] r^2 dr, \quad (66)$$

$$< A_{Tm0}^{2} > \;=\; \frac{1}{V}\sum_{l=0}^{L_{max}}(l+1)\,N_l\int\left(A_{Tl}^{0}(r)\right)^2 dr. \quad (67)$$

This data file is used toe same data format as the volume mean square data in Subsection 7.6.2.

Default volume averge and mean square data is integrated over the fluid domain. Calypso also can output volume integration data between manually defined inner and outer boundary. Each volume integration parameter is defined in array block `volume_spectrum_ctl`. For each integration, the inner and outer boundary radii are set by `inner_radius_ctl` and `outer_radius_ctl`, respectively.

7.6.5 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

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

The header format is the same as the volume mean square data, but each item is used for as

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

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

`line 6:` Not used

`line 8:` Radius for the reference radius r_e

`line 10:` Nunmber of Gauss coeffifients

`line 11:` Number of components (All number are 1)

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.

7.6.6 Spectrum monitor data

`[picked_sph_prefix]_l[degree]_m[order].dat`

This outputs of spherical harmonics coefficients at specified spherical harmonics modes and radial points in text files. Data for each spherical harmonic mode are saved in the files named `[picked_sph_prefix]_l[degree]_m[order].dat`. Consequently, the file prefix `[picked_sph_prefix]` is recommended to be defined including subdirectory name to save under a subdirectory. Spectrum data marked `[Monitor_On]` are written in one line for each 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.

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

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

`[field_name]_tor :` F_{Tl}^{m}

7.6.7 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.

The header format is the same as the volume mean square data, but number of components are 2 for the Nusselt number for the inner and outer boundaries.

7.6.8 Dipolrity data [dipolarity_file_prefix].dat

The dipolarity f_{dip} is evaluated the poloidal magnetic field at CMB is written for each step in one line. The dipolarity represents the relative strength of the axial dipole magnetic field, which is defined by the ration of the magnetic energy of the dipole component to the total magnetic energy at the CMB as

$$f_{\mathrm{dip}} = \left(\frac{E_{\mathrm{B}\,1}^{\,0}(r = r_o)}{\sum_{l=1}^{L_{\mathrm{dip}}} \sum_{m=-l}^{l} E_{\mathrm{B}\,l}^{\,m}(r = r_o)} \right)^{1/2}. \tag{68}$$

The magnetic energy at the CMB, $E_{\mathrm{B}}(r = r_o)$, is calculated as

$$E_{\mathrm{B}\,l}^{\,m}(r = r_o) = \frac{1}{2S_o} \int_{S_o} \left[\left(\boldsymbol{B}_l^m \right)^2 + \left(\boldsymbol{B}_l^{-m} \right)^2 \right] dS, \tag{69}$$

where $S_o = 4\pi r_o^2$ is the surface area of the outer core.

7.6.9 Typical length scale data [typical_scale_prefix].dat

The typical length scale is evaluated from kinetic and magnetic energy spectra as a wave number for the spherical harmonic degree l, order m, and difference between harmonic degree and order $n = l - m$. Each value is evaluated by

$$k_l = \frac{\sum_{l=1}^{L_{max}} l < \boldsymbol{A}_l^2 >}{< \boldsymbol{A}^2 >}, \tag{70}$$

$$k_m = \frac{\sum_{m=0}^{L_{max}} m < \boldsymbol{A}_m^2 >}{< \boldsymbol{A}^2 >}, \tag{71}$$

$$k_{l-m} = \frac{\sum_{n=0}^{L_{max}} (l - m) < \boldsymbol{A}_n^2 >}{< \boldsymbol{A}^2 >}, \tag{72}$$

where $\boldsymbol{A}$ is the velocity $\boldsymbol{u}$ or magnetic field $\boldsymbol{B}$, and volume mean squares $< \boldsymbol{A}^2 >$, $< \boldsymbol{A}_l^2 >$, $< \boldsymbol{A}_m^2 >$, and $< \boldsymbol{A}_n^2 >$, are defined by equations (53), (58), (61) , and (64), respectively.

The header format is the same as the volume mean square data, and the field name is defined by truncation_L_{dip}. The labels of each length scales are in Table 13.

Table 13: Data names for typical length scles.

	Velocity	Magnetic field
k_l	lscale_flow_degree	lscale_magnetic_degree
k_m	lscale_flow_order	lscale_magnetic_order
k_{l-m}	lscale_flow_diff_lm	lscale_magnetic_diff_lm

8 Utility programs

Calypso includes some utility programs. These programs are useful to data analysis and debugging the simulation programs.

8.1 Data transform program (sph_snapshot)

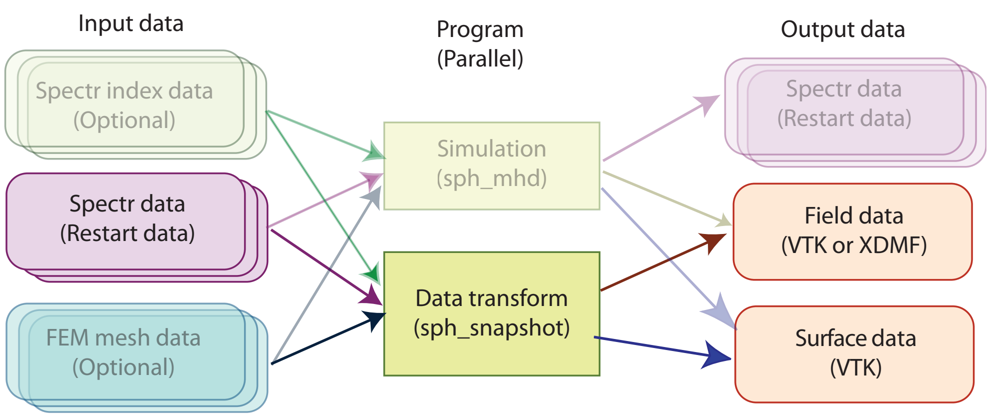


Figure 5: Data flow for data transform program.

Simulation program outputs spectrum data as a whole field data. This program is made from simulation program by replacing from time integration routines to restart data input routine. Consequently, Input/Output files in Table 1 are the same for sph_snapshot, except for the required input restart data [rst_prefix].[step #].[rst_extension]. This program requires control file control_snapshot insteacd of control_mhd. 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].

We recommend to output cross section data at $y = 0$ by using sectioning module (see 7.4) for zonal mean snapshot program sph_zm_snapshot to reduce data size.

8.2 Initial field generation program (sph_initial_field)

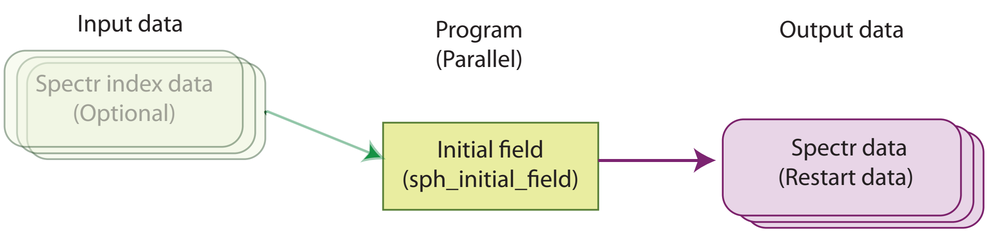


Figure 6: Data flow for initial field generation program.

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. Spherical harmonics indexing data files are also generated by using information in spherical_shell_ctl block if these indexing data files do not exist. 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 14. This program generates the spectrum data files [rst_prefix].[step #].[rst_extension]. To use generated initial data file, please set [ISTEP_START] to be 0 and [INITIAL_TYPE] to be start_from_rst_file.

8.2.1 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.

Table 14: List of files for simulation sph_initial_field

name	Parallelization	I/O
control_MHD	Serial	Input
[sph_prefix].[rj_extension]	-	Input/(Output)
[sph_prefix].[rlm_extension]	-	Input/(Output)
[sph_prefix].[rtm_extension]	-	Input/(Output)
[sph_prefix].[rtp_extension]	-	Input/(Output)
[rst_prefix].[step #].[rst_extension]	-	Input/Output

(Output): Marked files are generated if files do not exist.

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.

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 15.

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

Table 15: 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	-	-

`r = radius_1d_rj_r(k)`. The subroutines to define initial temperature for the dynamo benchmark Case 1 is shown below as an example.

After updating the source code, the program `sph_initial_field` needs to be updated. To update the program, move to the work directory `[CALYPSO_HOME]/work` and run make command as

```
% cd \verb|[CALYPSO_HOME]/work|
% make
```

Then, the program `sph_initial_field` and `sph_add_initial_field` are updated.

```
!
      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
!
```

8.3 Initial field modification program (sph_add_initial_field)

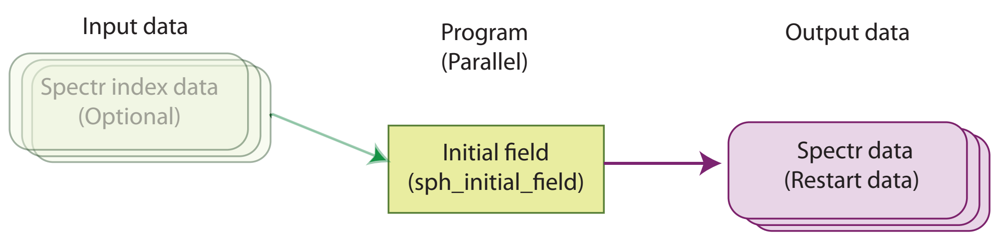


Figure 7: Data flow for initial field modification program.

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 16. This program generates the spectrum data files `[rst_prefix].[step#].[rst_extension]`. To use

Table 16: List of files for simulation sph_add_initial_field

name	Parallelization	I/O
control_MHD	Serial	Input
[sph_prefix].[rj_extension]	-	Input / (Output)
[sph_prefix].[rlm_extension]	-	Input / (Output)
[sph_prefix].[rtm_extension]	-	Input / (Output)
[sph_prefix].[rtp_extension]	-	Input / (Output)
[rst_prefix].[step #].[rst_extension]	-	Input/Output

(Output): Marked files are generated if files do not exist.

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.

This program also uses the module file const_sph_initial_spectr.f90 to define the initial field. The initial fields are defined as following the previous section 8.2.1. After updating the source code, the program sph_initial_field needs to be updated. After modifying const_sph_initial_spectr.f90, the program is build by make command in the work directory [CALYPSO_HOME]/work.

8.4 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.

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 $\frac{1}{V_i}\frac{1}{EPm}\int\frac{1}{2}B^2dV_i$ (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

Table 17: List of files for dynamo benchmark check sph_dynamobench

name	Parallelization	I/O
control_snapshot	Serial	Input
[sph_prefix].[rj_extension]	-	Input
[sph_prefix].[rlm_extension]	-	Input
[sph_prefix].[rtm_extension]	-	Input
[sph_prefix].[rtp_extension]	-	Input
[rst_prefix].[step#].[rst_extension]	-	Input
dynamobench.dat	Single	Output

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   2.405694011955
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
...
```

8.5 Sectioning program (sectioning)

This program generates cross sections and isosurfaces from FEM mesh data and field data using the sectioning and isosurface module in the simulation program sph_mhd. The data for this program is listed in Table 19. This program run on the parallel environment, and needs to use the same number of MPI processes as the number of processes which is used for the simulation program. VTK and compressed VTK data is not supported for the input field data.

8.5.1 Control file

The format of the control file control_viz is described below. The detail of each block is described in section A. You can jump to detailed description by clicking each item".

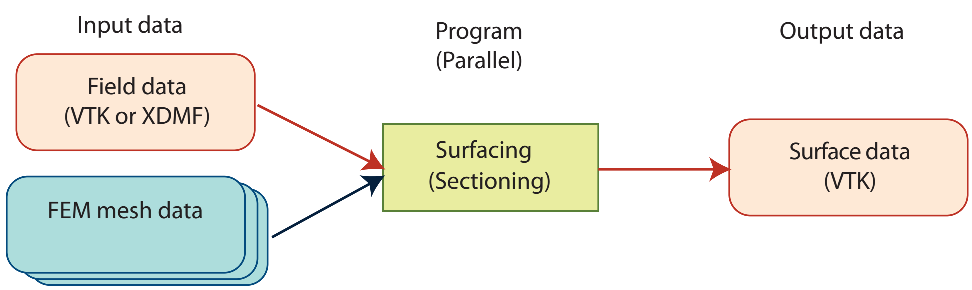


Figure 8: Data flow for sectioning program.

Table 18: List of files for sectioning `sectioning`

name	Parallelization	I/O
`control_viz`	Serial	Input
`[mesh_prefix].[fem_extension]`	-	Input
`[fld_prefix].[step#].[domain#].[extension]`	-	Input
`[section_prefix].[step#].[extension]`	Single	Output
`[isosurface_prefix].[step#].[extension]`	Single	Output

Block `visualizer` (Top block of the control file)

- Block `data_files_def`
  - `num_subdomain_ctl [Num_PE]`
  - `num_smp_ctl [Num_Threads]`
  - `mesh_file_prefix [mesh_prefix]`
  - `field_file_prefix [fld_prefix]`
  - `mesh_file_fmt_ctl [mesh_format]`
  - `field_file_fmt_ctl [fld_format]`
- Block `time_step_ctl`
  - `i_step_init_ctl [ISTEP_START]`
  - `i_step_finish_ctl [ISTEP_FINISH]`
  - `i_step_field_ctl [ISTEP_FIELD]`
  - `i_step_sectioning_ctl [ISTEP_SECTION]`
  - `i_step_isosurface_ctl [ISTEP_ISOSURFACE]`
- Block `visual_control`
  - `i_step_sectioning_ctl [ISTEP_SECTION]`
  - Array `cross_section_ctl`
    - File or Block `cross_section_ctl`
      `[section_control_file]`
      (See section 7.4)
  - `i_step_isosurface_ctl [ISTEP_ISOSURFACE]`
  - Array `isosurface_ctl`
    - File or Block `isosurface_ctl`
      `[isosurface_control_file]`
      (See section 7.5)

8.6 Field data converter program (field_to_VTK)

This program generates VTK data from FEM mesh data and field data. The data for this program is listed in Table **??**. This program run on the parallel environment, and needs to use the same number of MPI processes as the number of processes which is used for the simulation program.

Table 19: List of files for sectioning sectioning

name	Parallelization	I/O
control_viz	Serial	Input
[mesh_prefix].[fem_extension]	-	Input
[fld_prefix].[step#].[domain#].[extension]	-	Input
[fld_prefix].[step#].[domain#].[vtk] or [vtk.gz]	-	Output

8.6.1 Control file

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

Block `visualizer` (Top block of the control file)

- Block `data_files_def`
  - `num_subdomain_ctl [Num_PE]`
  - `num_smp_ctl [Num_Threads]`
  - `mesh_file_prefix [mesh_prefix]`
  - `field_file_prefix [fld_prefix]`
  - `mesh_file_fmt_ctl [mesh_format]`
  - `field_file_fmt_ctl [fld_format]`
- Block `time_step_ctl`
  - `i_step_init_ctl [ISTEP_START]`
  - `i_step_finish_ctl [ISTEP_FINISH]`
  - `i_step_field_ctl [ISTEP_FIELD]`
- Block `visual_control`
  - `output_field_file_fmt_ctl [VTK_format]`

8.7 Section and isosurface data converter program (`psf_to_VTK`)

This program generates VTK data from bindary sectioning and isosurface data. This program run on a single processor, and needs interactive input. The following is the console output of the program.

```
% /usr/local/Calypso/bin/psf_to_vtk
Input file prefix
zm_y0 <- Input file prefix
Input file extension from following:
vtk, vtk.gz, vtd, vtd.gz, inp, inp.gz, udt, udt.gz, psf, psf.gz, sdt, sdt.gz
sdt.gz <- Input extension
ifmt_input          23
```

```
Input start, end, and increment of file step
{\color{red} 2004 2000 1 <- Input start, end, and increment of file step}
Write ascii VTK file: zm_y0.2000.vtk
```

8.8 REstart data assemble program (`assemble_sph`)

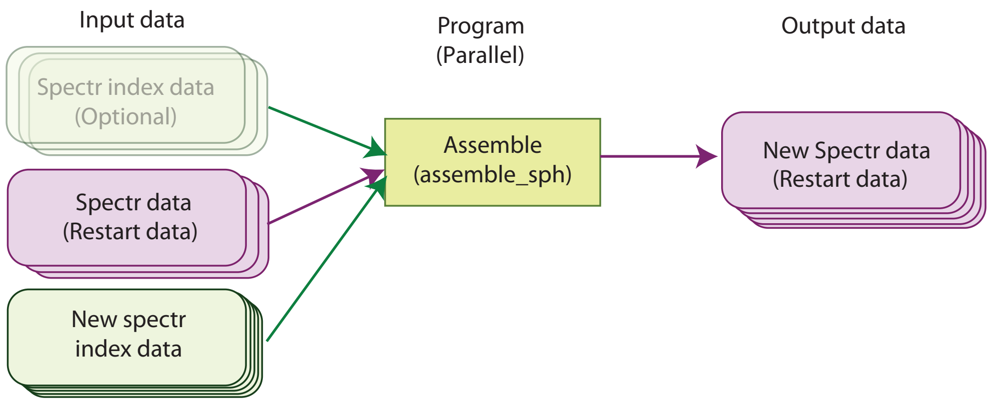


Figure 9: Data flow for spectrum data assemble program

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, but we recommend to run the program with **one** process or the same number of processes as the number of subdomains for the target configuration which is defined by `num_new_domain_ctl`. Data files for the program are shown In Table 20. The time and number of time step can also be changed by this program. The new time and time step are defined by the parameters in `new_time_step_ctl` block. The step number of the restart data will be `i_step_init_ctl` / `i_step_rst_ctl` in `new_time_step_ctl`. If `new_time_step_ctl` block is not defined, time and time step informations are carried from the original restart data.

Table 20: List of files for assemble_sph

extension	Distributed?	I/O
control_assemble_sph	Serial	Input
[sph_prefix].[rj_extension]	-	Input
[new_sph_prefix].[domain#].rj	Distributed	Input
[rst_prefix].[step#].[rst_extension]	-	Input
[new_rst_prefix].[step#].[domain#].fst	Distributed	Output

8.8.1 Format of control file

Control file consists the following groups.
Block assemble_control (Top lebel of the block)

- Block data_files_def (Detail)
  - num_subdomain_ctl [Num_PE]
  - sph_file_prefix [sph_prefix]
  - restart_file_prefix [rst_prefix])
  - sph_file_fmt_ctl [sph_format]
  - restart_file_fmt_ctl [rst_format]
- Block new_data_files_def (Detail)
  - num_subdomain_ctl [Num_PE]
  - sph_file_prefix [sph_prefix]
  - restart_file_prefix [rst_prefix])
  - sph_file_fmt_ctl [sph_format]
  - restart_file_fmt_ctl [rst_format]
  - delete_original_data_flag [YES or NO]
- Block control
  - Block time_step_ctl
    * i_step_init_ctl [ISTEP_START]

      * i_step_finish_ctl [ISTEP_FINISH]
      * i_step_rst_ctl [ISTEP_RESTART]
   - Block new_time_step_ctl
      * i_step_init_ctl [ISTEP_START]
      * i_step_rst_ctl [ISTEP_RESTART]
      * time_init_ctl [INITIAL_TIME]

- Block newrst_magne_ctl
   - magnetic_field_ratio_ctl [ratio]

8.9 Time averaging programs

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

8.9.1 Averaging for mean square and power spectrum (t_ave_sph_mean_square)

This program generate time average and standard deviation of power spectrum data. The program processes one of data files listed in Table 21. The number for the first and second interactive input is also listed in Table 21. For the third input, the file name excluding .dat is required. Start and end time is also required in the last input. If data is end before the end time, the program will finish at the end of file. t_ave and t_sigma are added at the beginning of the input file name for the time average and standard deviation data file, respectively.

8.10 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.

Table 21: List of programs to take time average

name	First input	Second input
[vol_pwr_prefix]_s.dat	1	1
[vol_pwr_prefix]_l.dat	2	1
[vol_pwr_prefix]_m.dat	2	1
[vol_pwr_prefix]_lm.dat	2	1
[layer_pwr_prefix]_s.dat	1	0
[layer_pwr_prefix]_l.dat	2	0
[layer_pwr_prefix]_m.dat	2	0
[layer_pwr_prefix]_lm.dat	2	0

8.11 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 10). 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 11)

1. Push calculator button.
2. Choose ”Point Data” in Attribute menu
3. Input data name for radial magnetic field (”B_r” in Figure 11)
4. Enter the equation to evaluate radial mantic field $B_r = \boldsymbol{B} \cdot \boldsymbol{r}/|r|$.
5. Finally, push ”Apply” button.

After obtaining the radial mantric field, the image in figure 12 is obtained by using ”slice” and ”Contour” tools with appropriate color mapping.

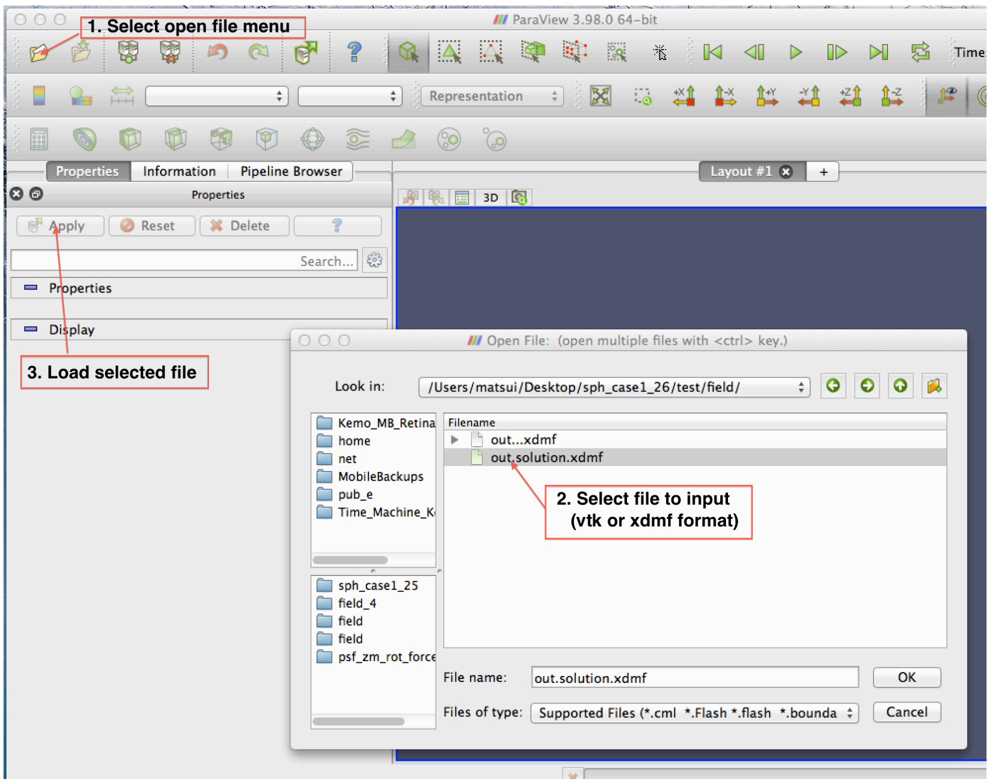


Figure 10: File open window for ParaView

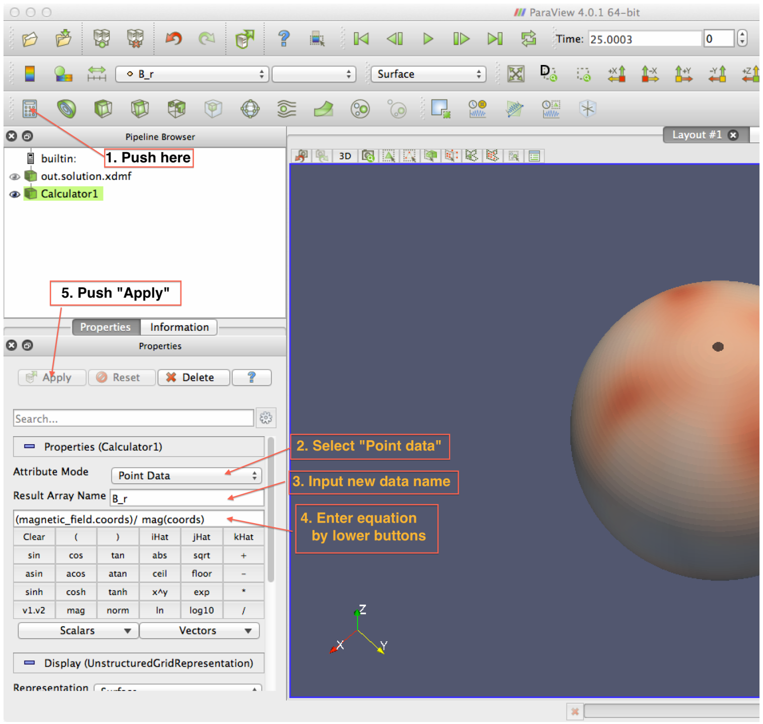


Figure 11: File open window for ParaView

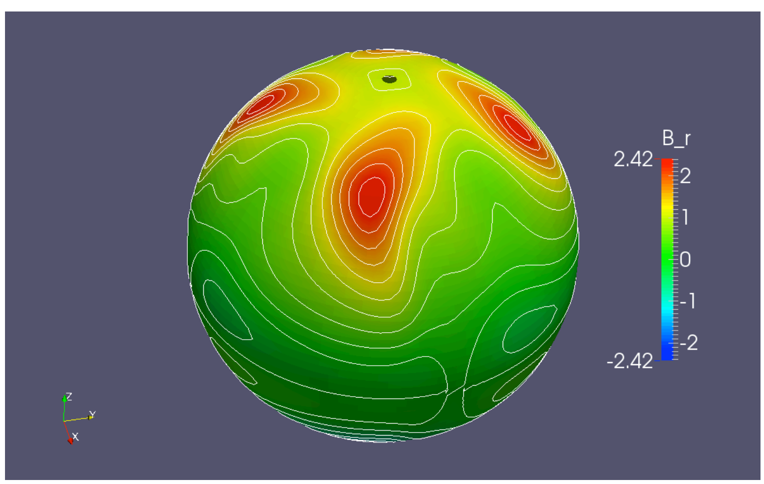


Figure 12: Visualization of radial magnetic field by Paraview.

9 Deprecated features

There are some deprecated features and programs from the Calypso Ver. 1.x. These programs are still helpful for development, debug, and to convert dat from Ver.1.x.

9.1 Preprocessing program (`gen_sph_grid`)

From Ver. 2, the spherical harmonic indices data is not necessary if the parameters for the spherical shell described below is included in the control for the simulation. However, This program `gen_sph_grid` is still useful for debug and I will describe how to set up the parallelization in this subsection. This program generates index table and a communication

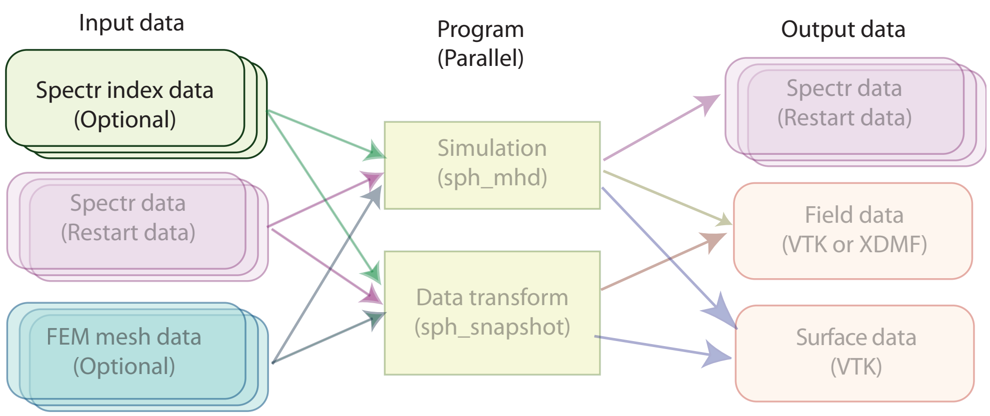


Figure 13: Generated files by preprocessing program in Data flow.

table for parallel spherical harmonics, table of integrals for Coriolis term, and FEM mesh information to generate visualization data (see Figure 13). This program needs control file for input. This program can perform with **any** number of MPI processes less than the number of subdomains, but is required to run with the SAME number of subdomains to generate MERGED data . The output files include the indexing tables.

9.1.1 Control file (`control_sph_shell`)

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

Block `MHD_control` (Top block of the control file)

Table 22: List of files for gen_sph_grid

Files	extension	Parallelization	I/O
Control file	control_sph_grid	Single	Input
Index for (r, j)	[sph_prefix].[rj_extension]	-	Output
Index for (r, l, m)	[sph_prefix].[rlm_extension]	-	Output
Index for (r, t, m)	[sph_prefix].[rtm_extension]	-	Output
Index for (r, t, p)	[sph_prefix].[rtp_extension]	-	Output
FEM mesh	[sph_prefix].[fem_extension]	-	Output
Radial point list	radial_info.dat	Single	Output

Table 23: data format flag [sph_file_fmt_ctl] and extensions.

Distributed files				
[sph_file_fmt_ctl]	ascii	binary	gzip	bin_gz
[rj_extension]	[#].rj	[#].brj	[#].rj.gz	[#].brj.gz
[rlm_extension]	[#].rlm	[#].blm	[#].rlm.gz	[#].blm.gz
[rtm_extension]	[#].rtm	[#].btm	[#].rtm.gz	[#].btm.gz
[rtp_extension]	[#].rtp	[#].btp	[#].rtp.gz	[#].btp.gz
[fem_extension]	[#].gfm	[#].gfb	[#].gfm.gz	[#].gfb.gz
Single file				
[sph_file_fmt_ctl]	merged	merged_bin	merged_gz	merged_bin_gz
[rj_extension]	.rj	.brj	.rj.gz	.brj.gz
[rlm_extension]	.rlm	.blm	.rlm.gz	.blm.gz
[rtm_extension]	.rtm	.btm	.rtm.gz	.btm.gz
[rtp_extension]	.rtp	.btp	.rtp.gz	.btp.gz
[fem_extension]	.gfm	.gfb	.gfm.gz	.gfb.gz

[#] is the domain or process number

- Block `data_files_def`
- File `spherical_shell_ctl [resolution_control]`
- or Block `spherical_shell_ctl`

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.
(see example `spherical_shell/with_inner_core`)

9.1.2 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].[rj_extension]` 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. Extension `[rj_extension]` is listed in Table 23.

`[sph_prefix].[rlm_extension]` 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. Extension `[rlm_extension]` is listed in Table 23.

`[sph_prefix].[rtm_extension]` 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. Extension `[rtm_extension]` is listed in Table 23.

`[sph_prefix].[rtp_extension]` 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. Extension `[rtp_extension]` is listed in Table 23.

9.1.3 Finite element mesh data (optional)

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 based on GeoFEM (`http://geofem.tokyo.rist.or.jp`) mesh data format, which consists of each subdomain mesh and communication table among overlapped nodes. The extension of the mesh file is listed in Table 23. This mesh data is only used in the programs `sectioning` and `field_to_VTK`.

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 Block 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.

mesh_file_prefix [mesh_prefix]
File prefix of FEM mesh file [mesh_prefix] is defined by text. Process ID and extension are added after this file prefix. This flag is only used for the sectioning program (sectioning) and data converter to VTK (field_to_VTK).

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.

sph_file_fmt_ctl [sph_formayt]
File format of spherical harmonics indexing and FEM mesh file [sph_format] is defined by text. Following data formats can be defined. Extensions of each data format is listed in Table 23.

ascii: Distributed ASCII data

binary: Distributed binary data

merged: Merged ASCII data

merged_bin: Merged binary data

gzip: Compressed distributed ASCII data

binary_gz: Compressed distributed binary data

merged_gz: Compressed merged ASCII data

merged_bin_gz: Compressed merged binary data

mesh_file_fmt_ctl [mesh_formayt]
File format of FEM mesh file [mesh_format] is defined by text. Data formats can be defined the same as sph_file_fmt_ctl. Extensions of each data format is listed in Table 23. This flag is only used for the sectioning program (sectioning) and data converter to VTK (field_to_VTK).

restart_file_fmt_ctl [rst_format]
File format of restart files [rst_format] is defined by text. Following data formats can be defined. Extensions of each data format is listed in Table 3.

ascii: Distributed ASCII data

binary: Distributed binary data

merged: Merged ASCII data

merged_bin: Merged binary data

gzip: Compressed distributed ASCII data

binary_gz: Compressed distributed binary data

merged_gz: Compressed merged ASCII data

merged_bin_gz: Compressed merged binary data

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

single_VTK_gz: Compressed merged VTK file (Available if zlib library is linked)

VTK_gz: Compressed distributed VTK file (Available if zlib library is linked)

A.2 spherical_shell_ctl

Configuration of the spherical shell and parallelization are defined by in this block. This block can be stored in an external file.

A.2.1 FEM_mesh_ctl

Configuration of the FEM mesh is defined in this block. This block is optional. (Back to control_sph_shell)

FEM_mesh_output_switch [ON or OFF]
Set ON if FEM mesh data need to be written.

A.2.2 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)

`ordering_set_ctl [ORDERING_SET]`
Ordering set of spherical harmonics and grid data is defined here. The following text parameter set `[ORDERING_SET]` is available:

`Ver_2` Optimized ordering for Ver. 2.

`Ver_1` Original data ordering for Ver. 1.x

If `[ORDERING_SET]` is not defined, ordering set for `Ver_2` is chosen.

`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_horizontal_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]` (Depricated)
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]` (Depricated)
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]` (Depricated)
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.2.3 `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.3 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 from 24 to 28

Table 24: 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	$\boldsymbol{E} = \sigma \left(\boldsymbol{J} - \boldsymbol{u} \times \boldsymbol{B} \right)$
truncated_magnetic_field	Truncated Magnetic field at Lt See truncation_degree_ctl	$\sum_{l=1}^{Lt} \boldsymbol{B}_l^m$
viscous_diffusion	Viscous diffusion	$-\nu \nabla \times \nabla \times \boldsymbol{u}$
inertia	Inertia term	$\boldsymbol{\omega} \times \boldsymbol{u}$
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	$-2\Omega \hat{z} \times \boldsymbol{u}$
pressure_gradient	Pressure gradient	$-\nabla P$
rest_of_geostrophic	Rest of geostrophic balance	$-\nabla P - 2\Omega \hat{z} \times \boldsymbol{u}$
thermal_diffusion	Termal diffusion	$\kappa_T \nabla^2 T$
grad_temp	Temperature gradient	∇T
heat_flux	Advective heat flux	$\boldsymbol{u} T$
heat_advect	Heat advection	$\boldsymbol{u} \cdot \nabla T = \nabla \cdot (\boldsymbol{u} T)$
composition_diffusion	Compositional diffusion	$\kappa_C \nabla^2 C$
grad_composition	Composition gradient	∇C
composite_flux	Advective composition flux	$\boldsymbol{u} C$
composition_advect	Compositional advection	$\boldsymbol{u} \cdot \nabla C = \nabla \cdot (\boldsymbol{u} C)$
magnetic_diffusion	Magnetic diffusion	$-\eta \nabla \times \nabla \times \boldsymbol{B}$
vecp_induction	Induction for the vector potential	$\boldsymbol{u} \times \boldsymbol{B}$
magnetic_induction	Magnetic induction	$\nabla \times (\boldsymbol{u} \times \boldsymbol{B})$
poynting_flux	Poynting flux	$\boldsymbol{E} \times \boldsymbol{B}$

Table 25: List of field name (Continued)

[Name]	field name	Description
rot_inertia	Curl of inertia	$\nabla \times (\boldsymbol{\omega} \times \boldsymbol{u})$
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	$-\nabla \times (\alpha_C C \boldsymbol{g})$
Lorentz_work	Work of Lorentz force	$\boldsymbol{u} \cdot (\boldsymbol{J} \times \boldsymbol{B})$
work_against_Lorentz	Work against Lorentz force	$-\boldsymbol{u} \cdot (\boldsymbol{J} \times \boldsymbol{B})$
buoyancy_flux	Thermal buoyancy flux	$-\alpha_T T \boldsymbol{g} \cdot \boldsymbol{u}$
composite_buoyancy_flux	Compositional buoyancy flux	$-\alpha_c C \boldsymbol{g} \cdot \boldsymbol{u}$
magnetic_ene_generation	Energy production by magnetic induction	$\boldsymbol{B} \cdot (\boldsymbol{u} \times \boldsymbol{B})$
kinetic_helicity	Kinetic helicity	$\boldsymbol{u} \cdot \boldsymbol{\omega}$
current_helicity	Current helicity	$\boldsymbol{B} \cdot \boldsymbol{J}$
cross_helicity	Cross helicity*	$\boldsymbol{u} \cdot \boldsymbol{B}$

*: Magnetic helicity $\boldsymbol{A} \cdot \boldsymbol{B}$ is not implimented.

Table 26: List of field names decomposed by equatorial symmetry

Symmetric		Anti-symmetric	
[Name]	expression	[Name]	expression
sym_velocity	$\boldsymbol{u}_{sym}$	asym_velocity	$\boldsymbol{u}_{asym}$
sym_vorticity	$\boldsymbol{\omega}_{sym} = \nabla \times \boldsymbol{u}_{asym}$	asym_vorticity	$\boldsymbol{\omega}_{asym} = \nabla \times \boldsymbol{u}_{sym}$
sym_pressure	P_{sym}	asym_pressure	P_a
sym_temperature	T_{sym}	asym_temperature	T_{asym}
sym_composition	C_{sym}	asym_composition	C_{asym}
sym_magnetic_field	$\boldsymbol{B}_{sym}$	asym_magnetic_field	$\boldsymbol{B}_{asym}$
sym_current_density	$\boldsymbol{J}_{sym} = \nabla \times \boldsymbol{B}_{asym}$	asym_current_density	$\boldsymbol{J}_{asym} = \nabla \times \boldsymbol{B}_{sym}$

Table 27: List of force and nonlinear term names decomposed by equatorial symmetry

[Name]	expression
sym_thermal_buoyancy	$-\alpha_T T_{sym} \boldsymbol{g}$
asym_thermal_buoyancy	$-\alpha_T T_{asym} \boldsymbol{g}$
sym_composite_buoyancy	$-\alpha_C C_{sym} \boldsymbol{g}$
asym_composite_buoyancy	$-\alpha_C C_{asym} \boldsymbol{g}$
wsym_x_usym	$\boldsymbol{\omega}_{sym} \times \boldsymbol{u}_{sym}$
wasym_x_uasym	$\boldsymbol{\omega}_{asym} \times \boldsymbol{u}_{asym}$
wsym_x_uasym	$\boldsymbol{\omega}_{sym} \times \boldsymbol{u}_{asym}$
wasym_x_usym	$\boldsymbol{\omega}_{asym} \times \boldsymbol{u}_{sym}$
Jsym_x_Bsym	$\boldsymbol{J}_{sym} \times \boldsymbol{B}_{sym}$
Jasym_x_Basym	$\boldsymbol{J}_{asym} \times \boldsymbol{B}_{asym}$
Jsym_x_Basym	$\boldsymbol{J}_{sym} \times \boldsymbol{B}_{asym}$
Jasym_x_Bsym	$\boldsymbol{J}_{asym} \times \boldsymbol{B}_{sym}$
usym_x_Bsym	$\boldsymbol{u}_{sym} \times \boldsymbol{B}_{sym}$
uasym_x_Basym	$\boldsymbol{u}_{asym} \times \boldsymbol{B}_{asym}$
usym_x_Basym	$\boldsymbol{u}_{sym} \times \boldsymbol{B}_{asym}$
uasym_x_Bsym	$\boldsymbol{u}_{asym} \times \boldsymbol{B}_{sym}$

Table 28: List of energy flux names decomposed by equatorial symmetry

[Name]	expression
sym_buoyancy_flux	$-\boldsymbol{u}_{sym} \cdot \alpha_T T_{sym} \boldsymbol{g}$
asym_buoyancy_flux	$-\boldsymbol{u}_{asym} \cdot \alpha_T T_{asym} \boldsymbol{g}$
-ua_d_ws_x_us	$\boldsymbol{u}_{asym} \cdot (\boldsymbol{\omega}_{sym} \times \boldsymbol{u}_{sym})$
-ua_d_wa_x_ua	$\boldsymbol{u}_{asym} \cdot (\boldsymbol{\omega}_{asym} \times \boldsymbol{u}_{asym})$
-us_d_ws_x_ua	$\boldsymbol{u}_{sym} \cdot (\boldsymbol{\omega}_{sym} \times \boldsymbol{u}_{asym})$
-us_d_wa_x_us	$\boldsymbol{u}_{sym} \cdot (\boldsymbol{\omega}_{asym} \times \boldsymbol{u}_{sym})$
ua_d_js_x_bs	$\boldsymbol{u}_{asym} \cdot (\boldsymbol{J}_{sym} \times \boldsymbol{B}_{sym})$
ua_d_ja_x_ba	$\boldsymbol{u}_{asym} \cdot (\boldsymbol{J}_{asym} \times \boldsymbol{B}_{asym})$
us_d_js_x_ba	$\boldsymbol{u}_{sym} \cdot (\boldsymbol{J}_{sym} \times \boldsymbol{B}_{asym})$
us_d_ja_x_bs	$\boldsymbol{u}_{sym} \cdot (\boldsymbol{J}_{asym} \times \boldsymbol{B}_{sym})$

A.4 `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 29.

Table 29: 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	$\boldsymbol{B}$

A.5 `boundary_condition`

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

`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 7.1.5.

`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 7.1.5.

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 7.1.5.

`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 7.1.5.

A.6 `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 30: 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.7 `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 31: 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.8 `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.8.1 `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 $\frac{\partial T}{\partial t}$ 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.8.2 `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.8.3 `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.8.4 `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 $\dfrac{\partial C}{\partial t}$ 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.9 `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.10 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.

`i_step_sectioning_ctl   [ISTEP_SECTION]`
Increment of time step to output cross section data for visualization `[ISTEP_SECTION]` is defined by integer. If `[ISTEP_SECTION]` is set to be 0, no cross section data are written. If `[ISTEP_SECTION]` is set in the block `visual_control`, The value in `visual_control` is used.

`i_step_isosurface_ctl   [ISTEP_ISOSURFACE]`
Increment of time step to output isosurface data for visualization `[ISTEP_ISOSURFACE]` is defined by integer. If `[ISTEP_ISOSURFACE]` is set to be 0, no isosurface data are written.If `[ISTEP_ISOSURFACE]` is set in the block `visual_control`, The value in `visual_control` is used.

`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.11 `new_time_step_ctl`

Time stepping parameters to update initial data are defined in this block. Items in this block is the same as `i_step_field_ctl`. (Back to `control_assemble_sph`)

A.12 `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.13 `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

Legendre_trans_loop_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.14 sph_monitor_ctl

Monitoring data is defined in this block. Monitoring data output (mean square, average, Gauss coefficients, or specific components of spectrum data) 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` or `.dat.gz` 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]_s.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_s.dat`.

`volume_pwr_spectr_format [file_format]`
File format for mean square spectrum data averaged over the fluid shell. If `[file_format]` is `gzip`, volume average, mean square and spectrum data are compressed by zlib. If the mean square data file exist, this setting is ignored and append data in the existing file.

`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` or `.dat.gz` 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 exsists.

`nusselt_number_format [file_format]`
File format for Nusselt number data `[nusselt_number_prefix].dat`. If `[file_format]` is `gzip`, Nusselt number data is compressed by zlib. If the Nusselt number data file exist, this setting is ignored and append data in the existing file.

A.14.1 `volume_spectrum_ctl`

Volume average of power spectrum and mean square data between any radius range are defined in this block.

`inner_radius_ctl [radius]`
Inner boundary of the volume average `[radius]` is defined. The closest radial grid point is chosen as a inner boundary of averaging.

`outer_radius_ctl [radius]`
Outer boundary of the volume average `[radius]` is defined. The closest radial grid point is chosen as the outer boundary of averaging.

A.14.2 `layered_spectrum_ctl`

Sphere average of power spectrum and mean square data are defined in this block.

`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.

`layered_pwr_spectr_format [file_format]`
File format for mean square spectrum data averaged over the fluid shell. If `[file_format]` is `gzip`, the mean square and spectrum data are compressed by zlib. Otherwise, data file is written by text format. If the mean square data file exist, this setting is ignored and append data in the existing file.

array spectr_layer_ctl [Layer #] List of radial grid point number [Layer #] to output power spectrum data by integer. If this array is not defined, layered mean square data are written for all radial grid points.

A.14.3 gauss_coefficient_ctl

Gauss coefficients data at specified radius are defined in this block.

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

gauss_coefs_format [file_format]
File format for the Gauss coefficients data file. If [file_format] is gzip, volume mean square and spectrum data is compressed by zlib. Otherwise, data file is written by text format. If the Gauss coefficients data file exist, this setting is ignored and append data in the existing file.

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.

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.

A.14.4 pickup_spectr_ctl

Spherical harmonic coefficients data output is defined in this block.

picked_sph_prefix [picked_sph_prefix]
File prefix for picked spectrum data files [picked_sph_prefix] is defined by Text. Program add [picked_sph_prefix]_l[degree]_m[order][c/s].dat extension after this file prefix and save each mode of spherical harmonics coefficients data file. If this file prefix is not defined, picked spectrum data are not generated. Because data files are generated for each spherical harmonics mode, we recommend to save these files in a sub directory.

picked_sph_format [file_format]
File format for picked spectrum data files. If [file_format] is gzip, picked spectrum data files are compressed by zlib. Otherwise, data files are written by text format. If the picked spectrum data files exist, this setting is ignored and data is appended in the existing file.

array pick_layer_ctl [Layer #] List of radial grid point number [Layer #] to output picked 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.

A.14.5 sph_dipolarity_ctl

Dipolrity data output is defined in this block.

`dipolarity_file_prefix [dipolarity_file_prefix]`
File prefix for dipolarity data files `[dipolarity_file_prefix]` is defined by Text. Program add `.dat` or `.dat.gz` extension after this file prefix and save dipolarity data. If this file prefix is not defined, dipolarity data are not generated.

`dipolarity_file_format [file_format]`
File format for dipolarity data files. If `[file_format]` is `gzip`, dipolarity data files are compressed by zlib. Otherwise, the data file is written by text format. If the dipolrity data file exist, this setting is ignored and data is appended in the existing file.

`array dipolarity_truncation_ctl [Degree]`
Truncation degrees l to evaluate dipolarity is defined in `[Degree]` by integer. More than 1 truncations can be defined in this array. The dipolarity with the truncation degree of the simulation L_{max} is automatically added in the dta output.

A.14.6 `mid_equator_monitor_ctl`

Parameters to generate data at mid-depth of euqtorial plane are defined in this block.

`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.15 `visual_control`

Visualization modules are defined in this block. Parameters for cross sections and isosurfaces are defined in this block.
(Back to `visual_control`)

A.16 `cross_section_ctl`

Control parameters for cross sectioning are defined in this block.
(Back to `cross_section_ctl`)

`section_file_prefix [file_prefix]`
File prefix for cross section data is defined as character `[file_prefix]`.

psf_output_type [file_format]
File format for cross section data is defined as character [file_format]. The following formats are available;

- VTK: VTK format
- VTK_gz: Compressed VTK format (Available if zlib library is linked)
- PSF: Binary section data format
- PSF_gzip: Compressed Binary section data format (Available if zlib library is linked)

A.16.1 surface_define

Each cross section is defined in this block.
(Back to cross_section_ctl)

section_method [METHOD]
Method of the cross sectioning is defined as character [METHOD]. Supported cross section is shown in Table 32

Table 32: Supported cross sections

[METHOD]	Surface type
equation	Quadrature surface
plane	Plane surface
sphere	Sphere
ellipsoid	Ellipsoid

coefs_ctl [TERM] [COEFFICIENT]
This array defines coefficients for a quadrature surface described by

$$ax^2 + by^2 + cz^2 + dyz + ezx + fxy + gx + hy + jz + k = 0.$$

Each coefficient a to k are defined by the name of the term [TERM] and real value [COEFFICIENT] as shown in Table 33.

Table 33: List of coefficient labels for quadrature surface

[TERM]	Defined value	[TERM]	Defined value	[TERM]	Defined value
x2	a	y2	b	z2	c
yz	d	zx	e	xy	f
x	g	y	h	z	i
const	h				

`radius [SIZE]`
`[SIZE]` defines radius r for a sphere surface defined by

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2.$$

`normal_vector [DIRECTION] [COMPONENT]`
This array defines normal vector (a, b, c) for a plane surface described by

$$a\,(x - x_0) + b\,(y - y_0) + c\,(z - z_0) = 0.$$

Each component is defined by `[DIRECTION]` and real value `[COMPONENT]` as shown in Table 34.

Table 34: List of coefficient labels for vector

[DIRECTION]	Defined value
x	a
y	b
z	c

`axial_length [DIRECTION] [COMPONENT]`
This array defines size (a, b, c) of an ellipsoid surface described by

$$\left(\frac{x - x_0}{a}\right)^2 + \left(\frac{y - y_0}{b}\right)^2 + \left(\frac{z - z_0}{c}\right)^2 = 1.$$

Each component is defined by `[DIRECTION]` and real value `[COMPONENT]` as shown in Table 34.

`center_position [DIRECTION] [COMPONENT]`
Position of center (x_0, y_0, z_0) of sphere or ellipsoid is defined this array. Position on a plane surface (x_0, y_0, z_0) is also defined. Each component is defined by `[DIRECTION]` and real value `[COMPONENT]` as shown in Table 35.

Table 35: List of coefficient labels for vector

`[DIRECTION]`	Defined value
`x`	x_0
`y`	y_0
`z`	z_0

`section_area_ctl` Areas for the cross sectioning are defined in this array. The following groups can be defined in this block.

`outer_core` Outer core.

`inner_core` Inner core (If exist).

`external` External of the core (If exist).

`all` Whole simulation domain.

A.16.2 `output_field_define`

Field data on the cross section are defined in this block.
(Back to `cross_section_ctl`)

`output_field` Field informations for cross section are defined in this array. Name of the output fields is defined by `[FIELD]`, and component of the fields is defined by `[COMPONENT]`. Labels of the field name are listed in Table 24, and labels of the component are listed in Table 36.

A.17 `isosurface_ctl`

Control parameters for isosurfacing are defined in this block.
(Back to `isosurface_ctl`)

Table 36: List of field type for cross sectioning and isosurface module

[COMPONENT]	Field type
scalar	scalar field
vector	Cartesian vector field
x	x-component
y	y-component
z	z-component
radial	radial (r-) component
theta	θ-component
phi	ϕ-component
cylinder_r	cylindrical radial (s-) component
magnitude	magnitude of vector

`isosurface_file_prefix [file_prefix]`
File prefix for isosurface data is defined as character `[file_prefix]`.

`iso_output_type` File format for isosurface data is defined as character `[file_format]`. The following formats are available;

`VTK`: VTK format

`VTK_gz`: Compressed VTK format (Available if zlib library is linked)

`ISO`: Binary isosurface data format

`ISO_gzip`: Compressed Binary isosurface data format (Available if zlib library is linked)

A.17.1 `isosurf_define`

Each isosurface is defined in this block.
(Back to `isosurface_ctl`)

`isosurf_field` Field name for isosurface is defined by `[FIELD]`. Labels of the field name are listed in Table 24.

`isosurf_component` Component name for isosurface is defined by `[COMPONENT]`. Labels of the component are listed in Table 36.

`isosurf_value` Isosurface value is defined as real value `VALUE`.

`isosurf_area_ctl` Areas for the isosurfacing are defined in this array. The same groups can be defined as `section_area_ctl`.

A.17.2 `field_on_isosurf`

Field data on the isosurface are defined in this block.
(Back to `isosurface_ctl`)

`result_type` Output data type is defined by `[TYPE]`. Following types can be defined:

- `constant` Constant value is set as a result field. The amplitude is set by `result_value`.
- `field` field data on the isosurface are written. Fields to be written are defined by `output_field` array.

`result_value` Isosurface value is defined as real value `VALUE`.

`output_field` Field informations for cross section are defined in this array. Name of the output fields is defined by `[FIELD]`, and component of the fields is defined by `[COMPONENT]`. Labels of the field name are listed in Table 24, and labels of the component are listed in Table 36.

A.18 `output_field_file_fmt_ctl [VTK_format]`

File format of field data is defined as character `[VTK_format]`. THe following formats are available.

- `single_HDF5:` Merged HDF5 file (Available if HDF5 library is linked)
- `single_VTK:` Merged VTK file (Default)
- `VTK:` Distributed VTK file
- `single_VTK_gz:` Compressed merged VTK file (Available if zlib library is linked)

VTK_gz: Compressed distributed VTK file (Available if zlib library is linked)

A.19 dynamo_vizs_control

Visualization for zonal mean, RMS, and truncated magnetic field are defined in this block. Parameters for cross section is set for zonal mean and RMS, and spherical harmonics degree of the truncated magnetic field is also defined here.
(Back to dynamo_vizs_control)

zonal_mean_section_ctl Control parameters for cross section of the zonal mean field are defined in this block. This block has the same control items as cross_section_ctl. In the external file [zonal_mean_section_control_file], con trol block starts from cross_section_ctl.

zonal_RMS_section_ctl Control parameters for cross section of the zonal RMS field are defined in this block. This block has the same control items as cross_section_ctl. In the external file [zonal_RMS_section_control_file], con trol block starts from cross_section_ctl.

crustal_filtering_ctl Set the truncation degree to make the truncated magnetic field by the crustal magnetic field. The spherical harmonics degree of the truncated magnetic field is defined in truncation_degree_ctl. In the external file [zonal_mean_section_contro
con trol block starts from cross_section_ctl.

A.20 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)

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.21 new_time_step_ctl

Parameters to modify time step and time data in the new restart file.
(Back to control_assemble_sph)

`magnetic_field_ratio_ctl [ISTEP_START]`
New time step `[ISTEP_START]` for the restart file is defined by integer.

`i_step_rst_ctl [ISTEP_RESTART]`
New step number of restrart file `[ISTEP_RESTART]` is defined by integer.

`time_init_ctl [INITIAL_TIME]`
New time data `[INITIAL_TIME]` is defined by real.

A.22 `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.