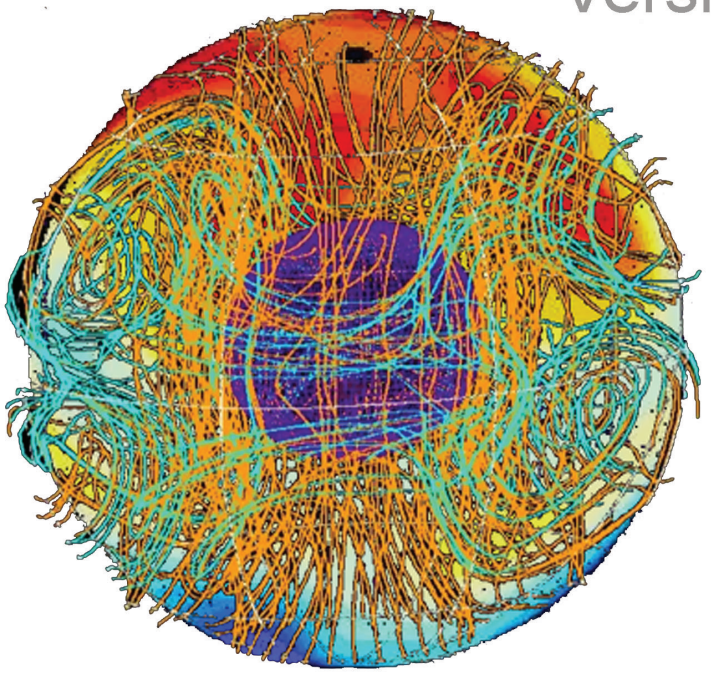
COMPUTATIONAL INFRASTRUCTURE FOR GEODYNAMICS (CIG)

MAG

User Manual
Version 1.0.1



Peter Olson
Wei Mi
Sue Kientz

www.geodynamics.org

MAG

Peter Olson and Wei Mi
Version 1.0.1

April 27, 2007

Contents

List of Figures

Part I

Preface

Preface

About This Document

This document is organized into three parts. Part I consists of traditional book front matter, including this preface. Part II begins with an introduction to MAG version 1.0.1 and its capabilities and proceeds to the details of implementation. Part III provides appendices and references.

The style of this publication is based on the Apple Publications Style Guide (`developer.apple.com/documentation/UserExperience/Conceptual/APStyleGuide/AppleStyleGuide2003.pdf`), as recommended by Python.org (`www.python.org`). The documentation was produced using LyX (`www.lyx.org`) to facilitate the transformation of files from one format to another. LyX is a document processor that encourages an approach to writing based on the structure of your documents, not their appearance. It is released under a Free Software/Open Source license.

Errors and bug fixes in this manual should be directed to CIG Geodynamo Mailing List (`cig-geodyn@geodynamics.org`).

Who Will Use This Document

This documentation is aimed at scientists who prefer to use prepackaged and specialized analysis tools. Users are likely to be experienced computational Earth scientists and have familiarity with basic scripting, software installation, and programming; but are not likely to be professional programmers. Of those, there are likely to be two classes of users: those who just run models and those who modify the source code.

Citation

Computational Infrastructure for Geodynamics (CIG) is making this source code available to you in the hope that the software will enhance your research in geophysics. The underlying Fortran code was donated to CIG in July of 2006. A number of individuals have contributed a significant portion of their careers toward the development of MAG. It is essential that you recognize these individuals in the normal scientific practice by citing the appropriate peer reviewed papers and making appropriate acknowledgements.

The MAG development team asks that you cite the following:

- Olson, P., G.A. Glatzmaier (1993), Highly supercritical thermal convection in a rotating spherical shell: centrifugal vs. radial gravity. *Geophys. Astrophys. Fluid Dyn., 70*, 113-136.
- Olson, P., G.A. Glatzmaier (1995), Magnetoconvection in a rotating spherical shell: structure of flow in the outer core. *Phys. Earth Planet Int., 92,* 109-118.
- Olson, P., G.A. Glatzmaier (1996), Magnetoconvection and Thermal Coupling of the Earth's Core and Mantle. *Phil. Trans. R. Soc. Lond., A354,* 1413-1424.
- Christensen, U.R., J. Aubert (2006), Scaling properties of convection-driven dynamos in rotating spherical shells and application to planetary magnetic fields. *Geophys J. Int. 166*, 97-114.
- Olson, P., U. Christensen, G.A. Glatzmaier (1999), Numerical Modeling of the Geodynamo: Mechanisms of Field Generation and Equilibration. *J. Geophys. Res., 104*, 10,383-10,404.

- Christensen, U., P. Olson, G.A. Glatzmaier (1999), Numerical modelling of the geodynamo: a systematic parameter study. *Geophys. J. Int., 138*, 393-409.
- Christensen, et al. (2001), A numerical dynamo benchmark. *Phys. Earth Planet Int., 128*, 25-34 (benchmark cases).

[Note: there are more recent papers by the same authors.] The developers also request that in your oral presentations and in your paper acknowledgements that you indicate your use of this code, the authors of this code (G. Glatzmaier, U. Christensen, P. Olson), and CIG (`geodynamics.org`).

Support

MAG development was funded by grants from NASA HPC and NSF Geophysics. Continued support of MAG is made possible under NSF EAR-0406751.

Part II

Chapters

Chapter 1

Introduction

Dynamo codes represent a powerful new tool for the quantitative study of a broad range of geophysical processes, ranging from short time-scale phenomena such as magnetic variations, rotational variations, and flow in the core, to long-term phenomena such as magnetic excursions, reversals, superchrons, and the evolution of the core and its thermal and chemical interaction with the mantle. The primary objective of CIG in this area is to provide the Earth Science community with robust, reliable, efficient, flexible, state-of-the-art numerical codes for modeling dynamo processes in the Earth's core and in the interiors of other planets. Another CIG objective is to support graphical- and user-interfaces for these codes that allow Earth scientists to analyze, display, and interpret dynamo code results, and to compare results from the various codes that we support, as well as with geomagnetic, space magnetic, and paleomagnetic data.

1.1 About MAG

MAG is a serial version of Gary Glatzmaier's rotating spherical convection/magnetoconvection/dynamo code, modified by Uli Christensen and Peter Olson. The code solves the non-dimensional Boussinesq equations for time-dependent thermal convection in a rotating spherical shell filled with an electrically conducting fluid. The equations of motion are: the Navier-Stokes equation including Coriolis, Lorentz, Buoyancy, pressure, viscous, and inertial terms; the heat equation including advection, diffusion, and uniform-density heat sources; the continuity equation for velocity and Gauss' law for magnetic field; and the induction equation for the magnetic field.

All variables are non-dimensional (see Appendix A); time scale is viscous diffusion, length scale is shell thickness, temperature scale is boundary temperature difference, magnetic field and electric currents use Elsasser number scaling. A variety of boundary and initial conditions has been selected as options.

Mag uses toroidal-poloidal decomposition for velocity and magnetic field with explicit time steps. Linear terms are evaluated spectrally (spherical harmonics plus Chebyshev polynomials in radius) and nonlinear terms are evaluated on a spherical grid.

Additional technical information is found in [1]-[8].

1.2 Governing Equations

MAG solves the following non-dimensional Boussinesq magnetohydrodynamics equations for dynamo action due to thermal convection of an electrically conducting fluid in a rotating spherical shell (e.g., Olson et al. 1999)[5].

$$E\left(\frac{\partial \boldsymbol{u}}{\partial t}+\boldsymbol{u}\cdot\nabla\boldsymbol{u}-\nabla^2\boldsymbol{u}\right)+2\hat{z}\times\boldsymbol{u}+\nabla P=Ra\frac{r}{r_o}T+\frac{1}{Pm}\left(\nabla\times\boldsymbol{B}\right)\times\boldsymbol{B} \tag{1.1}$$

$$\frac{\partial \boldsymbol{B}}{\partial t}=\nabla\times(\boldsymbol{u}\times\boldsymbol{B})+\frac{1}{Pm}\nabla^2\boldsymbol{B} \tag{1.2}$$

$$\frac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T = \frac{1}{Pr}\nabla^2 T + \epsilon \tag{1.3}$$

$$\nabla \cdot \boldsymbol{u} = 0, \;\; \nabla \cdot \boldsymbol{B} = 0 \tag{1.4}$$

where u is the velocity, B is the magnetic field, T is temperature, t is time, $\hat{z}$ is a unit vector in the direction of the rotation axis, P is pressure, and r is the position vector in the spherical coordinates $r\theta\phi$.

Four basic non-dimensional parameters in 1.1 - 1.4 control the dynamo action. The Rayleigh number represents the strength of buoyancy force driving the convection

$$Ra = \frac{\alpha g_0 \Delta T D^3}{\nu\kappa} \tag{1.5}$$

where α is thermal expansivity, g_0 is gravitational acceleration on the outer boundary at radius R, ΔT is the temperature difference between the inner and outer boundaries, D is shell thickness, ν is kinematic viscosity, and κ is thermal diffusivity. The Ekman number represents the ratio of viscous and Coriolis forces

$$E = \frac{\nu}{\Omega D^2} \tag{1.6}$$

Here Ω is rotation rate. The Prandtl number is the ratio of kinematic viscosity to thermal diffusivity

$$Pr = \frac{\nu}{\kappa} \tag{1.7}$$

and the magnetic Prandtl number is the ratio of kinematic viscosity to magnetic diffusivity λ

$$P_m = \frac{\nu}{\lambda} \tag{1.8}$$

An additional (optional) control parameter is the non-dimensional volumetric heat source (or heat sink) strength ϵ.

Chapter 2

Installation and Getting Help

2.1 Introduction

To test run MAG, download the source package (in the form of a compressed `tar` file) from the Geodynamics Software Packages web page (`geodynamics.org/cig/software/packages`). After unpacking the source, use the `make` utility to build MAG from source, and background execute MAG with the provided benchmark input file.

Advanced users and software developers may be interested in downloading the latest MAG source code directly from the CIG source code repository, instead of using the prepared source package. See the 2.6 section later in this chapter. MAG has been tested on Linux, Mac OS X, and Windows.

2.2 Getting Help

For help, send e-mail to the CIG Geodynamo Mailing List (`cig-geodyn@geodynamics.org`). You can subscribe to the Mailing List and view archived discussion at Geodynamics Mail Lists (`geodynamics.org/cig/lists`).

2.3 System Requirements

MAG requires the following:

- A Fortran compiler, g77 or gFortran.
- For Windows you need to install cygwin (`cygwin.com`).

2.4 Downloading and Unpacking Source

Download MAG from the Geodynamics website (`geodynamics.org`). Click the "software" tab at the top of the page. Then click "Software Packages" and then "Geodynamo." Once you click the MAG link, download the source archive and unpack it using the `tar` command:

```
$ tar xzf MAG-1.0.1.tar.gz
```

If you don't have GNU Tar, try the following command instead:

```
$ gunzip -c MAG-1.0.1.tar.gz | tar xf -
```

2.5 Installation Procedure

2.5.1 MAG File Structure

After unpacking the source, you will find the following directories:

`~/src` Contains the set of FORTRAN source code files with suffix "`.f`." This includes sample grid parameter value files with names like `param32s4.f` for a coarse grid (up to 32 spherical harmonics, 24 radial grid intervals, and 4-fold symmetry in ϕ). A makefile named `makefile`. Sample files with input parameters, `par.XXX`. The case `par.bnch0` is for rotating convection at an Ekman number of 1E-3, starting from a conductive temperature perturbation with imposed perturbation with l=4, m=4, and running for a short time. This is the "benchmark0" test case in Christensen et al., 2001[6]. Another input file is `par.bnch1`, the dynamo "benchmark1" case in Christensen et al. [6].

`~/doc` The directory where you will find this manual and other documentation files.

`~/bench-data` Contains output files `ls.benchX`, `l.benchX`, `g.benchx`, and `d.benchx` obtained with short runs of benchmark0 and benchmark1 on a Linux workstation. Explanations of the contents of these files are found in Appendix C. These data files can be used for comparison with the result obtained by your local run of MAG.

`~/rev-data` Contains output files from runs of the reversal dynamo case; movie files are also included.

`~/idl` This is where the postprocessing IDL (Interactive Data Language) routines reside.

2.5.2 Prepare MAG for Running

1. First you need to create a path for execution of `magx` (below is an example; use your path):

   ```
   printenv PATH $ PATH=$PATH:/your_mag_dir_path $ export PATH
   ```

2. Compile the program using `make` in the source directory, which by default uses the existing `param.f` grid and symmetry

   ```
   $ make
   ```

   Note that makefile uses -g77 or other Fortran compiler, and creates executables, either `magx` (default) or `magxYYsZ`, where `YY`=spherical harmonic truncation and `Z`=longitudinal symmetry.

3. To delete all the object files and executables, type:

   ```
   $ make clean
   ```

2.6 Installing from the Software Repository

The MAG source code is available via a Subversion server at the Geodynamics website (`geodynamics.org`). This allows users to view the revision history of the code, and check out the most recent development version of the software.

NOTE: If you are content with the prepared source package, you may skip this section.

2.6.1 Tools You Will Need

In addition to the usual system requirements, you must have a Subversion client installed in order to work with the source from the CIG software repository. To check whether you have a subversion client installed on your machine, type:

```
$ svn help
```

It should return a usage message. For more information on Subversion, visit the Subversion website (`subversion.tigris.org`).

2.6.2 Download Source from Subversion

To check out the latest version of the software, use the `svn checkout` command:

```
$ svn checkout http://geodynamics.org/svn/cig/geodyn/3D/MAG/trunk MAG
```

where "MAG" is the directory created with the file structure mentioned in 2.5.1.

Chapter 3

Running MAG

3.1 Using MAG

For test-running the code, perform the following steps:

1. Uncompress all files, and create a path (see 2.5.2)

2. Link the grid parameter file to `param.f`,[1] which enters into most subroutines through "include" statements. For example, a grid parameter file named `param32f4.f` (32 spherical harmonics truncation degree, longitude symmetry is 4) is linked using

   ```
   $ ln -sf param32s4.f param.f
   ```

3. Compile the program with:

   ```
   $ make
   $ mv magx magx32s4 (Renaming is optional)
   ```

4. MAG uses a standard input file. Background execute using `par.XXX` as the input file and `.YYY` as the output file's extension:

   ```
   $ magx32s4 <par.XXX >p.YYY &
   ```

   For running with the benchmark input files (`par.bnch0` or `par.bnch1`) , the execution statement should be:

   ```
   $ magx32s4 <par.bnch0 >p.bench0 &
   ```

5. If there is a problem with the input file list, it is often the final three lines; with some systems, a "`$`" may be required at the end.

6. MAG produces a series of output files. For example, when using input file `par.bnch0` (the example in step 4) MAG generates: `l.bench0, ls.bench0, g[i].bench0` and `d[i].bench0`, where i=0,1,2...9. See Appendix C for details on MAG's output files. Compare your output files with the data provided in the directory `~/bench-data/data_bench0`.

   Warning: You must delete, move, or rename all of the output files in the current directory before re-running with the same "output" filename. **Retaining same-named output files in the current directory causes MAG to crash.**

[1] To change grids or symmetry (in `param.f`), MAG needs to be recompiled. The code looks for `param.f`, which needs to be changed for remaking. Examples of `param.f` are `param32s6.f` and `param32s4.f`.

3.2 Changing Parameters

Physical and time-step parameters can be changed in the par-file namelist without re-compiling MAG. See Appendix B for a list of the input parameter names and definitions. Grid parameters must be changed in `param.f` and MAG must be then re-compiled. There are some numerical restrictions on the grid parameter combinations, which are given in Appendix A.

Chapter 4

Postprocessing and Graphics

4.1 Introduction

Once you finish running MAG, you should have a series of output data files. To visualize your results, the MAG software package provides a set of IDL routines found in the directory called `PREFIX/idl`, where `PREFIX` is the directory under which you installed MAG. We should mention here that IDL is a commercial visualization tool by ITT Visual Information Solutions (`www.ittvis.com/idl`). A free IDL compatible program called GDL (`gnudatalanguage.sourceforge.net`) works with MAG's line plot IDL procedure `MAGTS.pro`, but not with the interactive IDL procedures in MAG.

4.2 Time Series and Spectra Plots

Procedures `MAGTS.pro` and `MAGXY.pro` take data from l-files generated by MAG and create time series plots and statistics. This version reads an l-file consisting of 17 time series, the first record being dimensionless time (see Appendix C for details on output file format). Energies and rms magnetic field and velocity are scaled as in MAG; tilt is dipole vector colatitude; pole longitude is dipole vector longitude. Figures 4.1 and 4.2 show the energy time series plot and spectra plot from a time-dependent dynamo.

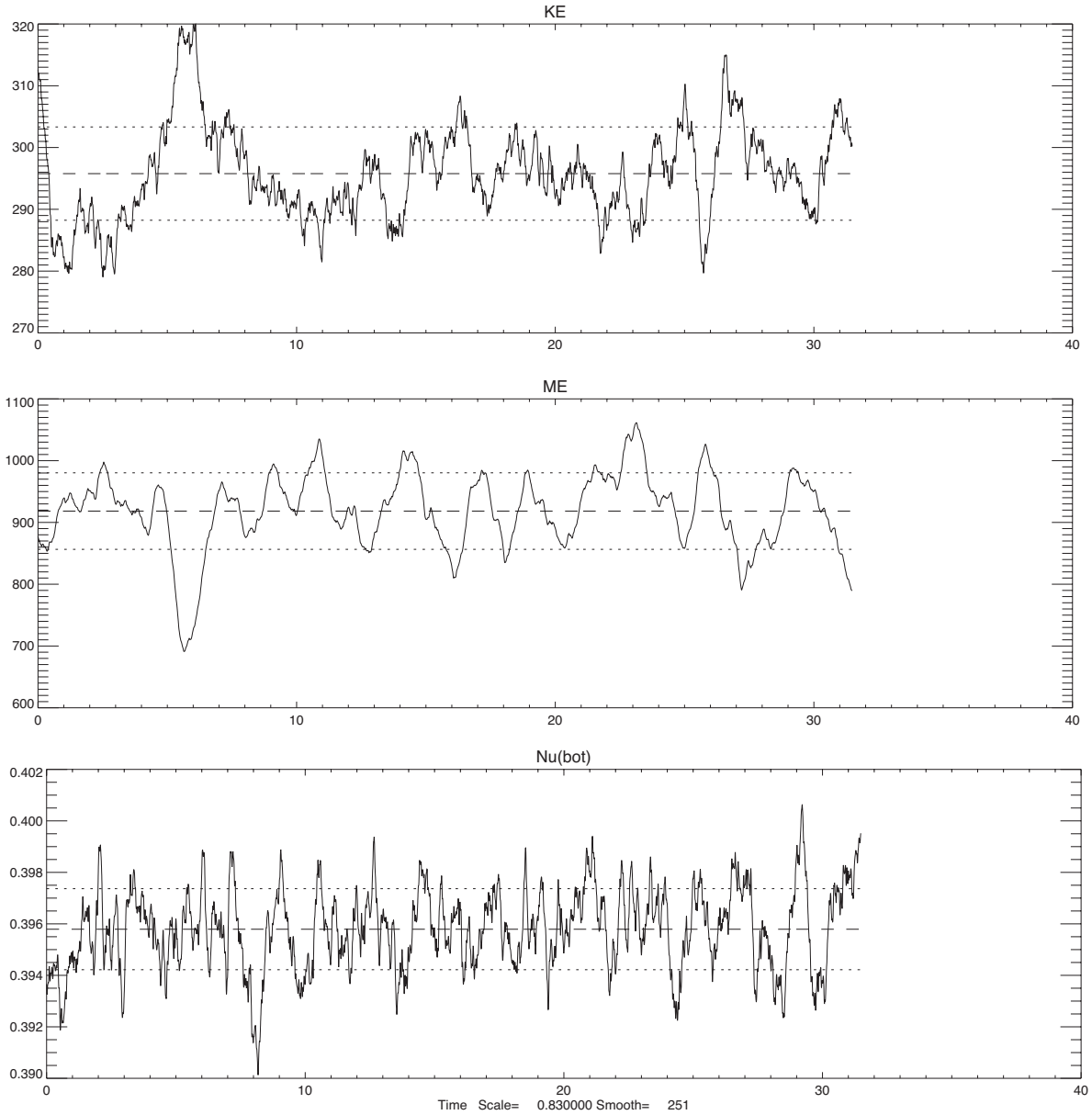


Figure 4.1: Time series plot of energy

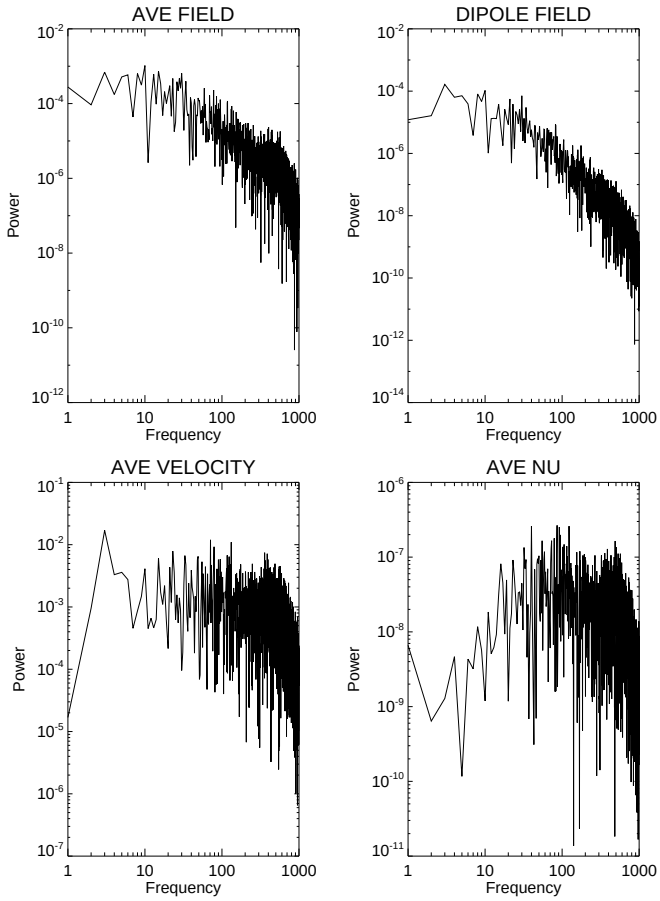


Figure 4.2: Spectra plot of a time depentant dynamo

4.3 Interactive IDL Procedures

`MAGSYM.pro` is an interactive procedure display which results from a g-file produced by MAG. This version uses modified IDL color tables and assumes formatted input. It creates either postscript `.ps` or `.gif` files. If other output file formats are required, you must modify "LABELOUT." MAGSYM has many plot options: map, closeup, equator, slice, etc. Producing each plot is straightforward and accomplished by choosing from the option menu. Figure 4.3 is plotted with the map option.

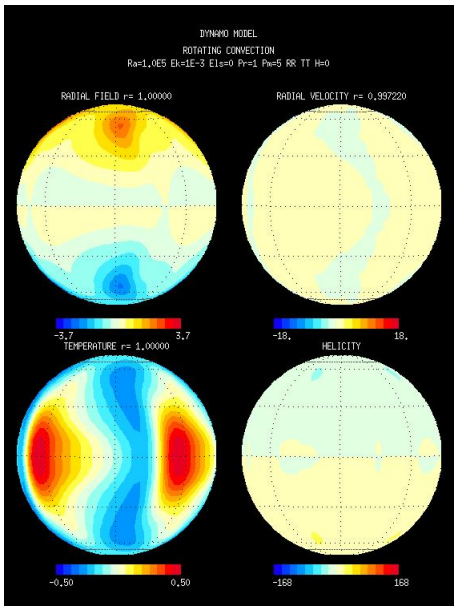


Figure 4.3: IDL figure with Map option

MAGVOL.pro is another interactive IDL procedure to display volume results from rotating convection, magnetoconvection and dynamo calculations (written by P. Olson). It uses g-files produced by MAG (some longitudinal symmetry may be assumed in the g-file). This version uses modified IDL color tables and assumes formatted or unformatted input (it asks for `.gif` but creates `.jpg` files); if other output file formats are required, modifications of "labelout" are required. This version assumes x-window screen graphics; for other graphics devices, change the `set_plot`,'`x`' and `tvrd()` commands accordingly. MAGVOL procedure creates volume-rendered images of temperature, helicity, the z-component of vorticity, kinetic and magnetic energy, joule heating, work by Lorentz forces and buoyancy forces. Figure 4.4 shows a plot of kinetic energy and magnetic energy obtained from a numerical dynamo model.

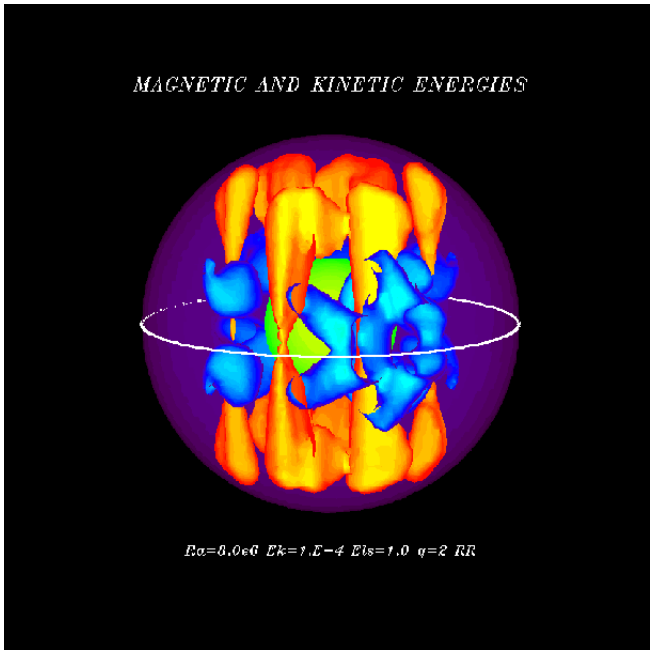


Figure 4.4: Kinetic energy (yellow) and Magnetic energy (blue) plot by the MAGVOL procedure

Chapter 5

Examples

Included in this chapter are two benchmark cases and a reversal dynamo case. We will present the cases with their input parameters and a typical run of the case with output data analysis.

5.1 Benchmark Cases

Historically, these are the cases defined in the benchmark study published in the 2001 paper by Gary Glatzmaier et al. [6]. Case 0 is a benchmark of rotating non-magnetic convection. Case 1 is a dynamo with an insulating inner core co-rotating with the outer boundary. The regions outside the fluid shell are electrical insulators and the magnetic field on the boundaries matches with appropriate potential fields in the exterior that imply no external sources of the field.

In both cases the Ekman number is $E = 10^{-3}$ and the Prandtl number is $Pr = 1$. The Rayleigh number is set to $Ra = 100000$. Note that the definition of the Rayleigh number differs from the one in the published cases [6] by a factor of Ekman number E, i.e., $Ra = \frac{Ra}{E}$.[6] The magnetic Prandtl number is zero in the non-magnetic convection case 0, and is $Pm = 5$ in case 1. The spherical harmonic expansion is truncated at degree $lmax = 32$ and a four-fold symmetry is assumed in the longitudinal direction (`param.f` should be linked to `param32s4.f` when you compile MAG). The input parameter files are `par.bench0` for case 0 and `par.bench1` in case 1; both files reside in the `~/src` directory.

The output files of the benchmark cases are stored in the directory `~/bench-data/data_bench0` and `~/bench-data/data-bench1` respectively. In the following table we see the solutions from MAG agree with the benchmark suggested value with a small difference margin. In both case 0 and case 1, the values listed were obtained with low resolution and a relatively short run of MAG.

	Case 0 Suggested Value	Mag Case 0	Case 1 Suggested Value	Mag Case 1
E_{kin}	58.348 ± 0.050	58.35	30.733 ± 0.020	30.72
E_{mag}			626.41 ± 0.40	627.15
T	0.42812 ± 0.00012		0.37338 ± 0.00040	
μ_ϕ	-10.1571 ± 0.0020	-10.80	-7.6250 ± 0.0060	-7.84
B_θ			-4.9289 ± 0.0060	
ω	0.1824 ± 0.0050		-3.1017 ± 0.0040	

5.2 Reversal Dynamo Case

In this example, we produce a magnetic field reversal using MAG. The input parameter in the source directory for this case is `~/src/par.Rev`. There is no longitudinal symmetry in this case, so when you compile MAG,

use `param32s1.f` linking to `param.f`. The Ekman number is $E = 0.02$, the Prandtl number is $Pr = 1$ and the magnetic Prandtl number is $Pm = 10$. The Rayleigh number is $Ra = 12000$.

5.2.1 Results and Discussions

This case has run on 32-bit and 64-bit Intel processors. Figure 5.1 shows a plot of mean velocity Vrms, mean magnetic field Brms, the axial dipole and the dipole tilt on the outer boundary. It indicated a magnetic field reversal between time steps 25 and 30. Figure 5.2 shows a longer run of MAG, where we see the magnetic field reversed again. At this time, the magnetic field had weakened substantially. In Figure5.3, the top is the pole plot before the second field reversal and the bottom is the pole plot after the second field reversal.

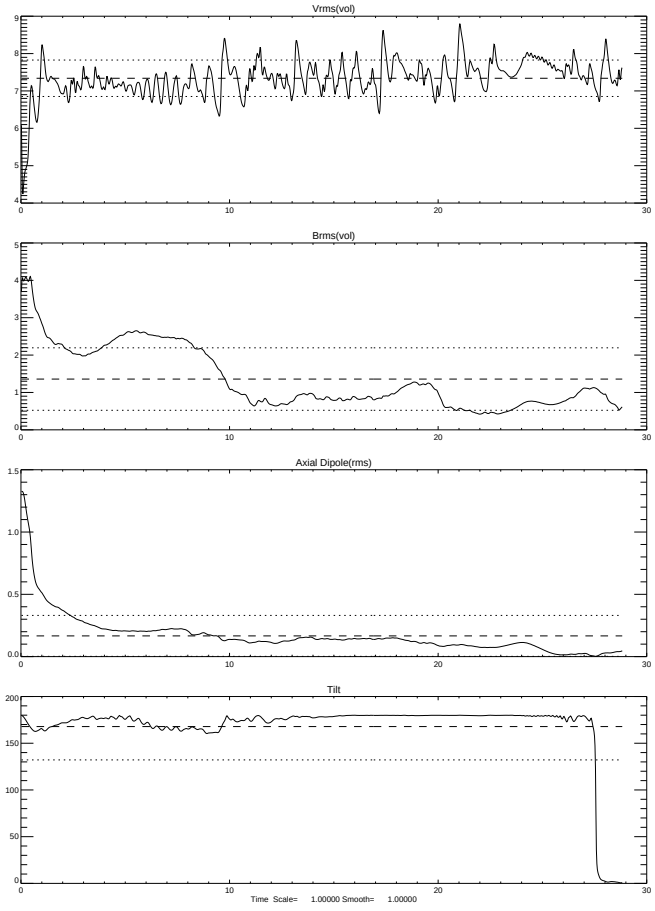


Figure 5.1: Field Plot for Reversal Dynamo Case

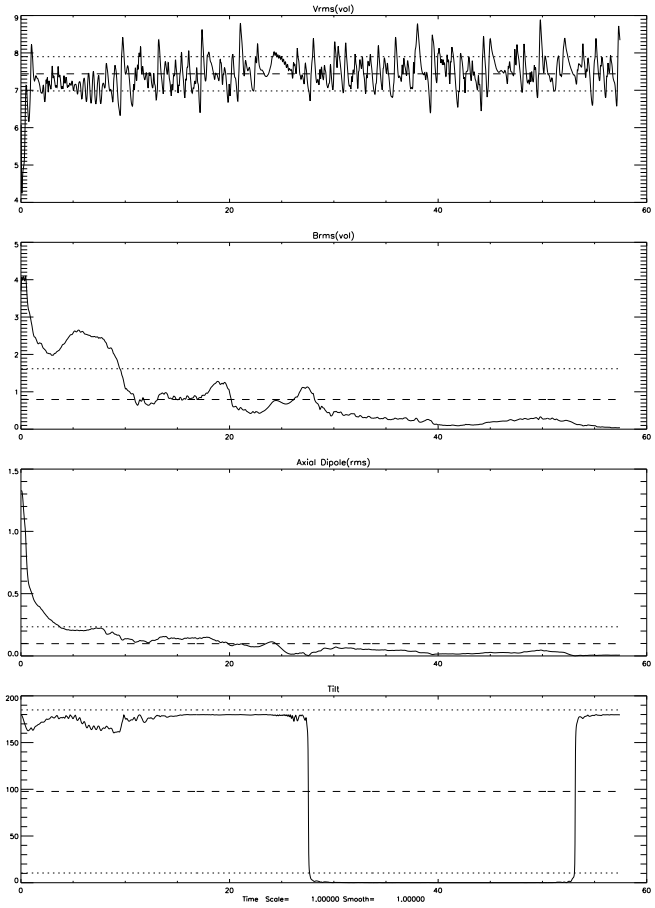


Figure 5.2: Field Plot for Reversal Dynamo Case (longer run)

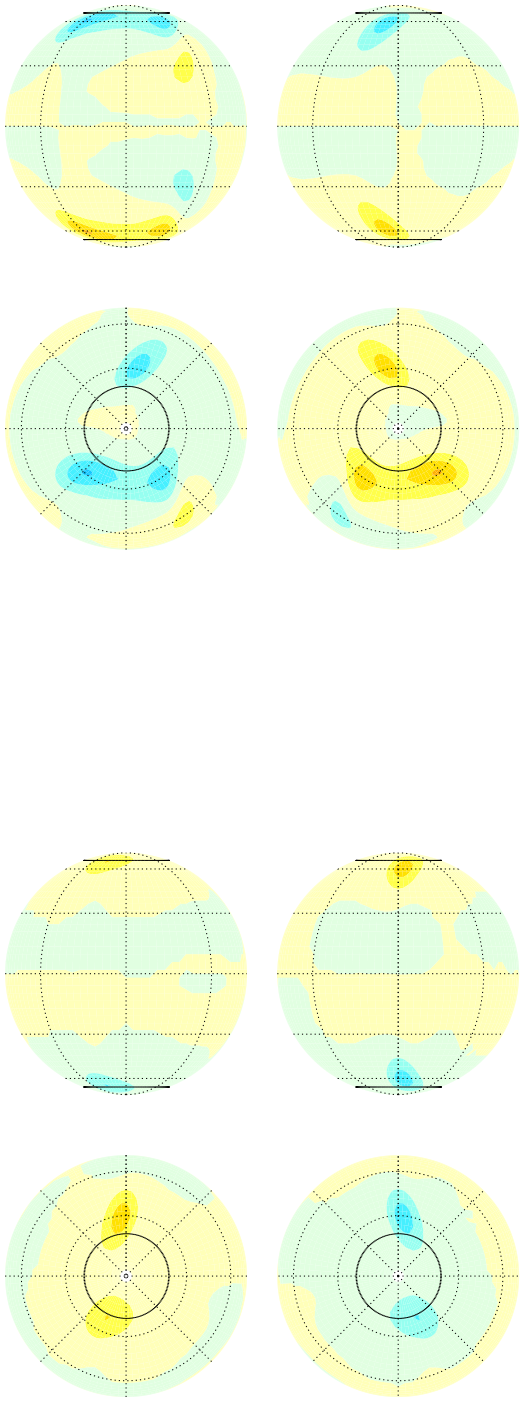


Figure 5.3: Magnetic Field Pole Plot. The top is the pole plot at the begining of the second field reversal; the bottom is the pole plot at the end of the second field reversal.

5.2.2 Creating a Reversal Dynamo Movie

5.2.2.1 Generating Movie Files

MAG has the function to record a movie. The input paramter `imovopt` gives the option to write B_r at the outer surface, the output file is produced with prefix "`mm.`" when `imovopt=100` or the digit at the hundreds is larger than zero. For our reversal dynamo case we examine the field plot and decide to record the first field reversal. As shown the figure 5.4, we pick the time interval to generate the movie file. The input parameter for movie recording is in `~/src/par.revRmv`. We choose restart file d5 as our starting point, and set to record the movie at time $t = 22$. This records 400 frames over an 8 time unit. The sample output files are in `~/rev-data`.

This version of MAG provides an IDL routine `magmovieCIG.pro` (`~/idl/magmovieCIG.pro`), it reads in the movie file `mm.` and displays the magnetic field at the outer surface. This procedure can also create JPG files of the movie-frame images. Figure 5.5shows the first and the last frames of the reversal dynamo movie.

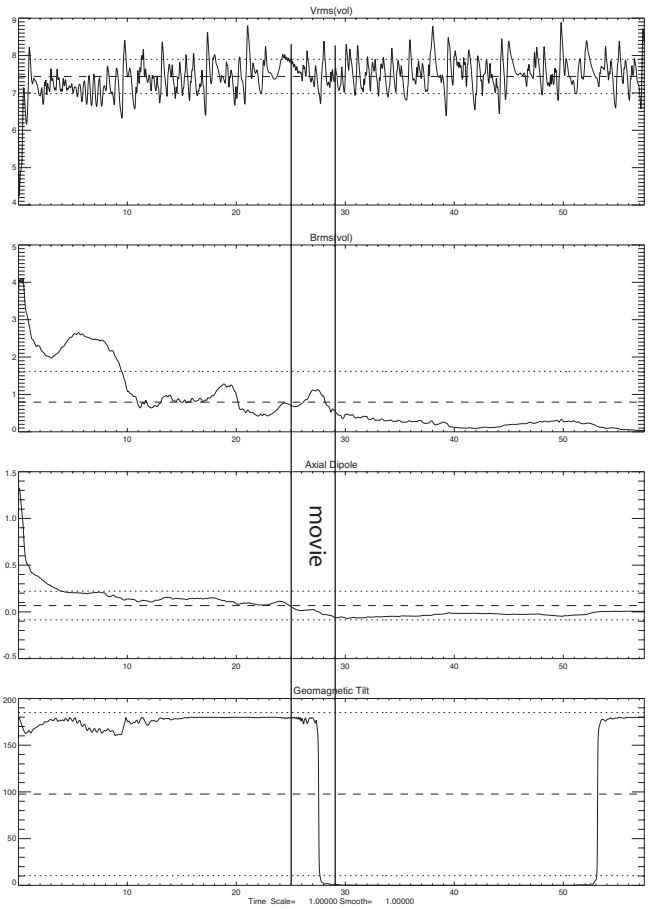


Figure 5.4: Interval to record movie

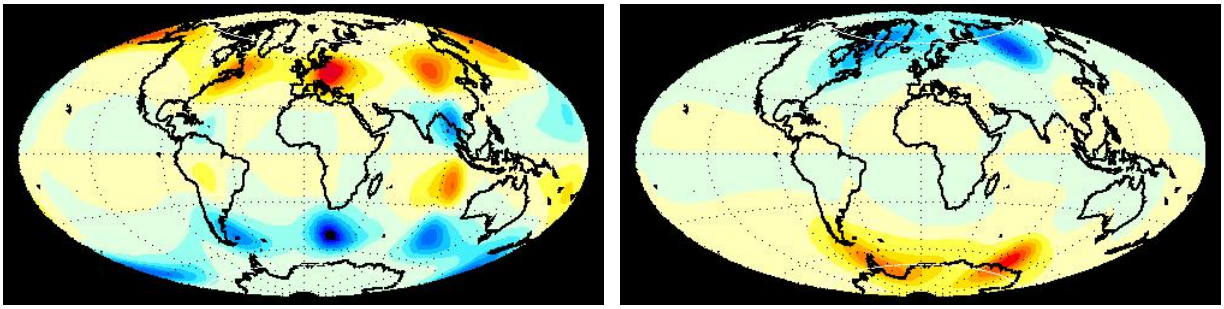


Figure 5.5: Reversal Dynamo Movie. Left: first frame of movie; right: last frame of movie

5.2.2.2 Creating MPEGs with IDL

IDL provides an IDLgrMPEG class that allows the user to save an array of image frames as an MPEG movie. See the IDL Reference Guide (`www.rsinc.com/idl/pdfs/refguide.pdf`).

The MPEG_PUT procedure stores the specified image array at a specified frame index in an MPEG sequence. The MPEG_SAVE procedure encodes and saves an open MPEG sequence. The images can be read using the READ_IMAGE function.

The module `mpg.pro` reads a sequence of images (named `prefix00001.suffix`) and produces an MPEG1 or MPEG2 movie. Supported formats are BMP, GIF, JPEG, PNG, PPM, SRF, TIFF, and DICOM. `mpg.pro` runs with the following parameters:

prefix string (required). A string specifying the prefix of the images to read included the path, e.g., "`/home/data/images/Image`"

suffix string (required). A string specifying the suffix of the images to read, e.g., "`jpg`"

n_start integer (required). An integer specifying the first image in the sequence

n_end integer (required). An integer specifying the last image in the sequence

digits integer (required). An integer specifying the number of digits of the number field in the sequence, e.g., `image00001.jpg` would require digits=5

dims integer (optional). An integer array specifying the size of the output image; if not specified the size of the first image is used

format integer (optional). An integer with values 0 for MPEG1 and 1 for MPEG2 (default is MPEG1)

frame_rate integer (optional). An integer with values

```
1 = 23.976 frames/sec: NTSC encapsulated film rate.
2 = 24 frames/sec: standard film rate.
3 = 25 frames/sec: PAL video frame rate
4 = 29.97 frames/sec: NTSC video frame rate
5 = 30 frames/sec: NTSC drop frame video rate (the default).
6 = 50 frames/sec: double frame rate/progressive PAL
7 = 59.94 frames/sec: double frame rate NTSC
8 = 60 frames/sec: double frame rate NTSC
```

mpeg_file string (optional). A string specifying the output MPEG file (default `outfile.mpg`)

tmp_dir string (optional). A string specifying the temporary directory to use for the temporary image files

To run the IDL movie generator, type in a shell:

```
$ idl
> .compile mpg.pro
> make_mpg, prefix='image', suffix='JPG', n_start=0, n_end=100, digits=3,
            dims=[512,512], frame_rate=2,mpeg_file='mympeg.mpg'
```

A movie file (`magrev#.mpg`) produced by `magmovieCIG.pro` and `mpg.pro` is also included in the `~/rev-data` directory.

5.3 Gauss Coefficients Conversion

Making the output useful to the wider community is one of MAG's primary goals. In this release MAG includes Gauss coefficients in the output file.

Here is some background: To extrapolate the magnetic field between the core, the Earth's surface, and nearby space requires a spherical harmonic representation. MAG uses fully-normalized, complex-valued spherical harmonics for its magnetic field and other variables. These harmonics are packed into one-dimensional arrays for optimal computation and they are also made non-dimensional in MAG.

In contrast, the geomagnetism scientific community has a completely different standard. They ***always*** use real-valued, un-normalized spherical harmonics, whose coefficients are called the Gauss coefficients. The Gauss coefficients – denoted by g(l,m) and h(l,m) respectively, where (l,m) are harmonic degree and order, respectively – are real-valued and have dimensional units, usually expressed in nanoTeslas. For example, the present-day axial dipole field has a Gauss coefficient near g(1,0)=-29,000 nT, the minus sign indicating the field points radially inward in the northern hemisphere. Users from that whole community often just want dynamo model output consisting of standardized files of the Gauss coefficients at various times.

Part III

Appendices

Appendix A

Variables Used in MAG

This is a list of variables and names used in the program set in MAG. The list is in alphabetical order for ease of reference.

adrke axisymmetric toroidal kinetic energy (diagnostic)

ai imaginary unit = complex(0,1)

aj (nlma,nn+1) poloidal magnetic field potential (spectral form); the second index is either the Chebyshev degree (n) or the radial grid point (kc)

ajmat (nn,nn,lmax) LU-decomposed matrix from Chebyshev collocation of toroidal induction equation. Built in ludc, used in amhd.

aleg1 (nlma,ni) Value of associated Legendre function at grid points

aleg2 (nlma,ni) Value of associated Legendre function, multiplied with Gaussian weight, at grid points

aleg3 (nlma,ni) Value of derivative of associated Legendre function multiplied with sin(theta) at grid points

alfilt [INPUT] Filter parameter for B_r in graphics output, see nfilt

alffac [INPUT] Controls the contribution of the (modified) Alfven velocity to the Courant time step limit (see under "courfac"). The modified Alfven velocity is given by v_alfven' = (v_a)^2 / {(v_a)^2 +[pi*(eta+nu)/delx]^2} where v_a = B/sqrt(mu*rho) and delx is the Courant length (either delxr or delxh)

alpha [INPUT] =0 linear terms in the equations are treated fully explicit, =1 linear terms are treated fully implicit, =0.5: Crank-N.

alumn0 Factor for scaling heat flow in output amcke: axisymmetric poloidal kinetic energy

amhd [SUBROUTINE] The "workhorse" of the program: advance solution by nstep time steps

amps [INPUT] can be used to re-scale entropy

ampj [INPUT] can be used to re-scale toroidal magnetic field

ampb [INPUT] can be used to re-scale poloidal magnetic field

ampw [INPUT] can be used to re-scale poloidal velocity

ampz [INPUT] can be used to re-scale toroidal velocity

anorm = sqrt(2/[nn+1])

apome axisymmetric poloidal magnetic field energy

atome axisymmetric toroidal magnetic field energy

b (nlma,nn+1) poloidal magnetic field potential (spectral form, see aj)

bleg1 (lmax) auxiliary array for calculation of aleg1

bleg2 (lmax) auxiliary array for calculation of aleg2

bleg3 (lmax) auxiliary array for calculation of aleg3

bmat(nn,nn,lmax) LU-decomposed matrix from Chebyshev collocation of poloidal induction equation. Built in ludc, used in amhd.

bnlc1 (nja/2,ni) bnlr1 stored in complex form

bnlc2 (nja/2,ni) bnlr2 stored in complex form

bnlc3 (nja/2,ni) bnlr3 stored in complex form

bnlr1 (nja,ni) nonlinear products for updating b (on grid points)

bnlr2 (nja,ni) nonlinear products for updating aj (on grid points)

bnlr3 (nja,ni) nonlinear products for updating aj (on grid points)

bots(0:lmax,0:mmax) [INPUT] harmonic coefficients of prescribed temperature (entropy) on inner boundary

br (nja,ni) = r^2 * B_r on gridpoints

brc br stored as complex array

bscl = dt * radtop^2

bt (nja,ni) = r * sin(theta) *b_theta

btrdt (ni) used in movmout to calculate j_phi

bts (ni,3) used in movmout to calculate j_phi

btc bt stored as complex array

bp (nja,ni) = r * sin(theta) * b_phi

bpc bp stored as complex array

bpeak [INPUT] maximum value of imposed field on boundaries

bpeakbot maximum value of imposed field on inner boundary

bpeaktop maximum value of imposed field on outer boundary

cbr (nja,ni) = r^2 * curl (B) * e_r

cbrc cbr stored as complex array

cbt (nja,ni) = r * sin(theta) * curl (B) * e_theta

cbtc ctr stored as complex array

cbp (nja,ni) = r * sin(theta) * curl (B) * e_phi

cbpc cpb stored as complex array

cheb (nn,nn) cheb(i,j) = value of Chebyshev polynomial i at grid point j

chebi [SUBROUTINE] initialize subroutine chebtf

chebtf [SUBROUTINE] multiple fast Chebyshev transform

clm (lmax,mmax) normalization factors of spherical harmonics

cmb [INPUT] integrated conductivity of thin D"-layer

colat (ni) vector of colatitudes (Gauss points), local array in subroutine prep

courfac [INPUT] factor controlling the time step as fraction of courant advection length. The time step is limited to dt < min(dx/[courfac * v + alffac * v_alfven'])

cvr (nja,ni) = r^2 * curl(v) * e_r

cvrc cvr stored as complex array

db (nlma,nn+1) radial derivative of poloidal magetic potential (spectral form, see aj)

dbdt (nlma,nn,2) time derivative of poloidal magnetic potential b

dcheb (nn,nn) dcheb(i,j) = 1st derivative of Chebyshev polynomial i at grid point j

d2cheb (nn,nn) d2cheb(i,j) = 2nd derivative of Chebyshev polynomial i at grid point j

d3cheb (nn,nn) d2cheb(i,j) = 3rd derivative of Chebyshev polynomial i at grid point j

ddb (nlma,nn+1) 2nd radial derivative of poloidal magetic potential b

ddj (nlma,nn+1) 2nd radial derivatic of toroidal magnetic potential aj

ddw (nlma,nn+1) 0.25 * 2nd radial derivative of poloidal velocity potential w

ddz (nlma,nn+1) 0.25 * 2nd radial derivative of toroidal velocity potential z

djdt (nlma,nn,2) time derivative of toroidal magnetic potential z

dpdt (nlma,nn,2) time derivative of pressure

dsdt nlma,nn,2) time derivative of temperature (entropy)

dw (nlma,nn+1) 0.50 * radial derivative of poloidal velocity potential w

dwdt (nlma,nn,2) time derivative of poloidal velocity potential w

dz (nlma,nn+1) 0.50 * radial derivative of toroidal velocity potential z

dzdt (nlma,nn,2) time derivative of toroidal velocity potential z

delxh (nn) horizontal Courant length squared

delxr (nn) radial Courant length

difamp [INPUT] amplitude of hyperdiffusivity D=D*(1 + difamp *[(l+1-ldif)/(lmax+1-ldif)]^ldifexp) when l>ldif

dipfilt [INPUT] If nfilt>0 multiply axial dipole component of B_r on outer surface by dipfilt in graphics file

dj (nlma,nn+1) radial derivative of toroidal magnetic potential (spectral form, see aj)

dt current time step

dtchck [SUBROUTINE] controls time step

dth Courant time based on horizontal velocity + Alfven velocity

dtmax [INPUT] Upper limit on time step (and initial step)

dtmin Lower limit on time step (stop when dt < dtmin)

dtold Time step of previous iterative step

dtr Courant time based on radial velocity + Alfven velocity

dtstart [INPUT] Initial time step. If =0, dtmax, or when beginning from restart file, the old dt is taken

dvpdr (nja,ni) = d [r * sin(theta) * v_phi]/dr on gridpoints

dvpdrc dvpdr stored as complex array

dvpdp (nja,ni) = d [r * sin(theta) * v_phi]/dphi on gridpoints

dvpdpc dvpdp stored as complex array

dvrdp (nja,ni) = d [r^2 * v_r]/dphi on gridpoints

dvrdpc dvrdp stored as complex array

dvrdr (nja,ni) = d [r^2 * v_r]/dr on gridpoints

dvrdrc dvrdr stored as complex array

dvrdt (nja,ni) = sin(theta) * d [r^2 * v_r]/dtheta on gridpoints

dvrdtc dvrdt stored as complex array

dvtdp (nja,ni) = d [r *sin(theta) * v_theta]/dphi on gridpoints

dvtdpc dvtdp stored as complex array

dvtdr (nja,ni) = d [r * sin(theta) * v_theta]/dr on gridpoints

dvtdrc dvtdr stored as complex array

dw (nlma,nn+1) 0.5 * radial derivative of poloidal velocity potential w

dz (nlma,nn+1) 0.5 * radial derivative of toroidal velocity potential z

escale scaling factor for energies in output

ek [INPUT] Ekman number

enb [OUTPUT] magnetic energy

ens [OUTPUT] thermal energy

enscale [INPUT] in output listing, energies are multiplied by enscale

ent [OUTPUT] total energy

env [OUTPUT] kinetic energy

epsc0 [INPUT] internal heating rate

flmb1 (nlma+..) r-component of (v x B) term

flmb2 (nlma+..) theta-component of (v x B) term

flmb3 (nlma+..) phi-component of (v x B) term

flms1 (nlma+..) r-component of entropy advection term

flms2 (nlma+..) theta-component of entropy advection term

flms3 (nlma+..) phi-component of entropy advection term

flmw1 (nlma+..) r-component of v*grad(v) + Lorentz force term

flmw2 (nlma+..) theta-component of v*grad(v) + Lorentz force term

flmw3 (nlma+..) phi-component of v*grad(v) + Lorentz force term

gauss (ni) vector with Gaussian weighting factors, local array in subroutine prep

gquad [SUBROUTINE] finds zeros and Gaussian weight of associated Legendre function

grafile [CHARACT] file name for data on spatial grid for graphics with prefix "g."; added to outfile set

grav (nn) gravity at radial levels

ib (nn,lmax) pivot array for LU-decomposition of matrix bmat created in sgefa, used in sgesl

ic stepping variable commonly used for steps in colatitude

icour [INPUT] Courant criterion is checked each ICOUR'th time step

idiftype [INPUT] controls radial variation of diffusivity; =0, no variation

ifaxc [13] auxiliary array (factorization) for Chebyshev transform

ifaxf [13] auxiliary array (factorization) for Fourier transform

ifbfrz [INPUT] logical; if .TRUE., do not update magnetic field

ifirst =1 before first call of time-step checking routine, =0 thereafter

iframes [INPUT] write altogether iframes frames on the movie files (see description under imovopt)

ifsfrz [INPUT] logical; if .TRUE., do not update temperature (entropy)

ifvfrz [INPUT] logical; if .TRUE., do not update velocity

ij (nn,lmax) pivot array for LU-decomposition of matrix ajmat created in sgefa, used in sgesl

imagcon [INPUT] <0 imposed poloidal field (l=1,m=0) at ICB >=0 imposed toroidal field (l=2,m=0) at ICB >=10 additionally imposed field at CMB, field is of same sign and amplitude if imagcon=10 and of opposite sign if imagcon=11

imovopt [INPUT] four-digit integer number, controls options for generating movie files.

- **Last digit>0** write B_z, W_z (vortic) and T in the equatorial plane on file with prefix "me.", imovopt=0001
- **2nd last digit>0** write longitudinally averaged B_phi, j_phi and v_phi on file with prefix "ma.", imovopt=0010
- **3rd last digit>0** write B_r at outer surface on file with prefix "mm.", imovopt=0100
- **4th last digit>0** write spherical harmonic coefficients for poloidal field at outer boundary and for velocity potentials at radial level on file with prefix "cc.". (This option works only when any or all of the movie options are turned on, i.e., any of the "m?" files are also produced. imovopt=1000 will not produce a "cc." file.)

imovct counter variable for movie frames

infile [CHARACT INPUT] name of input file for initial values (restart)

init [INPUT] =0 start from dat-file, =1: random initial cond., =-1: hydro. condition from dat-file, magnetic random >=100: initial temperature perturbation in a single mode l,m. Here m is given by the last two digits of init and l by the preceding digits.

ip0 (nn) pivot array for LU-decomposition of matrix p0mat created in sgefa, used in sgesl

iprnt counting blocks in time iteration sequence with printed output created at completion of block

is (nn,lmax) pivot array for LU-decomposition of matrix smat created in sgefa, used in sgesl

is0 (nn) pivot array for LU-decomposition of matrix s0mat created in sgefa, used in sgesl

iscale [INPUT] determines which diffusivity is used for scaling of time, velocity, energy. 1=viscous, 2=thermal, 3=magnetic

istep time step counter (routine amh)

istor counting superblocks in time iteration sequence, upon completion of superblock disk file with data saved

ivfilt [INPUT] Apply filter to v_r at radial level ivfilt and right into first radial position in graphics file; see nfilt

iwp (nn,lmax) pivot array for LU-decomposition of matrix wpmat created in sgefa, used in sgesl

iz (nn,lmax) pivot array for LU-decomposition of matrix zmat created in sgefa, used in sgesl

k2k (nn1) auxiliary array for Chebyshev transform

kc stepping variable commonly used for steps in radius

kcour auxiliary variable for time step checking procedure

kbotb [INPUT] magnetic bottom condition; =1 insulating, =2 perfect condition

kbotv [INPUT] mechanical bottom condition; =1 free, =2 rigid

kbots [INPUT] thermal bottom condition; =1 fixed entropy, =2 flux

kei [SUBROUTINE] calculates kinetic energy

kstep global time step counter

ktops [INPUT] thermal top condition; =1 fixed entropy, =2 flux

ktopb [INPUT] magnetic top condition; =1 insulating, =2 perfect condition

ktopv [INPUT] mechanical top condition; =1 free, =2 rigid

ldif [INPUT] control parameter for hyperdiffusivity, see difamp

ldifexp [INPUT] control parameter for hyperdiffusivity, see difamp

lm stepping variable used to cover all l and m lm = m*(lmax+1)/minc - m*(m-minc)/(2*minc) +l-m+1

lmax maximum harmonic degree, calculated as (nj-1)/3

logfile [CHARACT] file name for continuous log of energies and other data prefix "l."; added to outfile set

lot [PARAM] = 2 * nlma (twice the number of harmonic modes)

lpfile [CHARACT] file name for continuous log of specified values with prefix "lp."; added to outfile set

lsfile: [CHARACT] file name for power spectra of magnetic and kinetic as function of l and m with prefix "ls."; added to outfile set

ludc: [SUBROUTINE] Chebyshev collocation

mclm (nlma) used to unscramble harmonic order m from variable lm

mclma (nlma) = m/minc+1 for given lm (storage order of m)

kei [SUBROUTINE] calculates magnetic energy

minc [PARAM] if >1, minc-fold symmetry in longitude assumed

mmax maximum harmonic order, is the largest integer <= lmax divisible by minc

movafile [CHARACT] file name for movie data (longitudinal averages) with prefix "ma."; added to outfile set

movefile [CHARACT] file name for movie data in equatorial plane with prefix "me." ; added to outfile set

movmfile [CHARACT] file name for movie data in map views with prefix "mm." ; added to outfile set

n, nc stepping variables commonly used for steps over Chebyshev polynomial

ncp [PARAM] = nja/2 used for storage of points in phi in complex array

nfilt [INPUT] Apply filter F(l)=exp(-[l/lfilt]^nfilt) to B_r on outer surface in graphics output file (if nfilt>0 and alfilt>0) When nfilt>0, alfilt<0, apply cos-tapered filtered with cutoff at nfilt and taper width |alfilt|

ngcolat [INPUT] graphics output on each ngcolat'th point in latitude

ngform [INPUT] if .ne. 0, graphics output is written each time a restart file is (finally) written. ngform=1 or -1: formatted graphics file, ngform=2: unformatted for ngform=-1 additional comment lines are inserted (this is to look at the file, not for graphics)

nglon [INPUT] graphics output for each nglon'th point in longitude

ngrad [INPUT] graphics output on each ngrad'th radial level

ni [PARAM] # of grid points in colatide; must be even

nip1 [PARAM] = ni+1

nj [PARAM] # of grid points in longitude; nj/minc must be multiple of four

nja = nj/minc, # of actually needed grid points in phi

njp1 [PARAM] = nj+1

nlaf [PARAM] = lmax+1

nlafp1 [PARAM] = lmax+2

nlm [PARAM] = (mmax+1)*(mmax+2)/2

nlma [PARAM] # of angular modes employed nlma = mmax*(lmax+1)/minc - mmax*(mmax-minc)/(2*minc) + lmax-mmax+1.

nlmpa [PARAM] = nlma + mmax/minc + 1

nlogstep [INPUT] write data on logfile (prefix l.) after each nlogstep steps.

nmaf [PARAM] = mmax+1

nmafa [PARAM] = mmax/minc+1

nn [PARAM] # of radial grid points, nn-1 must be multiple of 4, and contain no prime factors larger than 5

nn1 [PARAM] = nn-1

nn2 [PARAM] = nn-2

nn3 [PARAM] = nn-3

nnp1 [PARAM] = nn+1

nnp2 [PARAM] = nn+2

nnaf [PARAM] # of radial Chebyshev modes, must be <= nn

nnx2 [PARAM] = 2*nn

nplog [INPUT] if >0 write velocity values at specific points of the grid on separate logfile (prefix "lp.") after every nplog steps (for arrays, see vrpoint, vppoint, vtpoint in subroutine amhd for details)

nprnt [INPUT] # of printed output blocks created until next data storage for restart

nps2 [PARAM] = (nn+1)/2

nrp [PARAM] = nja+2 (# of points in phi +2)

ns2 [PARAM] =(nn-1)/2

nstep [INPUT] # of time steps done until next printed output (total # of time steps is nstep*nprnt*nstor)

nstor [INPUT] # of data storages before program termination

ntf [PARAM] =3*nja/2+1, used for Fourier transform array trigsf

ocorevol volume of spherical shell (outer core)

oek = 1. / Ekman number

oekpm = 1. / (Ekman number * Magnetic Prandtl number)

oodt = 1. / dt (inverse time step)

oosscl = 1. / dt

opr = 1. / Prandtl number

outfile [CHARACT INPUT] Name of output files (pre-fixes d.,l.,ls.,g.,me.,ma.,mm., lp. added)

p0mat (nn,nn) LU-decomposed matrix from Chebyshev collocation of pol. equation of motion, l=0-term for pressure. Constructed in ludc, used in amhd

pbar [SUBROUTINE] Calculates value of associated Legendre function

pscale scaling pressure in output

pr [INPUT] Prandtl number

prmag [INPUT] Magnetic Prandtl number

prnt [SUBROUTINE] print diagnostic data

pscl = radtop^2

qi (ni,5) array with various coefficients depending on colatitude (look in subroutine prep, loop "do 32" for details)

qk (nn,16) array with various coefficients depending on radius (look in subroutine prep for details)

ql (nlma,10) various expressions depending on l and m (look in subroutine prep, loop "do 35" for details)

qn (nn,6) Chebyshev integrals

r (nn) vector with radial levels, r(1)=radtop, r(nn)=radbot

ra [INPUT] Rayleigh number

rapr = Rayleigh number / Prandtl number

radbot radius of inner boundary

radratio [INPUT] ratio of inner radius to outer radius

radtop radius of outer boundary

rderiv [SUBROUTINE] radial derivative

rffti [SUBROUTINE] subroutine called in chebi

rstfile [CHARACT] file name for data in spectral form ('restart data') with prefix "d." or "d0.", "d1." ...; added to outfile set

runid [CHAR*64] text identifying the run

rva (nn) auxiliary array used in prep

rvap (nn) auxiliary array used in kei, mei

rvat (nn) auxiliary array used in kei, mei

rvb (nn) auxiliary array used in prep, kei, mei

rvc (nn) auxiliary array used in kei, mei

p (nlma,nn+1) pressure (spectral form)

p00co = 4/sqrt(3)

prep [SUBROUTINE] parameter input, set up auxiliary arrays, set initial conditions, etc.

s (nlma,nn+1) entropy perturbation (spectral form)

sc (ncp,ni) sr stored in complex form

snlc1 (ncp,ni) slnr1 stored in complex form

snlc2 (ncp,ni) slnr2 stored in complex form

snlc3 (ncp,ni) slnr3 stored in complex form

snlr1 (nrp,ni) nonlinear term (radial advection) for updating temperature

snlr2 (nrp,ni) nonlinear term (theta advection) for updating temperature

snlr3 (nrp,ni) nonlinear term (phi advection) for updating temperature

sr (nrp,ni) temperature (entropy) on grid points

s0mat (nn,nn) LU-decomposed matrix from Chebyshev collocation of temperature equation, l=0-term. Constructed in ludc, used in amhd

samp [INPUT] amplitude of initial entropy perturbation

smat (nn,nn,lmax) LU-decomposed matrix from Chebyshev collocation of temperature equation. Built in ludc, used in amhd

sr (nja,ni) entropy on gridpoints

src sr stored as complex array

sscl = dt

stor [SUBROUTINE] store data in restart file

tei [SUBROUTINE] calculates thermal energy

time time

timediff time

tipdipole [INPUT] rotate poloidal dipole term when beginning from restart file

tmovnext auxiliary variable (next output time) for movie file generation

tmovstart [INPUT] time at which to start writing movie-frames on m.*-file

tmovstep [INPUT] time increments for writing movie-frames on m.*-file

tops (0:lmax,0:mmax) [INPUT] harmonic coefficients of prescribed temperature (entropy) on outer boundary

treset [INPUT; LOGICAL] if true reset time and step counter to zero when starting from a stored dataset

trigsc (nn) auxiliary array for Chebyshev transform routine created in chebi, used in chebtf

trigsf (ntf) auxiliary array for Fourier transform routine created in fftrig, used in fourtf

tscale scaling of time in output

up (nja,3) phi-component of velocity in equatorial plane for three consecutive radial levels; used in moveout to calculate vorticity

urdp (nja) derivative dv_r/dphi in equatorial plane; used in moveout to calculate vorticity

vr (nja,ni) = r^2 * v_r on grid points

vrc vr stored as complex array

vp (nja,ni) = c * sin(theta) * v_phi on grid points

vpc vp stored as complex array

vscale scaling of velocity in output

vt (nja,ni) = r * sin(theta) * v_theta on grid points

vtc vt stored as complex array

w (nlma,nn+1) poloidal velocity potential (spectral form)

wpmat (nn,nn,lmax) LU-decomposed matrix from Chebyshev collocation of poloidal equation of motion; built in ludc, used in amhd

wnlc1 (nja/2,ni) wnlr1 stored in complex form

wnlc2 (nja/2,ni) wnlr2 stored in complex form

wnlc3 (nja/2,ni) wnlr3 stored in complex form

wnlr1 (nja,ni) nonlinear products for updating w (on grid points)

wnlr2 (nja,ni) nonlinear products for updating z (on grid points)

wnlr3 (nja,ni) nonlinear products for updating z (on grid points)

work (lot,nnp2) work array used in Fourier and Chebyshev transforms

wsave (nn) auxiliary array used for Chebyshev transform

wscl = dt * radtop^2

y00 = 1/sqrt(4*pi)

z (nlma,nn+1) toroidal velocity potential (spectral form)

zscl = dt * radtop^2

zmat (nn,nn,lmax) LU-decomposed matrix from Chebyshev collocation of toroidal equation of motion; built in ludc, used in amhd

Appendix B

MAG Input File Format

Introduction

This is an overview of the components of the code, input parameters, structure of output files, etc. MAG expects Unix-styled ASCII files (i.e., no carriage-return character following new line character) for all input files. This can be a nuisance in DOS/Windows systems. You may want to find a text editor that can write Unix-style ASCII files. All parameters are in non-dimensional units unless specified.

Input Parameters

Parameters have pre-defined (default) values. They are read through a namelist in the subroutine "prep."

INPUT, OUTPUT, STEPPING CONTROL, INITIALIZATION OF THE RUN

outfile Name of output files (prefixes `d., g., l., ls., me., ma., mm.`, are added)

infile Complete name of file from which initial values are read (restart file)

runid Arbitrary text of up to 64 characters to describe the model

init Set 1 to start from scratch (random noise initial condition); set 0 to start from a previous result obtained on the same grid which has been written into a file named d[0-9].<name> set to a value >= 100 to start from an initial temperature perturbation of one given mode l,m. Here, m is given by the two last digits of init and l by the preceding digits; for example init=606 means that a temperature perturbation of l=6 and m=6 is imposed.

samp Amplitude of initial perturbation (whether random or single mode)

nstep Do one block of nstep time steps before producing a summary printout of some diagnostics standard output; nstep should be an even number

nprnt Do one "superblock" consisting of nprnt blocks of nstep time steps each, before saving all data in file 'd[0-9].name'. If nstor=1, there is no number added after the 'd'; if nstor>1 the number is incremented by one for each new superblock, starting with zero.

nstor Do nstor "superblocks" consisting of nstep*nprnt time steps before terminating the process. The total number of time steps is nstep*nprnt*nstor; nstor must be <=10.

ngform Write data at grid points for graphics processing and other post-processing (programs column.f diagnos.f) into file 'g[0-9].<name>' each time a superblock is written.

ngform=2 unformatted file

ngform=1 formatted file

ngform=0 nothing written

ngform=-1 comment lines are included into file for easier reading (cannot be used for graphics processing in this form)

ngrad Output on graphics file for each ngrad'th radial point

ngcolat Output on graphics file every ngcolat'th point in colatitude

nglon Output on graphics file every nglon'th point in longitude

nfilt If>0 apply filter of type F(l)=exp[-(l/alfilt)^nfil] to the radial component of the magnetic field on the outer radius (kc=1) before writing data into graphics file (for alfilt >0). When alfilt<0 then apply filter F(l)=(1+sin(pi*(l-nfilt)/alfilt) as long as |l-nfilt|<0.5*alfilt, and F=1 and F=0 respectively for small/large l.

alfilt See under nfilt

ivfilt If >0 apply the same filter as above to the radial velocity at radial level ivfilt and write the result into graphics file at the first radial location (kc=1)

dipfilt If nfilt>0 multiply axial dipole component of B_r on outer surface by dipfilt in graphics output

nlogstep Writes data on logfile (prefix l.) after each nlogstep step

nplog If >0, writes velocity values at specific points of the grid on separate logfile (prefix "lp.") after every nplog step (for arrays, see vrpoint, vppoint, vtpoint in subroutine amhd for details)

iscale Determines which diffusivity is used for scaling of time, velocity and energy. 1=viscous, 2=thermal, 3=magnetic

enscale In output listings, energies are multiplied by enscale

treset [LOGICAL] If true, reset time and step counter to zero when starting from a stored dataset

tipdipole When starting calculation without imposed symmetry (minc=1) from a data file with symmetry (minc>1), add an equatorial dipole component with tipdipole times the magnitude of the polar dipole

amps Option for rescaling temperature perturbation (from restart file) by factor amps (if not equal to 1)

ampw Same for poloidal velocity

ampz Same for toroidal velocity

ampb Same for poloidal magnetic field

ampj Same for toroidal magnetic field

ifvfrz [LOGICAL] If true, do not update velocity during iteration

ifbfrz [LOGICAL] If true, do not update magnetic field during iteration

ifsfrz [LOGICAL] If true, do not update temperature during iteration

TIME STEP CONTROL

dtmin Minimum time step (in seconds). If the dynamically determined time step becomes less, the program terminates.

dtmax Maximum (and usually initial) time step. This must be less than 0.25*ek. Between dtmax and dtmin the actual time step is controlled by a Courant criterion (see below).

dtstart Initial time step. If dtmax=0, dtmax is used for the initial time step when init>0 and the last time step used in the previous run (stored in the restart file) is used when init=0.

courfac Controls the contribution of the fluid velocity to the Courant time step limit (a larger value leads to smaller dt)

alffac Controls the contribution of the (modified) Alfven velocity to the Courant time step limit (a larger value leads to smaller dt)

icour Check Courant criterion after each icour time step (even numbers)

PHYSICAL CONTROL PARAMETERS

ra Rayleigh number (defined with gravity on outer boundary)

ek Ekman number

pr Prandtl number

prmag Magnetic Prandtl number

radratio Ratio of inner to outer radius

bpeak Peak value of magnetic field imposed by boundary conditions at ICB (also when imagcon=0, bpeak controls the initial magnetic field: toroidal when bpeak>0, poloidal dipole when bpeak<0!)

epsc0 Volumetric rate of internal heating

BOUNDARY CONDITIONS AT INNER AND OUTER RADII

ktops thermal boundary condition at CMB. 1-fixed temp, 2-fixed radial heat flow. (ktops=2 not tested !).

kbots thermal boundary condition at ICB. As above.

ktopv velocity condition at CMB. 1-free, 2-rigid.

kbotv velocity condition at ICB. As above.

kbotb =1 for insulating inner core =2: ideally conducting inner core

ktopb =1 for insulating mantle =2: not implemented! imagcon: <0 imposed poloidal field (l=1,m=0) at ICB >=0 imposed toroidal field (l=2,m=0) at ICB >=10 imposed toroidal field (l=2,m=0) at both CMB and ICB (same amplitude and same sign if =10, opposite sign if =11)

cmb If >0, thin conducting layer at bottom of mantle (not tested!)

HYPERDIFFUSIVITIES

difamp Amplitude of hyperdiffusivities

ldif Hyperdiffusivites applied for harmonic degrees l >= ldif

ldifexp Exponent for increase of hyperdiffusivities with l (analytical details see definition of ql(lm,11) in prep.f)

PARAMETERS FOR GENERATING MOVIE FILES

imovopt Three-digit integer number, options for generating movie files

- **Last digit>0** Write B_z, W_z (vorticity) and T in the equatorial plane on file with prefix "me."
- **2nd last digit>0** Write longitudinally averaged B_phi, j_phi and v_phi on file with prefix "ma."
- **3rd last digit>0** Write B_r at outer surface and B_r and v_r at mid- depth on file with prefix "mm."
- **4th last digit>0** Write spherical harmonic coefficients for poloidal field at outer boundary and for velocity potentials at radial level on file with prefix "cc."

iframes Write altogether iframes frames on the movie files

tmovstart Time at which to start writing movie-frames

tmovstep Time increments for writing movie-frames

Appendix C

MAG Output File Format

MAG produces a set of output files for further processing. All outputs are in non-dimensional units unless specified.

l.[outfile] Lists a set of diagnostic values each nlogstep time-steps.

ls.[outfile] Spectra of kinetic energy and magnetic field every nprint time steps, sorted for modes with equal l, and additionally sorted for modes with equal m.

g.[outfile] or g[i].[outfile] where i=0,1,2,..9 (optional, written when ngstep>0). Contains temperature, velocity and magnetic field components for graphics processing (idl-program magts).

d.[outfile] or d[i].outfile Restart files with the complete set of variables (stored as spectral values `l,m` in the angular coordinates for radial grid-levels).

lp.[outfile] Written when nplog>0. Velocity at specific points written every nplog'th time step.

me.[outfile] Written when first digit of imovopt>0, i.e., imovopt=0001. Values in the equatorial plane for producing movie (idl-program: magmov EQ3.pro).

ma.[outfile] Written when second digit of imovopt>0, i.e, imovopt=0010. Longitudinal averages for producing movie (idl-program, not provided).

mm.[outfile] Written when third digit of imovopt>0, i.e., imovopt=0100. Values on spherical surfaces for producing movie (idl-program: magmovieCIG.pro, magmovCMB.pro).

cc.[outfile] Written when the last (fourth) digit and any of imovopt>0 and any of the other three digits is larger than zero, i.e. imovopt>1000. Record the spherical harmonic coefficients for poloidal field at outer boundary and their converted Gauss coefficients.

Note: If one of the above files already exists, the program will not run.

The standard output file contains summaries of grid paramaters and all process control and physical parameters that occur in the namelist statements. It lists the values of non-dimensional parameters and of the various diffusive time-scales. Then, at the end of each block, it lists a number of diagnostic values:

Parameters	Definitions
dt	actual time step
dtrmin	Courant time calculated with radial velocities
dthmin	Courant time calculated with horizontal velocities
cour	maximum inverse Courant time based on radial fluid velocity
couh	maximum inverse Courant time based on horizontal fluid velocity
alfr	maximum inverse Courant time based on radial modified Alfven velocity
alfh	maximum inverse Courant time based on horizontal modified Alfven velocity (in addition, the radial level at which the maximum is reached is indicated)
ent	total energy
env	kinetic energy
enb	magnetic energy

The meaning of other quantities is obvious.

For the primary variables, the modes for which they assume their absolute maximum and the maximum are printed. Maxima are determined for the toroidal potential multiplied by $1/r$, and for poloidal potentials multiplied by $l(l+1)/r^2$, in order to find the modes which exhibit the maximum longitudinal toroidal velocity (field strength) and the maximum radial velocity (field strength), respectively.

l.[outfile] printed every nlogstep time step, one record is printed that contains 17 output fields:

1) time
2) mean kinetic energy density
3) mean poloidal kinetic energy density
4) mean magnetic energy density
5) mean poloidal magnetic energy density
6) mean axisymmetric toroidal kinetic energy density
7) mean axisymmetric poloidal kinetic energy density
8) mean axisymmetric poloidal magnetic energy density
9) mean axisymmetric toroidal magnetic energy density
10) mean top heatflow (nusselt number)
11) mean bottom heatflow (nusselt number)
12) mean magnetic field strength
13) rms dipole, outer boundary
14) rms axial dipole, outer boundary
15) dipole tilt, outer boundary
16) dipole longitude, outer boundary
17) mean velocity

ls.[outfile] Printed each nprint time step are four records with time being the first variable followed by the spectral power of kinetic and magnetic energy, respectively, as a function of harmonic degree l, from l=0 to lmax (first two records in a block), and spectral power as a function of harmonic order m in the last two records of a block.

Appendix D

License

GNU GENERAL PUBLIC LICENSE Version 2, June 1991. Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

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.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

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

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. For example:

> One line to give the program's name and a brief idea of what it does. Copyright (C) (year) (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., 59 Temple Place, Suite 330, Boston, MA 02111-1307 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) year 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 2006
> 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.

Bibliography

[1] Olson, P., G.A. Glatzmaier (1995), Highly supercritical thermal convection in a rotating spherical shell: centrifugal vs. radial gravity. *Geophys. Astrophys. Fluid Dyn., 70*, 113-136

[2] Olson, P., G.A. Glatzmaier (1995), Magnetoconvection in a rotating spherical shell: structure of flow in the outer core. *Phys. Earth Planet Int., 92,* 109-118

[3] Olson, P., G.A. Glatzmaier (1996), Magnetoconvection and Thermal Coupling of the Earth's Core and Mantle. *Phil. Trans. R. Soc. Lond., A354,* 1413-1424

[4] Olson, P., U. Christensen, G.A. Glatzmaier (1999), Numerical Modeling of the Geodynamo: Mechanisms of Field Generation and Equilibration. *J. Geophys. Res., 104*, 10,383-10,404

[5] Christensen, U., P. Olson, G.A. Glatzmaier (1999), Numerical modelling of the geodynamo: a systematic parameter study. *Geophys. J. Int., 138*, 393-409

[6] Christensen, et al. (2001), A numerical dynamo benchmark. *Phys. Earth Planet Int., 128*, 25-34 (benchmark cases)

[7] Olson, P., G.A. Glatzmaier (2005), Probing the geodynamo. *Scientific American 15:2,* 29-35

[8] Christensen, U.R., J. Aubert (2006), Scaling properties of convection-driven dynamos in rotating spherical shells and application to planetary magnetic fields. *Geophys J. Int. 166*, 97-114.