DRS (1.10.4)
A User's Manual

Luis Silva

October 6, 2015

Contents

Introduction 1

1 How to use this code 3
1.1 DEPENDENCIES . . . 3
1.2 Compiling and installing the code . . . 3
1.2.1 Building the online documentation . . . 4
1.2.2 COMPILATION (sequential) . . . 4
1.2.3 COMPILATION (parallel) . . . 4
1.3 Running DRS . . . 4
1.3.1 Command-line options . . . 4
1.3.2 Before running DRS . . . 4
1.3.3 Running on one CPU . . . 6
1.3.4 Running on multiple CPU's with MPI . . . 6
1.4 A variety of queueing systems . . . 6
1.4.1 No queueing system . . . 6
1.4.2 SLURM . . . 7
1.4.3 TORQUE/PBS . . . 7

2 Theory 9
2.1 Problem set up and approximations . . . 9
2.1.1 The Boussinesq approximation . . . 10
2.1.2 The adimensionalisation . . . 11
2.2 Adding composition . . . 12
2.2.1 Testing the compositional convection implementation . . . 13
2.3 Boundary conditions . . . 13
2.3.1 Boundary conditions on the flow . . . 13
2.3.2 Boundary conditions on the temperature . . . 14
2.3.3 Boundary conditions on the magnetic field . . . 14
2.3.4 Boundary conditions on the composition . . . 14

3 Discretization and numerical methods 15
3.1 Poloidal/toroidal decomposition . . . 15
3.2 Polynomial decompositions . . . 15
3.2.1 Spherical harmonic decomposition . . . 16
3.2.2 Chebyshev decomposition . . . 17
3.2.3 Spectra . . . 18
3.3 Crank-Nicholson/Adams-Bashford integration . . . 18
3.3.1 Crank-Nicholson integration . . . 19

3.3.2 Adams-Bashford integration . . . 19
3.3.3 Putting it all together . . . 19
3.4 Time-stepping . . . 20
3.4.1 Adaptative time-stepping . . . 20

4 Input/Output **21**
4.1 Inputs . . . 21
4.1.1 State files . . . 21
4.1.2 par files . . . 21
4.2 Outputs . . . 22
4.2.1 Quantities saved at the end of the run . . . 22
4.2.2 Frequently probed quantities . . . 23

A Utilities **27**
A.1 drs-setup . . . 27
A.2 drs-spectra . . . 27
A.3 drs2dx . . . 27
A.4 drs-state-change . . . 28
A.5 drs-state-average . . . 29
A.6 Utility scripts . . . 29
A.6.1 compressdata/uncompressdata . . . 29
A.6.2 drs_plot.sh . . . 29
A.6.3 drs_cleanRun.sh . . . 29
A.6.4 latestNormalizedSpectra.sh . . . 29
A.6.5 resubmitJob.sh . . . 29
A.6.6 rebuildTimeSeries.sh . . . 29
A.6.7 drs_plot.sh . . . 29
A.6.8 stopRun.sh . . . 30

B Benchmarks **31**
B.1 Christensen 2001 . . . 31
B.2 2013/2014 . . . 31

Introduction

This is the user manual for the DRS (Dynamo in a Rotating Shell) code. API documentation can be found on the online APIDOX. Ported to f95 by Luis Silva (lacsilva@gmail.com)

Chapter 1

How to use this code

1.1 DEPENDENCIES

We use `cmake` version 2.8 as the build system. We depend on `FFTW3` version >= 3.2.1. Tested up to version 3.3.3. We use `OpenMPI` >= 1.6 for the parallel version. The binary wrappers must be in your `$PATH` and the libraries in your `$LD_LIBRARY_PATH`. Online documentation can be built using `doxygen` version >= 1.6.1. In order to compress the state files we use `gzip`. Most unix systems will have it

1.2 Compiling and installing the code

The DRS code and utilities use the cmake build system in order to generate native builds for *nix and Windows alike. Some features can be activated or deactivated at compile time by passing the appropriate "-D" option to cmake. Table 1.1 summarises the present set of feature options. Check the cmake documentation [author, 2014] for other build options.

Option	Values	Description
MPI	ON/OFF	Activates/deactivates compilation of the parallel parts of the code. OFF by default.
COMP	ON/OFF	Activates/deactivates compilation of support for composition. OFF by default.
BUILD_UTILS	ON/OFF	Activates/deactivates compilation of the utilities. OFF by default.
BUILD_TESTS	ON/OFF	Activates/deactivates compilation of the unit tests. OFF by default.
BUILD_DOCS	ON/OFF	Build the user manual from the latex sources. ON by default when pdflatex is found.

Table 1.1: List of options that activate/deactivate features at compile time.

1.2.1 Building the online documentation

```
$ mkdir BUILD
$ cd BUILD
$ cmake ..
$ make doc
```

The code documentation can then be found by pointing a browser to:
`<your favorite browser> BUILD/APIDOX/index.html`

1.2.2 COMPILATION (sequential)

```
$ mkdir BUILD
$ cd BUILD
$ cmake ..
$ make && make install
```

After all run successfully, the file drs.exe should be in `bin/`.

1.2.3 COMPILATION (parallel)

```
$ mkdir BUILD
$ cd BUILD
$ cmake -DMPI=ON ..
$ make && make install
```

After all run successfully, the file drs.exe should be in the `bin/` folder of the source code folder.

1.3 Running DRS

1.3.1 Command-line options

Since version 1.6.1, DRS acceps the `-v` command line option. When this option is used the program will write the version and compile time options and exit with code 1.

1.3.2 Before running DRS

Each run needs to be configured. Configuration is passed on the standard input and obeys a format similar to the following.

```
| inputfilename | outputfilname |
   abc.x             def.y
|     noise    |
    0.D0
| magic | lformat | drs_calc_type | tempBC | compBC | flowBC | magBC |
  10109     1          13            0        0        1       0
|  eta  | Therm. Prandtl | Compo. Prandtl | Taylor | Therm. Rayleigh | Compo. Rayleigh |  Magn. Prandtl  |
 0.35          0.5            1.0          4.0D6        1.0D5            1.0D5              1.0
|  Nr  |  Nt  |  Np  |  Nr_s  |  Nt_s  |  Np_s  | lsymm |  m0  |
  129     96    193      129      64       129      0     1
|delta_t|nsteps|cpu_max_time|transient|sampling_rate|
  0.0d0  10000   2.0e0       0          200
'A general comment no greater than 60 characters.'
```

At this point several formats are available, described by the `magic` number. Templates for several `magic` numbers are provided in the templates folder.

The first two lines indicate the basename for the input and output statefiles and `.par` files. Next we can find the noise level each variable is to be initialised to in case the corresponding state file is not found.

The next two lines are particularly worthy of note. They determine the output format, computation to carry and the boundary conditions to be used for each quantity. The meaning of each of `temBC`, `compBC`, `flowBC` and `magBC` is explained in section 2.3.

Possible valued for the format, `lformat` are 0 for ascii unformated and any other value for binary formated output.

Possible values for the computation type, `drs_calc_type`, are:

1 `LinearThermalOnset`

2 `KinematicDynamo`

3 `NonlinearConvection`

4 `NonlinearDynamo`

5 `MagnetoConvection`

6 `LinearCompositionalOnset`

Add 10 to include composition.

Possible values for the boundary conditions are 0 or 1 acording to the following table:

Opt.	`temBC`	`compBC`	`flowBC`	`magBC`
0	Fixed temperature	Fixed composition	Free slip	Vacuum
1	Fixed heat flux	Fixed chemical flux	No slip	Pseudo-vacuum

Table 1.2: Possible values for the the boundary conditions and their meanings.

In the folder where DRS is run, the program must also be able to find the initial state files. These contain the coefficients for the flow, temperature, magnetic field, etc.. A typical directory listing prior to the execution looks like:

```
 > ll
total 680K
-rw------- 1 user group 130K Sep 30 15:56 e035p1t2r100000m1p5.1.Bp.gz
-rw------- 1 user group 129K Sep 30 15:56 e035p1t2r100000m1p5.1.Bt.gz
-rw------- 1 user group  528 Nov 12 12:34 e035p1t2r100000m1p5.1.par
-rw------- 1 user group 131K Sep 30 15:56 e035p1t2r100000m1p5.1.pol.gz
-rw------- 1 user group 129K Sep 30 15:56 e035p1t2r100000m1p5.1.temp.gz
-rw------- 1 user group 129K Sep 30 15:56 e035p1t2r100000m1p5.1.tor.gz
-rwx--x--x 1 user group 4.2K Nov 20 15:42 jobscript-parallel
-rwx--x--x 1 user group 4.3K Nov 26 12:27 jobscript-serial
```

Notice that the input state files are compressed using unix gzip. This is because space saving requirements and the decompression of these files is dealt with by the `jobscrit-*` scripts described below. The file with extension `.par` describes the structure and contents of the other files. Extensions `.pol` and `.tor`

are for the coefficients of the poloidal and toroidal flow. Similarly, `.Bp` and `.Bt` are for the coefficients of the poloidal and toroidal magnetic field. Extension `.temp` refers to the temperature. See section 4.1 for details on the structure of the `.par` and state files.

1.3.3 Running on one CPU

1.3.4 Running on multiple CPU's with MPI

1.4 A variety of queueing systems

Because some preparation is needed prior to running DRS and because some post-processing (like storing the output data in a predetermined place) may be needed, we recommend using job scripts. Typically, these scripts will be adapted to run under a queueing system that provides resource and I/O management and may require modifications to the pre/post processing. A few examples of these job scripts follow for a variety of queueing systems.

1.4.1 No queueing system

```
#!/bin/bash
#############################################################
######
######    Example shell script to serve as a job launcher
######    and controller.
######
#
#-------------------------------------------------
# Check the following settings before submitting!

# The base name of the state files before numbering.
file=MyJobFilename

# Where to find the utilities required to deal with the state files.
BINDIR=${HOME}/bin

# Where to find the program executale
command=${BINDIR}/drs.exe

# Number of the run on input
# This is overriden by the value in ${job}.num
numi=1

# Highest number of the run on output.
maxnum=10

# Construct the input/output file numbering.
numo=$numi + 1
infile="$file.$numi"
outfile="$file.$numo"

# The following few lines establish the parameters to be used in the
```

```
# computation. Create and check drs.conf.template carefully before
# running.
sed -e "s//$infile/g" \
    -e "s//$outfile/g" \
    drs.conf.template > ${job}.in

# Uncompress the input files
${BINDIR}/uncompressdata $infile

# Run the simulation
${command} < ${job}.in
```

The format for `drs.conf.template` is described in section 1.3.2. An occurrence of and should exist in that file so that they can be replaced by the appropriate values.

1.4.2 SLURM

The SLURM queueing system requires the above example to be slightly modified.

```
#!/bin/bash
#################################################################
######
######    Example shell script to serve as a job launcher
######    and controller.
######    This example is meant for execution of
######    the program the slurm queueing system.
######
#SBATCH -J myMPI        # Job name
#SBATCH -o myjob.%j.out # stdout output file (%j expands to jobId)
#SBATCH -e myjob.%j.err # stderr output file (%j expands to jobId)
#SBATCH -p development  # Queue name
#SBATCH -N 2            # Number of nodes requested (16 cores/node)
#SBATCH -n 32           # Number of mpi tasks requested
#SBATCH -t 12:00:00     # Run time (hh:mm:ss) - 1.5 hours
.
.
.
# Run the simulation
ibrun -n $NPROCS ${command} < ${job}.in
```

1.4.3 TORQUE/PBS

The TORQUE/PBS style of queueing systems requires the above example to be slightly modified.

```
#!/bin/bash
#################################################################
######
######    Example shell script to serve as a job launcher
######    and controller.
######    This example is meant for execution of
######    the program under the torque/pbs queueing system.
```

```
######
#PBS -N e07p075t20r180000m1p1-high-eb
#PBS -l nodes=1:ppn=4
#PBS -l walltime=48:00:00
#PBS -m bea -M luis.silva@glasgow.ac.uk
.
.
.
mpirun -np $NPROCS ${command} < ${job}i
```

Chapter 2

Theory

2.1 Problem set up and approximations

We consider a spherical fluid shell of inner radius r_i and outer radius r_o. The shell thickness is then $d = r_o - r_i$. The shell is rotating at constant angular velocity Ω about an axis aligned with the z direction. Boundaries are impermeable and electrically insulating. Figure 2.1 shows a depiction of this geometry.

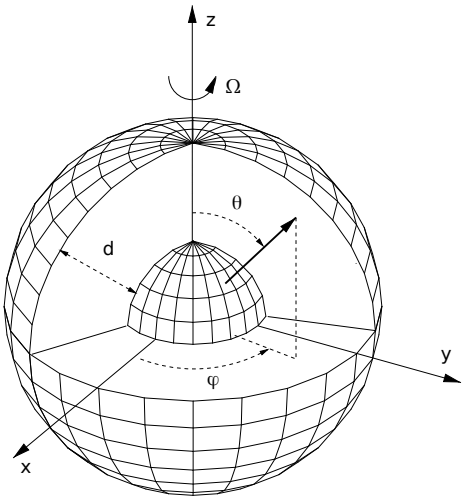


Figure 2.1: Depiction of the problem set-up.

The fluid inside the shell is incompressible, electrically and thermally conductive with magnetic diffusivity λ and thermal diffusivity κ, and has a viscosity ν. Optionally, we can also consider that the fluid is a nearly homogeneous mixture composed of a heavy solvent and a light solute with chemical diffusivity D. The system is permeated by a magnetic field that evolves in time through both advection and diffusion. The equations that describe this system are then:

$$\nabla \cdot \mathbf{u} = 0, \qquad \nabla \cdot \mathbf{B} = 0, \tag{2.1a}$$

$$\partial_t(\rho \mathbf{u}) = -\mathbf{u} \cdot \nabla(\mathbf{u}\rho) - 2\rho\boldsymbol{\Omega} \times \mathbf{u} - \nabla P + \rho\mathbf{g} + \nabla \times (\nu \nabla \times \mathbf{u}) + \mathbf{j} \times \mathbf{B}, \tag{2.1b}$$

$$\partial_t T = -\mathbf{u}\cdot\nabla T + \kappa\nabla^2 T + S_T, \tag{2.1c}$$
$$\partial_t C = -\mathbf{u}\cdot\nabla C + \kappa\nabla^2 C + S_C, \tag{2.1d}$$
$$\partial_t \mathbf{B} = -\mathbf{u}\cdot\nabla\mathbf{B} + \mathbf{B}\cdot\nabla\mathbf{u} + \nabla\times(\lambda\nabla\times\mathbf{B}). \tag{2.1e}$$

An equation similar to 2.1c can be added that describes the evolution of the concentration of the light solute. The system is closed with an equation of state that describes the evolution of ρ as a function of the other variables.

Before we continue we are going to make a few more approximations. We will assume that all diffusivities are constant both in time and space and that the electric field is small enough and the electric conductivity is high enough that there is no net charge imbalance. This last assumption leads to:

$$\mathbf{j} = \frac{1}{\mu}\nabla\times\mathbf{B}, \tag{2.2}$$

with μ the magnetic permeability of the core material.

2.1.1 The Boussinesq approximation

Up until this point, the only approximations that were made relate to the fact that the fluid is incompressible, that the electric field is deemed small on the relevant time scales and all problem parameters are constant. Besides that, we limited the body forces that act on the fluid to the viscous force and the magnetic force. The flow is also heated in the volume by a non-descript source S which we will discuss later.

The Boussinesq approximation consists in making use of the yet unprescribed equation of state to postulate that the density variations both in time and space are negligeable except when it comes to their effect on the gravity term. That is, $\rho = \rho_0$, except when it comes to the term $\rho\mathbf{g}$. There we have:

$$\rho \approx \rho_0 + \frac{\partial\rho}{\partial T}\delta T + \frac{\partial\rho}{\partial C}\delta C, \tag{2.3}$$

where C is the concentration of light solute. We can now identify the partial derivatives with respect to the thermo-dynamic quantities as being related to the thermal expansion factor and the compositional density reduction factor:

$$\alpha_T = -\frac{1}{\rho_0}\frac{\partial\rho}{\partial T} \tag{2.4a}$$
$$\alpha_C = -\frac{1}{\rho_0}\frac{\partial\rho}{\partial C} \tag{2.4b}$$

Another consequence of the small density anomalies is that the gravity field, $\mathbf{g}$, can be replaced by that of a sphere of uniform density, thus reducing to:

$$\mathbf{g} = -\gamma r\hat{r} \tag{2.5}$$

where γ is a constant given in s^{-2}.

Replacing eq. 2.3, 2.4 and 2.5 into 2.1b we obtain:

$$\partial_t\mathbf{u} = -\mathbf{u}\cdot\nabla\mathbf{u} - 2\boldsymbol{\Omega}\times\mathbf{u} - \nabla\Pi + \gamma r\hat{r}(\alpha_T\delta T + \alpha_C\delta C) + \frac{\nu}{\rho_0}\nabla^2\mathbf{u} + \frac{1}{\mu\rho_0}(\nabla\times\mathbf{B})\times\mathbf{B}, \tag{2.6}$$

where we introduced all the simplifications that we made so far and introduced the generalized pressure gradient $-\nabla\Pi$. We will come back to the Navier-Stokes equation in the next section.

The Boussinesq approximation has another more physically relevant consequence. In order for one to be able to write eq. 2.3, one must assume that, at all times, both the temperature and the concentration must only slightly depart from static (constant in time) underlying fields T_S and C_S. Furthermore, these fields must not be arbitrary, but reflect the average state of the convection as buoyancy driven motion should only occur for deviations from these states. It is usual to chose profiles that are spherically symmetric and obey the temperature and composition equations 2.1c and 2.1d for a given set of sources. Other choices, as is the case of a homogeneously distributed concentration field are based upon observations.

In DRS we assume a static thermal state described by a radial temperature profile $T_S(r)$ that can be one of the values described in Table 2.1. Note that

Opt.	Equation	Description
0	$T_S(r) = T_0 + \beta(r_i r_o/r - r_i)$	The conductive temperature profile used in [Christensen et al., 2001]. See Section B.1.
1	$T_S(r) = T_0 - \beta r^2/2$	This profile closely follows the adiabat [Labrosse et al., 1997, Davies and Gubbins, 2011] and alludes to the possibility that at least a fraction of the energy available to planetary dynamos is due to radiogenic heat release.

Table 2.1: Possible temperature profiles, in K, hardcoded into DRS. First column refers to the option values used in the code.

the conductive profile exactly obeis eq. 2.7c with $S_T = 0$ and that the quasi-adiabatic profile emulates a non-zero volumetric heat source.

Constants T_0 and β have values that depend on the profile and can be written, in dimensional form, as a function of the temperature at the top (T_o) and bottom (T_i) boundaries. Opt. 0 has $T_0 = T_o$ and $\beta = (T_i - T_o)/d$ and refers to the same temperature profile used in the first dynamo benchmark exercise [Christensen et al., 2001] whereas Opt. 1 has $T_0 = (T_i r_o^2 - T_o r_i^2)/(r_o^2 - r_i^2)$ and $\beta = (T_o - T_i)/(r_o^2 - r_i^2)$, thus closely following an adiabatic temperature profile [Labrosse et al., 1997, Davies and Gubbins, 2011].

2.1.2 The adimensionalisation

DRS solves the set of equations 2.1a-2.1e in adimensional form. To perform the adimensionalisation we made use of the following set of scales. Lengths are measured in units of d and times in units of d^2/ν. The temperature is measured in units (T^*) that depend on the radial profile: $T^* = \beta d\nu/\kappa$ for the case of the conduction profile (Opt. 0); and $T^* = \beta d^2\nu/\kappa$ for the case of

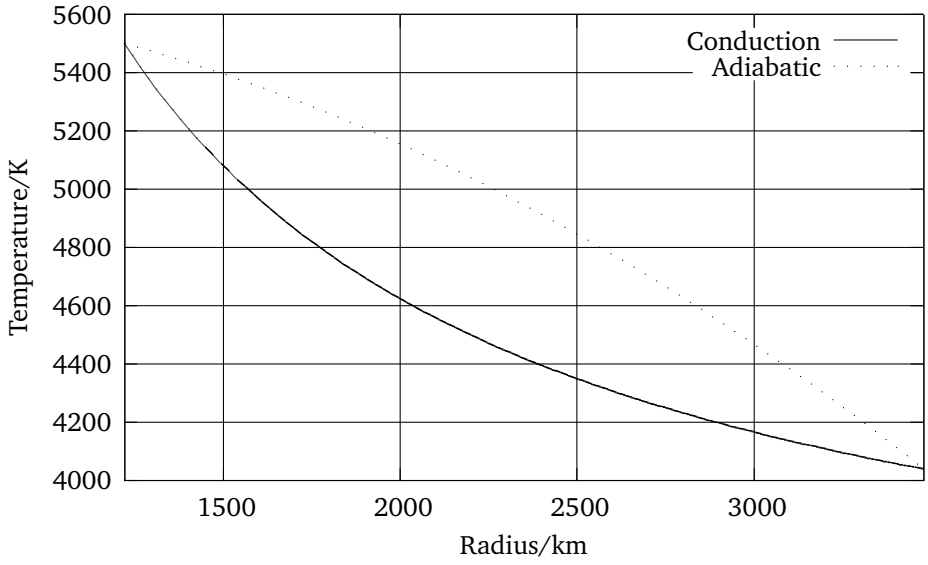


Figure 2.2: Possible temperature profiles, in K, hardcoded into DRS.

the quasi-adiabat (Opt. 1). Note that T^* must always have units of temperature so the units of β must depend on the specific profile. Finally, the magnetic flux density is measured in units of $\nu(\mu\varrho)^{1/2}/d$, with μ, the magnetic permeability [Ardes et al., 1997]. The equations of motion for the velocity vector $\mathbf{u}$, the heat equation for the deviation Θ from the radial temperature profile, and the equation of induction for the magnetic flux density $\mathbf{B}$ are then given by:

$$\nabla \cdot \mathbf{u} = 0, \qquad \nabla \cdot \mathbf{B} = 0, \tag{2.7a}$$

$$(\partial_t + \mathbf{u} \cdot \nabla)\mathbf{u} + \tau \mathbf{k} \times \mathbf{u} = -\nabla\pi + R_T \Theta \mathbf{r} + \nabla^2 \mathbf{u} + \mathbf{B} \cdot \nabla \mathbf{B}, \tag{2.7b}$$

$$\partial_t \Theta + \mathbf{u} \cdot \nabla(\Theta + T_S) = \frac{1}{P_T} \nabla^2 \Theta, \tag{2.7c}$$

$$\nabla^2 \mathbf{B} = P_m(\partial_t \mathbf{B} + \mathbf{u} \cdot \nabla \mathbf{B} - \mathbf{B} \cdot \nabla \mathbf{u}), \tag{2.7d}$$

where all gradient terms in the equation of motion have been combined into $\nabla\pi$. The dimensionless parameters in our formulation are the Rayleigh number R, the Coriolis number τ, the thermal Prandtl number P_T and the magnetic Prandtl number P_m,

$$R_T = \frac{\alpha_T \gamma d^4 T^*}{\nu^2}, \quad \tau = \frac{2\Omega d^2}{\nu}, \quad P_T = \frac{\nu}{\kappa}, \quad P_m = \frac{\nu}{\lambda}. \tag{2.8}$$

2.2 Adding composition

Two major changes had to be made in order to accommodate compositional convection in the code. The first relates to solving a new equation for the com-

position anomalies χ around a static profile of composition $C_S(r)$.

$$\partial_t \chi + \mathbf{u} \cdot \nabla(\chi + C_S) = \frac{1}{P_C} \nabla^2 \chi, \tag{2.9}$$

This equation now depends on one parameter $P_C = \nu/D$, with D being the diffusivity of the composition field.

The second change relates to the buoyancy in the Navier-Stokes equation. The buoyancy term now reads

$$R_T \Theta \mathbf{r} + R_C \chi \mathbf{r}, \tag{2.10}$$

where R_T is the usual thermal Rayleigh number and now R_C is a new compositional Rayleigh number defined as

$$R_C = \frac{\alpha_C \gamma d^4 C^*}{\nu^2}. \tag{2.11}$$

The value C^* is a scaling for the composition anomaly that can be chosen arbitrarily or, as is the case of the temperature, based on the profile C_S.

2.2.1 Testing the compositional convection implementation

A simple way to test the implementation of the compositional convection is to simulate the codensity approximation, that is, assume that the thermal and compositional diffusivities are the same. The buoyancy term in equation 2.7b can be written as

$$R\zeta \mathbf{r} = R_T \Theta \mathbf{r} + R_C \chi \mathbf{r}, \tag{2.12}$$

where ζ is the codensity and plays the role of the temperature in the pure thermal convection case, and R is a new parameter playing the role of a Rayleigh number. Then equations 2.7c and 2.9 can be multiplied by R_T/R and R_C/R respectively and be combined into one equation that solves for the temporal evolution of ζ. In this formulation, results obtained with the pure thermal code with $R_T = 2R$ and a given P_T (effective codensity) should be exactly the same as in the thermo-compositional convection case with $R_T = R_C = R$ and $P_C = P_T$ (simulated codensity). Figure 2.3 shows a comparison of the kinetic energy for the two cases described above.

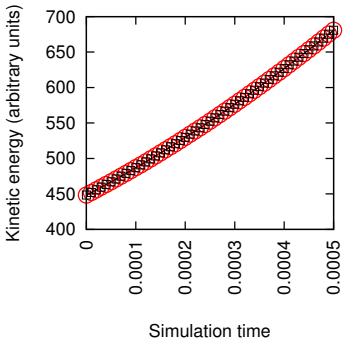
2.3 Boundary conditions

Boundary conditions are imposed for each quantity being evolved through equations 2.7b-2.7d and 2.9. For the temperature and composition, the boundary conditions are applied to the quantity being evolved plus the underlying static radial profile. The possible boundary conditions and the option values for the config file (see section 1.3.2) are shown in the following sections.

2.3.1 Boundary conditions on the flow

Besides obeying a non penetration condition, at each boundary the velocity field obeys one of the following conditions:

- Free slip (0). Sets the radial gradient of the flow to 0 at the boundary.
- No slip (1). Sets the flow to 0 at the boundary.



step	Effective	Simulated
0	0.448474977D+03	0.448474977D+03
50	0.467204101D+03	0.467204101D+03
100	0.487015790D+03	0.487015790D+03
150	0.507871571D+03	0.507871571D+03
200	0.529741404D+03	0.529741404D+03
250	0.552597964D+03	0.552597964D+03
300	0.576415084D+03	0.576415084D+03
350	0.601167084D+03	0.601167084D+03
400	0.626828411D+03	0.626828411D+03
450	0.653373446D+03	0.653373446D+03
500	0.680776383D+03	0.680776383D+03

Figure 2.3: Evolution of the kinetic energy in the simulated and effective co-density formulations. No differences can be found.

2.3.2 Boundary conditions on the temperature

At each boundary, the temperature obeys one of the following conditions:

- Fixed/constant temperature (0). This fixes the temperature anomalies to be 0 at the boundary. It also fixes the static temperature profile to be a conductive profile.
- Fixed/constant heat flux (1). Fixes the radial gradient of the temperature anomalies to be 0 at the boundaries. It also fixes the static temperature profile to be nearly adiabatic.

2.3.3 Boundary conditions on the magnetic field

At each of the boundaries, the magnetic field obeys one of the following conditions:

- Vacuum (0);
- Pseudo-vacuum (1);

2.3.4 Boundary conditions on the composition

At each boundary, the composition obeys one of the following conditions:

1. Fixed/constant composition (0);
2. Fixed/constant chemical flux (1).

Chapter 3

Discretization and numerical methods

3.1 Poloidal/toroidal decomposition

Being solenoidal vector fields, that is, divergence free vector fields, $\mathbf{u}$ and $\mathbf{B}$ can be represented uniquely in terms of poloidal and toroidal components,

$$\mathbf{u} = \nabla \times (\nabla \times (v\mathbf{r}) + r\nabla \times (w\mathbf{r}) \ , \tag{3.1a}$$

$$\mathbf{B} = \nabla \times (\nabla \times (h\mathbf{r})) + \nabla \times (g\mathbf{r}) \ . \tag{3.1b}$$

Notice that the decomposition is different for the flow and the magnetic field. This difference in decomposition justifies the difference in implementation of the laplacians, spectral-to-real and real-to-spectral transformations in the radial direction. The difference in implementation of the flow decomposition has its justification in that values of the flow coefficients under this expansion are better numerically behaved [Tilgner, 1999].

The scalar fields v, w, g and h can then be decomposed in terms of orthogonal polynomials.

3.2 Polynomial decompositions

One of the main reasons behind the being of spectral codes is that, by expanding all the scalar field in a basis of orthogonal polynomials, one can transform the set of coupled partial differential equations into a set of coupled algebraic equations that can then be solved by standard matrix decomposition/inversion techniques. Furthermore, differentiation becomes an exact procedure under this framework, thus reducing numerical errors.

There are, however, downsides to this approach. Some of the gains in speed and accuracy achieved by using a spectral method are, for very high resolutions, overcome by the need to perform all multiplications in real space and, therefore, all the forward/backward transform overhead. This has obvious consequences for parallelization. Furthermore, care must be taken when truncating the series of polynomials; the non-linear terms in the equations involved couple the large

and modelled scales to the small (sub-grid, un-modelled) scales in a non-trivial manner. This is a normally overlooked problem but a rule of thumb is that for all the cross-scale interactions to be negligible, all of the spectra of all the quantities being evolved should decay by at least three orders of magnitude between the highest and the lowest power.

DRS uses a spectral decomposition in both horizontal and radial direction. Chebyshev polynomials are used in the radial direction and Spherical Harmonics are used in the horizontal. The next subsections are dedicated at the details of these expansions.

3.2.1 Spherical harmonic decomposition

We use 4π normalised spherical harmonics, defined as :

$$Y_\ell^m(\theta,\varphi) = \sqrt{(2\ell+1)\frac{(\ell-m)!}{(\ell+m)!}}\, P_\ell^m(\cos\theta)\, e^{im\varphi}, \tag{3.2}$$

where $P_\ell^m(\cos\theta)$ are the un-normalized Associated Legendre Polynomials (ALP) and

$$\int Y_\ell^m(\theta,\varphi) Y^{*m}_\ell(\theta,\varphi)\mathrm{d}\Omega = 4\pi. \tag{3.3}$$

The polynomials $\bar{P}_\ell^m(\cos\theta) = \sqrt{(2\ell+1)\frac{(\ell-m)!}{(\ell+m)!}}$ are the normalised ALP, which we use throughout the code to construct the spherical harmonics.

> **Warning**
>
> Before version 1.6.0 (MAGIC numbers 101**) we used un-normalised ALP. After and including DRS 1.6.0 we use the new $\bar{P}_\ell^m(\cos\theta)$. This reflects on the imported and exported coefficients. The new code, however, is smart enough to recognise this situation and read coefficients with any of these normalisations. Other codes may not!!!

A scalar function $f(\theta,\varphi)$ can, therefore, be written as:

$$f(\theta,\varphi) = \sum_{l=0}^{L}\sum_{m=-l}^{l} Y_\ell^m(\theta,\varphi) f_l^m. \tag{3.4}$$

Recurrence relations for the Associated Legendre Polynomials

$$(\ell-m+1)P_{\ell+1}^m(x) = (2\ell+1)xP_\ell^m(x) - (\ell+m)P_{\ell-1}^m(x) \tag{3.5}$$

$$\begin{aligned}(1-x^2)\frac{d}{dx}P_\ell^m(x) = \frac{1}{2\ell+1}\big[&(\ell+1)(\ell+m)P_{\ell-1}^m(x)\\ &-\ell(\ell-m+1)P_{\ell+1}^m(x)\big]\end{aligned} \tag{3.6}$$

3.2.2 Chebyshev decomposition

Chebyshev polynomials of the first kind, $y = T_n(x)$, are the real valued functions solving the Chebyshev partial differential equation:

$$(1 - x^2)\frac{\partial^2 y}{\partial x^2} - x\frac{\partial y}{\partial x} + n^2 y = 0, \tag{3.7}$$

where n is an integer number greater than or equal to zero. They can easily be generated by a recurrence relation as follows:

$$T_0(x) = 1 \tag{3.8a}$$
$$T_1(x) = x \tag{3.8b}$$
$$T_n(x) = 2xT_n(x) - T_{n-1}(x) \tag{3.8c}$$

Alternatively they can be defined as the unique polynomials satisfying:

$$T_n(\cos(\theta)) = \cos(n\theta), \tag{3.9}$$

where θ is any angle in the domain $[0 : 2\pi[$ and not the colatitude. From this alternative definition we gather that the Chebyshev polynomials vary in the interval $[-1, 1]$.

In the present code, decompositions in the radial direction are made using Chebyshev polynomials.

First, we decompose the radial domain into a series of N_r adimensional radial nodes r_n (see section 2.1 for the problem setup). Recall that in adimensional terms, the shell thickness has size 1 and the radial coordinate starts at $r_i = \eta/(1-\eta)$, η being the aspect ratio of the shell:

$$r_k = r_i + \frac{1}{2}\left[x_k + 1\right], \tag{3.10}$$

with $x_k = \cos\left(\frac{\pi(k-1)}{Nr-1}\right)$. The Chebyshev coordinate can be written in terms of the radial coordinate as:

$$x_k = 2(r_k - r_i) - 1 \tag{3.11}$$

Notice that this is a mapping between $x_k = \cos(\theta_k)$ and r_k. Figure 3.1 shows a schematic diagram of the positions of the radial nodes for the case of 31 points and an aspect ratio $\eta = 0.35$.

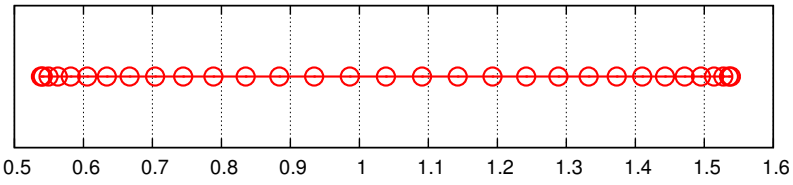


Figure 3.1: Positions of the radial nodes for the case of 31 points and an aspect ratio $\eta = 0.35$

A function f depending on the radial coordinate r can then be written as:

$$f(r) = \sum_n^N f_n T_n(x(r)) \tag{3.12}$$

Normalization

$$\int_{-1}^{1} \mathrm{d}x T_n(x) \frac{1}{\sqrt{1-x^2}} T_m(x) = \begin{cases} 0 & : \quad n \neq m \\ \pi & : \quad n = m = 0 \\ \pi/2 & : \quad n = m \neq 0 \end{cases} \tag{3.13}$$

3.2.3 Spectra

In spherical geometry, we can define three types of spectrum: a spectrum in terms of azimuthal wave number m, or *m-spectrum*, $\mathcal{R}_m$; a spectrum in terms of the spherical harmonic degree l, or *l-spectrum*, $\mathcal{R}_l$; and a spectrum in terms of the Chebyshev order n, or *n-spectrum*, $\mathcal{R}_n$.

Be consistent when using ℓ vs. l.

For a scalar field, $f(r,\theta,\varphi)$, like the temperature, that can be decomposed in terms of Spherical Harmonics and Chebyshev polynomials as:

$$f(r,\theta,\varphi) = \sum_{n=1}^{Nr_s} \sum_{m=0}^{Np_s/2} \sum_{l=m}^{Nt_s} T_n(r) Y_\ell^m(\theta,\varphi) f_{n,l}^m, \tag{3.14}$$

the total energy of that field is constructed as the volume integral of the square of the field:

$$E = \iiint f(r,\theta,\varphi)^2 d\Omega \tag{3.15}$$

$$= \iiint \sum_{n1,n2=1}^{Nr_s} \sum_{m1,m2=0}^{Np_s/2} \sum_{l1,l2=m1,m2}^{Nt_s} T_{n1} Y_{l1}^{m1} f_{n1,l1}^{m1} T_{n2} Y_{l2}^{m2} f_{n2,l2}^{m2} \tag{3.16}$$

$$= 4\pi \sum_{n1,n2=1}^{Nr_s} \int_{r_i}^{r_o} T_{n1} T_{n2} r^2 \mathrm{d}r \sum_{m=0}^{Np_s/2} \sum_{l=m}^{Nt_s} f_{n1,l}^m f_{n2,l}^m \tag{3.17}$$

$$\tag{3.18}$$

This energy can be expressed on a per index base and, therefore, three types of spectra can be constructed. These are defined as:

$$\mathcal{R}_l = \sum_{n=1}^{Nr_s} \sum_{m=0}^{l} \tag{3.19}$$

$$\mathcal{R}_m = \sum_{n=1}^{Nr_s} \sum_{l=m}^{Nt_s} \tag{3.20}$$

$$\mathcal{R}_n = \sum_{m=0}^{Np_s/2} \sum_{l=m}^{Nt_s} \tag{3.21}$$

See section 4.2.1 for the details on how these quantities are stored in files.

3.3 Crank-Nicholson/Adams-Bashford integration

All of the equations we have to solve are of the form:

$$\dot{f} = \partial_t f(\mathbf{r},t) = \epsilon \nabla^2 f(\mathbf{r},t) + F(\mathbf{r},t). \tag{3.22}$$

The partial derivative on the left hand side has two contributions: $\dot{f}_1$, coming from the term with the laplacian and $\dot{f}_2$, coming from the other terms. We use a Crank-Nicholson integration for $\dot{f}_1$ and an Adams-Bashford integration for the remaining terms.

3.3.1 Crank-Nicholson integration

We start with the Crank-Nicholson integration. Let,

$$\dot{f}_1(\mathbf{r}, t) = \epsilon \nabla^2 f(\mathbf{r}, t). \tag{3.23}$$

We can now discretise in time using index i for the time:

$$\frac{f_1(\mathbf{r}, t_{i+1}) - f_1(\mathbf{r}, t_i)}{t_{i+1} - t_i} = \epsilon \frac{\nabla^2 f(\mathbf{r}, t_{i+1}) + \nabla^2 f(\mathbf{r}, t_i)}{2}. \tag{3.24}$$

Collecting terms at time t_{i+1} o the l.h.s we get:

$$f_1(\mathbf{r}, t_{i+1}) - \frac{\epsilon h_i}{2} \nabla^2 f(\mathbf{r}, t_{i+1}) = f_1(\mathbf{r}, t_i) + \frac{\epsilon h_i}{2} \nabla^2 f(\mathbf{r}, t_i) \tag{3.25}$$

where $h_i = t_{i+1} - t_i$.

This part of the formulation is independent of whether the time-step varies or not.

3.3.2 Adams-Bashford integration

The forcing part of eq. 3.22 reads:

$$\dot{f}_2(\mathbf{r}, t) = F(\mathbf{r}, t). \tag{3.26}$$

We use a variable time-step Adams-Bashford integration method to solve this equation. The Adams-Bashford scheme tells us that, because F is the time derivative of f_2 at time t_i, then:

$$f_2(\mathbf{r}, t_{i+1}) = f_2(\mathbf{r}, t_i) + F(\mathbf{r}, t_i) h_1^* - F(\mathbf{r}, t_{i-1}) h_2^*, \tag{3.27}$$

with

$$h_1^* = \frac{h_i}{2h_{i+1}}(2h_{i-1} + h_i), \tag{3.28}$$

$$h_2^* = \frac{h_i}{2h_{i+1}} h_i. \tag{3.29}$$

Notice that the weights h_1^* and h_2^* will reduce to $3h/2$ and $h/2$ respectively when the h_i are constant and equal to h. Therefore, our implementation supports both variable and constant time-stepping (see section 3.4).

3.3.3 Putting it all together

The iterative solution of eq. 3.22 can now be found by adding eqs. 3.25 and 3.27 to give the final desired form:

$$f(\mathbf{r}, t_{i+1}) - \frac{\epsilon h_i}{2} \nabla^2 f(\mathbf{r}, t_{i+1}) =$$

$$= f(\mathbf{r}, t_i) + \frac{\epsilon h_i}{2}\nabla^2 f(\mathbf{r}, t_i) + F(\mathbf{r}, t_i)h_1^* - F(\mathbf{r}, t_{i-1})h_2^*. \quad (3.30)$$

We can now decompose eq. 3.30 in Spherical Harmonics ($Y_l^m(\theta, \phi)$) and Chebyshev polynomials ($R_n(r)$). The equation for the coefficients $\tilde{f}(l, m, n, t_{i+1}$ can then be written as:

$$\begin{aligned}\mathcal{R}\tilde{f}(l, m, n, t_{i+1}) = \mathcal{R}\tilde{f}(l, m, n, t_i) + \\ + R_n(r)\tilde{F}(l, m, n, t_i)h_1^* - R_n(r)\tilde{F}(l, m, n, t_{i-1})h_2^*. \quad (3.31)\end{aligned}$$

3.4 Time-stepping

Time stepping is controlled via the `delta_t` parameter in the configuration file (section 1.3.2). Positive values indicate the constant time step to be used; negative values indicate the initial time step to be used in an adaptative time step scheme; zero selects an adaptative time stepping scheme with an initial value ten times smaller than the time step recorded with the initial state.

3.4.1 Adaptative time-stepping

When an adaptative time stepping scheme is selected, a new value of h_i has to be selected at each step. Each new vale of h_i is chosen based on a set of conditions based on the state of the system, namely, the values of the flow everywhere, the resolution, magnetic field and Coriolis forces strength. The first three of these conditions is that the time-step h_i cannot be bigger than the CFL numbers proposed by Glatzmeier [Glatzmaier, 1984]. A fourth condition states that the time-step should not be bigger than $1/\sqrt{2\tau}$.

> **Warning**
>
> For stability, when using an adaptative time steping, the number of radial points should be chosen to be close to the number of theta points. The same applies to the spectral resolutions.

Chapter 4

Input/Output

4.1 Inputs

4.1.1 State files

A state file is a text file containing the spherical harmonic coefficients of a scalar field. The flow state files have extension `.pol` and `.tor` for the poloidal and toroidal coefficients respectively. The temperature and composition state files have extensions `.temp` and `.thetac` respectively. The poloidal and toroidal components of the magnetic field are stored in files with extension `.Bp` and `.Bt` respectively. The spectral and physical resolution is described by the acompaning `.par` file (see section 4.1.2)

Each of these files contains only one column of numbers grouped in blocks with size equal to the number of radial points on input. The loop structure for reading and writing is:

```
do j=1, Npi_s
   do l = m0i*(j/2), Nti_s
      do i=1, Nri
```

where Nri is the number of radial points, `m0i` is the azimuthal symmetry, `Nti_s` is the maximum spherical harmonic degree, and `Npi_s` is the maximum number of real and imaginary coefficients for any degree. l plays the role of SH degree, `j` plays the role of SH order and `i` is the radial index starting from the outer boundary.

4.1.2 par files

`.par` files describe the state files. They contain, not only the resolution at which the state files were writen, but also the parameter set that should be used to interpret them. They also contain the simulation time, time step and sampling rate corresponding to the state files, as well as the boundary conditions that were applied when creating the state files and the number of steps since last snapshot. As importantly, they contain the MAGIC parameters that determines which version of the code they were created with and is able to apropriately read them. A typical `.par` file looks like:

```
| magic | lformat | lcalc | lheat | lvel | lmag |
  10104       1       4       0       1     0
|  eta   | Prandtl |   Taylor   |  Rayleigh   |  magPr  |
0.350E+00 0.100E+01 0.4000000E+07 0.1000000E+06 0.500E+01
|  Nr  |  Nt  |  Np  |  Nr_s  |  Nt_s  |  Np_s  | lsymm |  m0  |
   33     64     128     33       42       84       0       1
| delta_t | nsteps | transient | sampling_rate | step | time | drift |
 0.500E-04   10000       0           100         000   0.0E+0  0.0E+0
'A useful comment.
```

4.2 Outputs

DRS outputs to several files. At the end of each run, the states of the flow, field and temperature are dumped to file. The spectra is also saved at this point.

During the run, DRS saves a number of quantities at the frequency specified by `sampling_rate`.

4.2.1 Quantities saved at the end of the run

Spectra

Spectra are saved at the end of each run. We save two spectra in terms of spherical harmonic degree (extension `.lspec`) and order (extension `.mspec`)and a third spectrum in terms of Chebyshev polynomial order (extension `.nspec`). Each file contains columns representing the spectra of the flow, temperature, composition and field if it varies with time.

- **c1** degree/order;
- **c2** temperature;
- **c3** composition (will only be present for compositional runs);
- **c4** flow;
- **c5** field (optional).

See section 3.2.3 for the definitions of each of these quantities.

Time averaged kinetic energies per component

These quantities are store in a file with extension `.uaz`.

- **c1** adimensional radius;
- **c2** time averaged mean meridional squared flow;
- **c3** instantaneous mean meridional squared flow;
- **c4** time averaged mean azimuthal squared flow;
- **c5** instantaneous mean azimuthal squared flow.

Radial flow energy

These files have the extension `.ur`.

- **c1** adimensional radius;
- **c2** time averaged surface integrated squared radial flow;

Temperature

These files have the extension `.t`.

- **c1** adimensional radius;
- **c2** time averaged integrated squared temperature anomaly;
- **c3** time averaged integrated temperature anomaly.

Advection

Stores the advective terms and contributions on average over time as a function of the radial coordinate. Resulting files have the extension `adv`.

The time average, horizontally integrated advective heat flux is defined as:

$$Q_a(r) = \iint u_r(r,\theta,\phi)(\Theta(r,\theta,\phi) + T_S(r)) \sin\theta d\theta d\phi, \tag{4.1}$$

where angle brackets (<>) represent an average over the period, Δt, covered by the present run.

The heat transferred by diffusion is:

$$Q_d(r) = \iint \partial_r(\Theta(r,\theta,\phi) + T_S(r)) \sin\theta d\theta d\phi. \tag{4.2}$$

The total heat flux is defined as:

$$Q(r) = r\frac{\partial_r\Theta(r) - PtQ_d(r)}{-\eta/(1-\eta)^2} + 1. \tag{4.3}$$

`.adv` files have the following structure:

- **c1** radius;
- **c2** $< Q_a(r) >$, the advective part of convection, horizontally and time averaged;
- **c3** $< Q_d(r) >$, the diffusive part of convection, horizontally and time averaged;
- **c4** $< Q(r) >$, the total time averaged heat flux.

4.2.2 Frequently probed quantities

Energies

Files with extensions `.eb` and `.ek` store the magnetic and kinetic energies and are appended to every `sampling_rate` steps.

The magnetic energy (`.eb`) files have 10 columns with the following structure in spherical coordinates.

- **c1** time
- **c2** total magnetic energy
- **c3** zonal poloidal equatorially anti-symmetric
- **c4** zonal toroidal equatorially symmetric
- **c5** non-zonal poloidal equatorially anti-symmetric
- **c6** non-zonal toroidal equatorially symmetric
- **c7** zonal poloidal equatorially symmetric
- **c8** zonal toroidal equatorially anti-symmetric
- **c9** non-zonal poloidal equatorially symmetric
- **c10** non-zonal toroidal equatorially anti-symmetric

Columns 3 to 6 start with the dipole's energy. Columns 7 to 10 starts with the quadrupole.

The kinetic energy files have 12 columns with the following structure in spherical coordinates.

- **c1** time
- **c2** total kinetic energy
- **c3** zonal poloidal equatorially symmetric
- **c4** zonal toroidal equatorially symmetric
- **c5** non-zonal poloidal equatorially symmetric
- **c6** non-zonal toroidal equatorially symmetric
- **c7** zonal poloidal equatorially anti-symmetric
- **c8** zonal toroidal equatorially anti-symmetric
- **c9** non-zonal poloidal equatorially anti-symmetric
- **c10** non-zonal toroidal equatorially anti-symmetric
- **c11** helicity in the northern hemisphere
- **c12** helicity in the southern hemisphere

Coefficients

Files storing probed coefficients have extension `.koeu` for the flow and `koeb` for the magnetic field. These coefficients are stored at every `sampling\_rate` steps, not just at the end.

Flow coefficients are stored in files with extension `.koeu`. Coefficients are taken to be at half radius. Azimuthal symmetry is taken into account via $m0$ which represents the common factor of all azimuthal wave numbers. Most cases will have $m0 = 1$. `.koeu` files have the following structure:

- **c1** time;
- **c2** $\Re(tor)$, l=1,m=0, solid body rotation;
- **c3** $\Re(pol)$, l=m0, m=m0, Real part of the sectoral poloidal flow of degree $m0$;
- **c4** $\Im(pol)$, l=m0, m=m0, Imaginary part of the sectoral poloidal flow of degree $m0$;
- **c5** $\Re(pol)$, l=2*m0, m=2*m0;
- **c6** $\Re(tor)$, l=2*m0+1, m=2*m0.

Magnetic field coefficients are stored in files with extension `.koeb`. Coefficients are taken to be at half radius. Azimuthal symmetry is taken into account via $m0$ which represents the common factor of all azimuthal wave numbers. Most cases will have $m0 = 1$. `.koeu` files have the following structure:

- **c1** time
- **c2** $\Re(tor)$, l=1, m=0, toroidal dipole;
- **c3** $\Re(pol)$, l=m0, m=m0, Real part of the sectoral poloidal field of degree $m0$;
- **c4** $\Im(pol)$, l=m0, m=m0, Imaginary part of the sectoral poloidal field of degree $m0$;
- **c5** $\Im(pol)$, l=m0+1, m=m0, Imaginary part of the last tesseral poloidal field of degree $m0$;
- **c6** $\Re(pol)$, l=2*m0, m=2*m0;
- **c7** $\Re(pol)$, l=2*m0+1, m=2*m0;
- **c8** $\Re(tor)$, l=m0, m=m0;
- **c9** $\Re(tor)$, l=m0+1, m=m0;
- **c10** $\Re(pol)$ l=1, m=0 poloidal axial dipole.
- **c11** $\Re(pol)$ l=2, m=0 poloidal axial quadropole.

CFL numbers

These files have the extension `.cfl`.

- **c1** time;
- **c2** based on radial flow;
- **c3** based on horizontal flow;
- **c4** based on the Coriolis force.

Nusselt number

The Nusselt number is here defined as

$$Nu = \frac{\partial_r (T + \Theta)}{\partial_r T} \tag{4.4}$$

and is probed both at the CMB and the ICB. It is stored in files with the extension `.nu` with the following structure:

- **c1** time;
- **c2** Value at the ICB;
- **c3** Value at the CMB.

Angular momentum

The angular momentum is computed as:

$$L_x = \int -r * (u_\phi \cos\phi \cos\theta + u_\theta \sin\phi) \mathrm{d}V \tag{4.5}$$

$$L_y = \int r * (u_\theta \cos\phi - u_\phi \sin\phi \cos\theta) \mathrm{d}V \tag{4.6}$$

$$L_z = \int r * u_\phi \sin\theta \mathrm{d}V \tag{4.7}$$

Files storing the angular momentum have extension `.am` and have the following structure:

- **c1** time;
- **c2** L_x;
- **c3** L_y;
- **c4** L_z.

Other files

- `.dissB`
- `.dissu`
- `.u\_mid`
- `.uzon`

Appendix A

Utilities

A.1 drs-setup

`drs-setup` is a tool to generate configuration files for the dynamo program. It asks you for values for the whole required set of parameters and provides options. It writes to a file called `drs.conf` if it does not exist or asks for an alternative name/overwrite if it does.

A.2 drs-spectra

`drs-setup` is a tool to generate the spectra files described in section 4.2.1 from state files. It should be invoked as:

```
$ drs-spectra <state-name>
```

where `<state-name>` is the basename of the respective `.par` file without the extension.

A.3 drs2dx

`drs2dx` is a tool to output the specified quantities in a format that opendx can understand. `drs2dx` reads an input file called drs.dx.in with the following format:

```
# Basename of the input file.
# It will also be used as the basename of the output files.
io_calc_file_in = e035p34h92

comment = 'A nice comment.'

## Resolution for the render
# Radial
Nr = 33

# Meridional
```

```
Nt = 64

# Azimuthal
Np = 129

# Makes a decision about what to render.\n
# Numbers are coded as:\n
#
# a b c d e
# | | | | |>e - component 1, 2 or 3 for vectors, irrelevant for scalars
# | | | |> d - coordinate system or stream lines
# | | |> c - quantity to be ploted
# | |> b - curl, gradient or divergence or 0
# |> a - scalar product with selection or 0
#
#
#  e = 1, 2 or 3 for first second or third coordinate or meridional, azimuthal and po
#      1 or 2 for total or anomaly scalar fiels
#  d = 1, 2 or 3 for cartesian (x,y,x), spherical (r,t,p) or cyllindrical (s, p, z) c
#  c = 1 for the flow
#      2 for the magetic field
#      3 for the temperature field
#      4 for the composition field
#      5 for the magetic field outside the core (up to ro+1)
#  b = 1 for the curl
#      2 for the gradient
#      3 for the divergence
#      0 for nothing
#  a = 1 for scalar product with flow
#      2 for scalar product with field
#      0 for nothing
#
# For example, if I want the meridional component of the curl of the flow,
# a=0, b=1, c=1, d=2, e=2 so \a what = 01122
what = 1

# The type of rendering to do.
# 1 for volume render
# 2 for meridional cut
render_type = 1

# For cuts, it tells drs2dx where the data is to be taken.
# For volume renders it is unused.
where = 0.0d0
```

A.4 drs-state-change

Modifies the state on input. It takes two parameters on the command line: the first is the base name for the state; the second is an integer indicating which

transformation to perform. At this point the only transformation encoded is the removal of the toroidal zonal flow, indicated by the value “0”.

A.5 drs-state-average

Averages a set of states. It reads a configuration file `state-average.in` of the form:

```
# Add the names of states here, one per line.
# First state defines the spatial resolution of the average.
# Any line starting with '#' or '*' is disregarded.
# An example follows:
e035p1t2r100000m1p5test.1
e035p1t2r100000m1p5test.2
```

Each line corresponds to a state. At present we perform only simple averages of the states. in the future we want to include proper time averaging.

A.6 Utility scripts

A.6.1 compressdata/uncompressdata

These are probably the most used scripts at the moment. They compress and uncompress the files of the state using `gzip`.

A.6.2 drs_plot.sh

A.6.3 drs_cleanRun.sh

Removes files with extension adv am cfl dissu ek koeu lspec mspec nspec nu t uaz u_mid ur uzon

A.6.4 latestNormalizedSpectra.sh

Computes and saves the latest normalized spectra of the flow, temperature and composition.

A.6.5 resubmitJob.sh

Resubmits a job to either a pbs queue or a shell

A.6.6 rebuildTimeSeries.sh

Concatenates the files for the energy, etc. for each run into files for the whole set.

A.6.7 drs_plot.sh

Simplifies plotting equatorial slices of the given quantities.

A.6.8 stopRun.sh

Cleanly stops a run and removes the jobs from any queue they may be in.

Appendix B

Benchmarks

B.1 Christensen 2001

Christensen et al. [2001] propose a series of quasi-static solutions to the convective and dynamo problems. For a series of cases they provide global and pointwise values with confidence intervals any numerical code must reproduce. In particular, their case 1.a represents a dynamo solution. This solution has the structure presented in figure B.1.

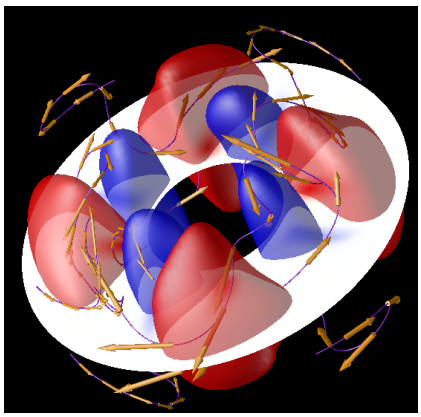


Figure B.1: Structure of the benchmark solution for case 1.a of [Christensen et al., 2001]. The dominant flow mode is of SH order 4, as is the patern of temperature. The whole structure rotates about the system's rotation axis. In the equatorial plane we ploted the radial component of the flow. Isosurfaces of temperature are in red for values above the static profile and in blue for values below. Instantaneous particle trajectories are shown in purple with velocities overlaied (yellow arrows). The magnetic field is not represented here.

B.2 2013/2014

	Reference Values			Our values
Ek	30.7330	$\pm$	0.020	30.7199
Eb	626.4100	$\pm$	0.40	625.6904
T	0.3734	$\pm$	0.00040	0.3727
up	-7.6250	$\pm$	0.0060	-7.6826
Bt	-4.9289	$\pm$	0.0060	-4.8958

Table B.1: Comparison of the results for a run of our code for the benchmark case 1.a of [Christensen et al., 2001].

Bibliography

M. Ardes, F. Busse, and J. Wicht. Thermal convection in rotating spherical shells. *Phys. Ear. Plan. Int.*, 99(1–2):55 – 67, 1997. ISSN 0031-9201. doi: http://dx.doi.org/10.1016/S0031-9201(96)03200-1. URL http://www.sciencedirect.com/science/article/pii/S0031920196032001.

A. author. *CMake user documentation*, 2014. URL http://www.cmake.org/cmake/help/documentation.html.

U. Christensen, J. Aubert, P. Cardin, E. Dormy, S. Gibbons, G. A. Glatzmaier, E. Grote, Y. Honkura, C. A. Jones, M. Kono, M. Matsushima, A. Sakuraba, F. Takahashi, A. Tilgner, J. Wicht, and K. Zhang. A numerical dynamo benchmark. *Phys. Ear. Plan. Int.*, 128:25–34, 2001.

C. J. Davies and D. Gubbins. A buoyancy profiles for the Earth's core. *Geophys. J. Int.*, 187:549–563, 2011. doi: 10.1111/j.1365-246X.2011.05144.x.

G. A. Glatzmaier. Numerical simulation of stellar convective dynamos. I. The model and method. *J. Comp. Phys.*, 55:461 – 484, 1984.

S. Labrosse, J.-P. Poirier, and J.-L. Le Mouël. On cooling of the Earth's core. *Phys. Ear. Plan. Int.*, 99:1 – 17, 1997. doi: 10.1016/S0031-9201(96)03207-4.

A. Tilgner. Spectral methods for the simulation of incompressible flows in spherical shells. *Int. J. Numer. Meth. Fluids*, 30(6):713–724, 1999.