MAGIC 3

A numerical computer program for simulating selfconsistent dynamo action in rotating spherical shells

Version 3.15, 2006/7/02

by Johannes Wicht

Table of contents

1 Running the code

The dynamo code can be downloaded from the ftp-server of the Institut for Geophysics in Göttingen:

ftp://www.mps.mpg.de/outgoing/wicht/MAGIC3

The code consists of a fairly large number of subroutines contained in the tar-file MAGIC3_??.tar.gz. Uncompressing and un-tarig the file will create a directory /MAGIC3_15/ that contains all necessary files including the make-file magic.make. Usage of this makefile for compiling and linking is strongly recommended. It has to be adapted to the specific situation, notably compiler commands and compiler options have to be changed. See the file magicCompilers for additional hints.

The command

```
make -f magic.make
```

will construct the executable magic.exe.

The code can be run by using a shell script. Examples can be found in the subdirectory /runTest/ that created by uncompressing and un-taring runTest.tar.gz. Alternatively to using scripts the input parameters can be supplied via STDIN, preferably by employing a file containing the name-lists-parameters:

```
magic.exe < namelists
```

Note that the scripts in /runTest/ as well as magic.exe will have to be made executable before running them in a unix eviroment.

Use /runTest/ for testing your code implementation. The test runs should reproduce the output-files with extensions (TAG)*Uli1P1, Uli1P2, and so forth.* Directory /runTest/ contains runs for up to 64 processors, results are supplied for up to eight processors. At the end of each log.TAG file run-times can be found that allow you to jugde speed and scaling of you computer. The listed results have been computed with AMD Opteron 875 2.2 Ghz processors. Note that the results may differ in the last two digits or so when run on different systems.

The number of input parameters has grown to more than 100. But dont be confused by this number. All parameters are set internally to a (hopefully) useful default value. You may start with just changing physical parameters (Rayleigh number, Ekman number,...) to the desired values and then checking the truncation. Use the test scripts as a guideline for preparing other input files. Any log-file contains a list of all input parameters in the required format, which can be used as a basis for a more general input file. But see the explanation on input in chapter 2.

Unfortunately, this manual is quite outdated by now. I didn’t find the time for an update for some years now. The following chapters can nevertheless give you an idea of the most important input parameters. Newer additions mostly concern new output options and new model analysis.

During a run the code can be controlled by two soft interrupts:

```
    kill -30 processID
```

will cause the code to finish the current time step, write a restart file (and a graphic file if demanded) and then stop. The run can be continued later by using the last restart file produced by the run.
Interrupt:

```
    kill -40 processID
```

causes the code to write a full graphic output for the current time step.

2 Input

2.1 Grid size and truncation

truncation.f Parameters concerning the numerical grid size are defined in a file called truncation.f.. This file is included into the code at compilation. Thus, re-compilation is necessary for any changes in the grid.
In radial direction the code uses a representation in Chebychev polynomials when radial derivatives or integrals over radius have to be calculated. The number of radial grid points and the degree of the Chebychev representation can be chosen independently for the inner and outer core.
In longitudinal and latitudinal direction the code switches back and forth between the physical grid and a representation in spherical harmonic functions. The size of the longitude-latitude grid is determined by the number of grid point in longitudinal direction *n_phi_tot*. The latitude-grid uses exactly *n_phi_tot/2* points.
Degree $l_{\max}$ and order $m_{\max}$ of the spherical harmonic expansion are determined by the dealiazing parameter *nalias:*

$$l_{\max} = nalias/30 \quad n\_theta\_max$$

The representation is alias-free for *nalias*=20. Generally the maximum degree and order are identical, in addition $m_{\max}$ has to be a multiple of *minc*:

$$m_{\max} = n\ minc \le l_{\max}$$

with *n=1,2,3...* The parameter *minc* defines the symmetry in longitude direction. For a *minc*-fold symmetry only orders that are a multiple of *minc* contribute to the representation in spherical harmonic functions.

n_r_max	Integer, number of grid points in outer core $[r_i, r_o]$. *n_r_max-1* must be a multiple of 4.
n_cheb_max	Integer, no. of Chebychev-polynomials in OC, maximum Chebychev degree used is *n_cheb_max-1*. $n\_cheb\_max \le n\_r\_max$

n_phi_tot	Integer, no. of longitudinal grid points. *n_phi_tot* must be a multiple of *minc*. *n_phi_tot/minc* must be a multiple of 4. *n_phi_tot* must be a multiple of 16.
n_r_ic_max	Integer, no. of radial grid points in IC $[0, r_i]$. *n_r_ic_max-1* must be a multiple of *4*.
n_cheb_ic_max	Integer, no. Chebychev-polynomials in IC, only even Chebs used in inner core with maximum degree $2\ n\_cheb\_ic\_max - 2$ $n\_cheb\_ic\_max \leq n\_r\_ic\_max$
minc	Integer, basic wavenumber for the longitudinal symmetry, orders *m* used in the spherical harmonic expansion are a multiple of *minc*.
nalias	Integer, parameter determining antialiasing used in the spherical harmonic representation.

blocking.f This is a second file that is included into the code at compilation. Only two free parameters are defined here: the size of a block *nChunk* and the number of threads *nThreads* for a parallel run. Blocking has been introduced to speed up the code. The optimum size of blocking depends on the specific arcitecture of a computer, especially on the cach-size. .
(More about this in the future.)

2.2 Input parameters

True run-time input parameters are read from STDIN as name-lists, a Fortran feature. A name-list is identified by its unique name *&name*. The name-statement is followed by the parameters that are part of the namelist in the format *parameter=value,*. The namelist is closed by a backslash. The subroutine s_default_namelist.f defines a default value for each parameter. Only the parameters whose value should differ from its default have to be stated in the namelist.
An example for the short namelist defining inner core parameters is:

```
&inner_core
  sigma_ratio = 1.0,
  nRotIc      = 1/
```

Comas can be used to seperate namelist entries since they are not interpre-

ted by the code.
Magic uses six namelists:

1. **&control** for control parameters and numerical parameters.
2. **&phys_param** for the physical parameters.
3. **&start_field** to define the starting fields.
4. **&output_control** for defining the output.
5. **&mantle** for setting mantle parameters.
6. **&inner_core** for setting inner core parameters.

&control namelist

This namelist defines the numerical parameters of the problem plus the variables that control and organize the run. An exeption are the grid-size and truncation parameters, that have to be set before compiling the code.

mode	Integer, controls the type of calculation performed
mode=0	Selfconsistent dynamo
mode=1	Convection
mode=2	Kinematic dynamo
mode=3	Magnetic decay modes

tag Character string, used as extension for all output files.

n_time_steps Integer, no. of time steps to be performed.

iscale	Integer, determins time scaling.
iscale=0	Use thermal time scale.
iscale=1	Use viscouse time scale.
iscale=2	Use magnetic time scale.

alpha Real, weight used for current time step in implicit time step.

enscale Real, scaling for energy output.

l_update_v=.true. Logical, time step velocity field.

l_update_b=.true. Logical, time step magnetic field.

l_update_s=.true. Logical, time step entropy.

Time step control

A modified courant criteria including a modified Alfen-velocity is used to account for the magnetic field. The relative and absolute importance of flow and Alfen-velocity can be controled by *courfac* and *alffac* respectively.

dtstart	Real, used as initial time step if start solution is initialized (see below) and $dtstart > 0$.
dtmax	Real, maximum allowed time step δt. If $\delta t > dtmax$ time step is decreased to at least *dtmax* (See routine s_dt_courant.f). Run is stopped if $\delta t < dtmin$, $dtmin = 10^{-4} dtmax$ in present version. This could be changed in s_input.f.
courfac	Real, used to scale velocity in courant criteria.
alffac	Real, used to scale Alfen-velocity in courant criteria.
n_cour_step	Integer, no. of time steps before consecutive checking of courant criteria. Note: the courant criteria is checked always after the time step has been changed if $n\_cour\_step > 0$.

Hyperdiffusivity

Hyperdiffusion is applied by multiplying a factor of the form

$$d(l) = 1 + D\left[\frac{(l + 1 - l_{hd})}{(l_{max} + 1 - l_{hd})}\right]^{\beta}$$

to diffusion terms for degrees $l \geq l_{hd}$.

difnu	Real, amplitude *D* of viscous and thermal hyperdiffusion.
difeta	Real, amplitude *D* of magnetic hyperdiffusion
ldif	Integer, degree l_{hd} where hyperdiffusion starts to act.
ldifexp	Integer, exponent β of hyperdiffusion.

&start_field namelist Controls whether a start field from a previous solution should be used, or a specific field should be initialized

Input of start fields:

l_start_file	Logical to tell the code that it should use a startfile.
l_start_field=.true.	Read start fields from file *start_file*.
l_start_field=.false.	Construct initial field.

start_file	Character string, name of startfile.

inform	Integer, gives format of fields in *start_file*.
inform=0	Oldest format used by Christensen.
inform=1	Newer format used by Christensen.
inform=2	Inner core information added by JW.
inform=3	Lorentz-torques added by JW.

l_reset_t=.true. Logical, time of start file is reset to zero.

scale_s	Real, input entropy field from *start_file* is scaled with *scale_s*.
scale_v	Real, input velocity field from *start_file* is scaled with *scale_v*.
scale_b	Real, input magnetic field from *start_file* is scaled with *scale_b*.

tipdipole Real, used to add non-axisymmetric disturbances to a startsolution if non-axisymmetric parts have been lost due to mapping to a different symmetry.
A (l=1,m=1) entropy term is added with:

$$S_{10}(r) = \textit{tipdipole}\ \sin(\pi(r - r_i))$$

If a magnetic field without an *m=1* term is mapped into a field that permits this term, the code adds the respective poloidal field using the (l=1,m=0) poloidal magnetic field and scaling it with *tipdipole*.

Initialization of Entropy:

The heat equation with heat sources give by epsc0 is solved for laterally symmetric term (l=0,m=0) to get its radial dependence. Two other laterally variing terms can be initialized. Their radial dependence is taken to be:

$$s(r) \sim 1 - 3x^2 + 3x^4 - x^6$$

with

$$x = 2r - r_o - r_i$$

Thus the field is zero at r_i and r_o and has a maximum amplitude of amp_s1 or amp_s2 at midradius $r_i + 1/2$.

init_s1	Integer, controls first harmonic to be used as an initial entropy
$init\_s1 = 0$	Nothing initialized
$init\_s1 < 100$	Random noise with amplitude amp_s1 . See s_init_s.f for details.
$init\_s1 > = 100$	Initialisation of mode with order given by the last two digits of $init\_s1$ and the degree given by the first one or two digits.

init_s2	Integer, can be used to initialize second harmonic. See $init\_s1$ above.

amp_s1 ***amp_s2***	Real, amplitudes used for entropy initialization. See above.

Initialization of magnetic field:

init_b1	Integer, controls initialization of magnetic field.
$init\_b1 = 1$	Diffusive toroidal field initialized. Mode determined by $imagcon$.
$init\_b1 = 2$	(l=1,m=0) toroidal field with a maximum field strength of amp_b1. Radial dependence is chosen, so that the field vanishes at inner and outer.

Insulating inner core:

$$t(r) \sim r\sin(\pi(r - r_o))$$

Conducting inner core:

$$t(r) \sim r\sin(\pi(r/r_o))$$

init_b1=3	(l=1,m=0) poloidal field whose field strength is *amp_b1* at $r = r_i$. The radial dependence is chosen so that the current density j is independend of r: $$\partial_r \vec{j} = 0$$ (l=2,m=0) toroidal field with maximum strength amp_b1. Radial dependence as for *init_b1=2.*
init_b1=4	(l=1,m=0) poloidal field as if the core where an insulator (potential field). Field strength at $r = r_i$ is again given by *amp_b1*.
init_b1=5	(l=1,m=0) poloidal field with field strength *amp_b1* *at* $r = r_o$. Radial dependence again defined by $\partial_r \vec{j} = 0$.
init_b1=6	(l=1,m=0) poloidal field independend of *r*.
init_b1=7	(l=1,m=0) poloidal field which fulfills symmetry condition in IC: $$s(r) \sim (r/r_i)^2(1 - 3/5(r/r_o)^2)$$ Field strength at $r = r_o$ given by *amp_b1*.
init_b1=8	Same poloidal field as for i*nit_b1=8*. Toroidal field fulfills symmetry conditions in inner core and has a field strength of amp_b1 at $r = r_i$: $$t(r) \sim (r/r_i)^3(1 - (r/r_o)^2)$$
imagcon	Determins imposed magnetic field for magnetoconvection. Magnetic field is imposed at boundaries.
imagcon=0	No magnetoconvection
imagcon<0	Axial poloidal dipol imposed at ICB with maximum magnetic field strength *amp_b1*.
imagcon=1	(l=2,m=0) toroidal field imposed at ICB with maximum field amplitude *amp_b1* and zero amplitude at CMB.
imagcon=10	(l=2,m=0) toroidal field imposed at ICB and CMB with maximum amplitude *amp_b1* at both boundaries.
imagcon=11	Same as *imagcon*=10 but the maximum amplitude is now *amp_b1* at the ICB and *-amp_b1* at the CMB.

imagcon = 12	(l=1,m=0) toroidal field with a maximum amplitude of *amp_b1* at the ICB and the CMB.
amp_b1	Real amplitude for magnetic field initialization

Initialization of velocity field:

init_v1	Integer to control velocity field initialization. Not used so far.
amp_v1	Real amplitude for velocity field initialization.

&phys _param namelist

Dimensionless Parameters:

ra	Real, Rayleigh number
ek	Real, Ekman number
pr	Real, Prandtl number
prmag	Real, magnetic Prandtl number
epsc0	Real, dimensionless internal heating rate desity
radratio	Real, inner core radius/outer core radius

Boundary conditions:

ktops	Integer, outer heat boundary condition
ktops = 1	Fixed entropy at outer boundary.
ktops = 2	Fixed heat flux at outer boundary
kbots	Integer, inner heat boundary condition
kbots = 1	Fixed entropy at inner boundary
kbots = 2	Fixed heat flux at inner boundary
ktopv	Integer, outer velocity boundary condition
ktopv = 1	Stress free outer boundary
ktopv = 2	No-slip outer boundary

kbotv	Integer, inner velocity boundary condition
kbotv=1	Stress free inner boundary
kbotv=2	No-slip inner boundary

ktopb	Integer, outer magnetic boundary condition
ktopb=1	Insulating outer boundary
ktopb=2	Perfectly conducting outer boundary
ktopb=3	Finitely conducting outer boundary

kbotb	Integer, inner magetic boundary condition
kbotb=1	Insulating inner boundary
kbotb=2	Perfectly conducting inner boundary
kbotb=3	Finitely conducting inner boundary

s_top Real array of laterally variing outer heat boundary conditions. Each four consecutivy numbers are interpreted as follows:
1) Spherical harmonic degree
2) Spherical harmonic order
3) Real amplitude (cos contribution)
4) Imaginary amplitude (sin contribution)
For example, if the boundary condition should be a combination of an (l=1,m=0) sherical harmonic with the amplitude 1 and an (l=2,m=1) spherical harmonic with the amplitude (0.5,0.5) the respective namelist entry could read:

```
 s_top  =  1,  0,  1.0,  0.0,
               2,  1,  0.5,  0.5,
```

The comas could be left away.

s_bot Same as *s_top* but for the bottom bounday.

&output_ control namelist Parameters that control the output. For explanation of output files and their contents see below.

Determining times for output:

There are four different ways to control at which time step a specific output should be written. Outputs are generally distributed over the total calculation intervall unless an output time intervall is defined by a start time *t_start* and a stop time *t_stop.* If no *t_start* is provided, the start time of the calculation is used. If no *t_stop* is provided or t_*stop* > *t_start* the total calculation intervall is assumed

1. *Prescribed number of outputs.* The outputs are distributed evenly over the total calculation intervall so that the number of timesteps between two outputs is always the same, with the possible exception of the first intervall. Last output is written for the last time step, and to compensate the intervall before the first output may be longer. However, if *t_stop* is provided, the outputs are distributed evenly over the intervall [*t_stop,t_start*] with equal times intervalls between them.
2. *Defined intervall between two outputs, given in number of time steps.* Again the last output is performed at the end of the run and a compensation may take place at the beginning.
3. *Defined time intervall between two outputs.*
4. *Defined times for output.* By default 2000 different times can be defined for each output type. This can be increased by increasing *n_time_hits* in the file c_output.f.

While the first three possibilities can only be used alternatively, the fourth one can be employed in addition to one of the two others.
An important new parameter in this context is *l_true_time.* If this is set to true, the time steps of the program are modified to meet a desired output time. This forces a recalculation of the inversion matricies and therefore requires some computing time. For *l_true_time=.false.* the values at the timestep closest to the desired output time are chosen. Since the timesteps are generally small, this option suffices for most applications.

l_true_time=.true.	Logical, causes code to change time steps to meet output times.

n_graph_step	Integer, intervall in no. of time steps for output of graphic files.
n_graphs	Integer, no. of graph files.
t_graph	Real array of times for output. 99 times allowed by default.
dt_graph	Real, time intervall between outputs.
t_graph_start	Real, time to start output.
t_graph_stop	Real, time to stop output.

n_rst_step	Integer, intervall for storage of restart files.
n_rsts	Integer, no. of restart files.
t_rst	Real array of times for restart files.
dt_rst	Real, time intervall between outputs.
t_rst_start	Real, time to start output.
t_rst_stop	Real, time to stop output.
n_stores	Integer, no. of storages of restart information. If *n_stores* > *n_rsts* a restart files is overwritten.

n_log_step	Integer, intervall for writing energies, dipole information (dipole-file) and rotation information (rot-file).
n_logs	Integer, no. of log-information sets written.
t_log	Real array, times for log-information.
dt_log	Real, time intervall between outputs.
t_log_start	Real, time to start output.
t_log_stop	Real, time to stop output.

n_p_step	Integer, intervall for point-information (see below).
n_ps	Integer, no. of point-informations written.
t_p	Real array, times for point information.
dt_p	Real, time intervall between point-info.
t_p_start	Real, time to start output.
t_p_stop	Real, time to stop output.

n_spec_step	Integer, intervall for output of spectra.
n_specs	Integer, no. of spectra files.
t_spec	Real array, times for spectra.
dt_spec	Real, time intervall between outputs.
t_spec_start	Real, time to start output.
t_spec_stop	Real, time to stop output.
n_diag_step	Integer, intervall for diagnosis output.
n_diags	Integer, no. of diagnosis outputs.
t_diag	Real array, times for diagnosis.
dt_diag	Real, time intervall between outputs.
t_diag_start	Real, time to start output.
t_diag_stop	Real, time to stop output.
n_cmb_step	Integer, intervall for CMB field output.
n_cmbs	Integer, no. of CMB field outputs.
t_cmb	Real array, times for CMB field output.
dt_cmb	Real, time intervall between CMB field outputs.
t_cmb_start	Real, time to start CMB field output.
t_cmb_stop	Real, time to stop CMB field output.
n_movie_step	Integer, time steps between movie frames.
n_movie_frames	Integer, total no. of movie frames.
t_movie	Real array, times for movie frames.
dt_movie	Real, time intervall between movie frames.
t_movie_start	Real, time for first movie frame.
t_movie_stop	Real, time for last movie frame.
ngform	Integer, format for graphic files.

ngform = -1	Formatted output with comments.
ngform = 0	Unformatted output.
ngform = 1	Formatted output, no comments.

The output grid for the graphic output can be chosen differently from the grid the calculations are performed on, but the former is always a subgrid of the latter.

ns_r_gra	Integer, output of each *ns_r_gra* radial grid point from the numerical grid in graphic file.
ns_theta_gra	Integer, output of each *ns_theta_gra* colatitude grid point in graphic file.
ns_phi_gra	Integer, output of each *ns_phi_gra* longitude grid point ...
ns_r_ic_gra	Integer, output of each *ns_r_ic_gra* radial grid point in the IC ...

runid	Character string printed in graphic output files. Can be printed by plotting routines.

Filtering of CMB Field:

To compare the results with the CMB field of geomagnetic inversions, the CMB field of the graphics output can be filtered. Two filters *w(l)* are offered, where *w* is the weight and *l* the harmonic degree:

1) For $alfilt > 0$ Super-Gaussian filter:

$$W(l) = e^{-(l/alfilt)^{nfilt}}$$

2) For $alfilt < 0$ Cos-taper filter:

$$W(l) = \{-\sin[\pi((l - nfilt)/|alfilt|)]\}/2$$

nfilt	Integer, see above for definition.
ivfilt	Integer, no. of radial level where the filtering is performed. Use $ivfilt = 1$ for CMB. No filtering is performed for $ivfilt < 1$.
alfilt	Real, see above for definition
$alfilt > 0$	Use super-Gaussian filter
$alfilt < 0$	Use cos-taper filter.
dipfilt	Real, additional factor applied to dipole (l=1,m=0).

Movies:

Parameters that determine the number and times of movie frames have already been explained above. Movie-files are only written if

l_movie=.true.	Logical, envokes production of movie files.

Generally several movie-files can be produced during a run [limited to 10 by *n_movies_max* in c_output.f]. The movies are defined by a keyword detemining the fields to be plotted and an expression that determins the movie surface. The code searches this information in a character string provided for each movie. These strings are elements of the array ***movie.***. For example, to invoke a movie(file) that shows (stores) the radial magnetic field at the CMB, you have to provide the line

```
   movie(1)= Br CMB
```

in the output namelist. Here, Br is the keyword for the radial magnetic field and CMB the expression that defines the movie surface. If, in addition, a movie of the temperature field at the meridional slice *phi=0* is desired, the following line has to be added:

```
   movie(2)= T phi=0
```

Note that the code does not interpret blanks and ignores additional characters that do not form a keyword or a surface definition. Thus, for example *Br* or *B r* or *Bradial* are all interpreted as the same keyword. Furthermore, the interpretation is not case-sensitive. Another example: the keywords *Toroidal axisymmetric field* and *toraxi* cause the same movie file to be written. Note that the keywords *fieldlines* and *toraxi* do not need an expression defining the surface.

Keyword	**Fields stored in movie file**
Br	Radial magnetic field
Btheta	Theta component of the magnetic field
Bphi	Azimuthal magnetic field
Bh	Horizontal magnetic field, two components depending on the surface.
Bo	Orthogonal magnetic field, one component depending on the surface.
Ba	All magnetic field components
Fieldline	Axisymmetric poloidal field lines in a meridional cut.
tor axi	Axisymmetric azimuthal toroidal field in a meridional cut.

Keyword	Fields stored in movie file
`Vr`	Radial velocity field
`Vtheta`	Theta component of the velocity field
`Vphi`	Azimuthal velocity field
`Vh`	Horizontal velocity field, two components depending on the surface.
`Vo`	Orthogonal velocity field, one component depending on the surface.
`Va`	All velocity field components
`T or S`	Temperature field

Surface expression	**Surface**
CMB	Core-mantle boundary
Surface	Earth surface
r=*radius*	Radial surface at r=`radius` with radius given in units of the outer core radius.
theta=*colatitude*	Cut at constant colatitude given in degree
phi=*phi*	Slice at constant longitude given in degree.

Similar to the graphic output grid, the movie grid is a subgrid of the numverical grid. A different grid can be chosen for movies at the Earth surface where the magnetic field in of larger scale than in the core.

ns_r_mov	Integer, output of each `ns_r_mov` radial grid point from the numerical grid in movie file.
ns_theta_mov	Integer, output of each `ns_theta_mov` colatitude grid point in movie file.
ns_phi_mov	Integer, output of each `ns_phi_mov` longitude grid point ...
ns_r_ic_mov	Integer, output of each `ns_r_ic_mov` radial grid point in the IC ...
ns_theta_sur	Integer, output of each `ns_theta_mov` colatitude grid point in movie file at Earth surface.
ns_phi_sur	Integer, output of each `ns_phi_mov` longitude grid point in movie file for Earth surface.

Magnetic field at the CMB:

The poloidal field coeffitients are stored in a special field at the times defined by the parameters mentionend above if:

l_cmb=.true.	Logical, envokes output of field at CMB.

Field Averages:

The code can performe on-the-fly time averaging of entropy, velocity field and magnetic field. Respective graphics output and spectra are written into the files numbered 0 (see below). Averaged energies are written into the log-file.

l_average=.true.	Logical, envokes averaging of fields.

Secure Output:

Secure output can be chosen be setting

l_save_out=.true.	Logical, envokes secure output.

The program closes each output file after a write command, and reopens it if necessary. This may cost some computing time, but guarantees that only minimal information is lost in case of a crash.

&mantle namelist

Defines mantle properties:.

conductance_ma	Real, conductance of a thin layer at the CMB, dimensionless.
nRotMa	Integer determining mantle rotation
nRotMa=1	Rotating mantle according to torques
nRotMa=0	Fixed, non-rotating mantle
nRotMa=-1	Mantle rotates with prescribed rate omega_ma1
omega_ma1	Real, a prescribed initial mantle rotation rate.
rho_ratio_ma	Real, (mean) mantle density in units of outer core density.

&inner_core namelist

Defines inner core properties:.

sigma_ic	Real, ratio of inner core and outer core conductivities.
nRotIc	Integer, determines IC rotation, see nRotMa

omega_ic1	Real, a prescribed inner core rotation rate
rho_ratio_ic	Real, (mean) inner core density in units of outer core density.

3 Output

While some information of a run is written into *STDOUT* to monitor its progress, most output is printed into files identified by the extension *.tag*. Some information found in *STDOUT* is also written to the log-file called log*.tag*. In addition, this file contains all input parameters, truncation, information on other output files, and some results like the time averaged energies (for *l_average=.true.*). Routines that process the results extract information concerning the run from log*.tag* and keywords are used in to identify the different contents. Other output files, are organised in columns or lines (datasets). Thier meaning is explained below.
The number of graphic files (G#*.tag* or g#*.tag*), restart files (rst#*.tag*), and spectrum files (spec#*.tag*) are determined by the input parameters in the namelist &output.contrl (see above). A new file is produced for each output time. If time averaging has been chosen the time averaged graphic and spectrum files will have the no. 0. All other files are numbered consecutively starting with 1. Times at which these files have been written can be found in log*.tag*.
Other files (except the log-file) contain time sequences. The times for the sequences in the energy files, the rot-file, the dipole-file, and the misc-file are determined by the log-times (see above). Output times for cmb-file and movie frames are chosen independently (see above).
Restart, movie and cmb-file (B_coeff_cmb*.tag*) are unformatted files (organised in datasets, called lines here).

log.*tag* Contents in order of appearence:

1	Program version
2	Mode (dynamo,convection,...)
3	Grid parameter
4	Name-lists
5	Start-field and start time
6	Information on time stepping progress in step of 10%
7	Name and time of graphic files.
8	Name and time of restart files.
9	Time averaged energies for *l_average=.true*.
10	Energies at end of run.

11	End time.

e_mag_ic.*tag* Inner core magnetic energy:

No. of column	Contents
1	time
2	IC poloidal energy
3	IC toroidal energy
4	IC axisymmetric poloidal energy
5	IC axisymmetric toroidal energy

e_mag_oc.*tag* Outer core magnetic energy and energy ouside the core.

No. of column	Contents
1	time
2	OC poloidal energy
3	OC toroidal energy
4	OC axisymmetric poloidal energy
5	OC axisymmetric toroidal energy
6	external (poloidal) energy
7	external axisymmetric (poloidal) energy

e_kin.*tag* Outer core kinetic energy.

No. of column	Contents
1	time
2	poloidal energy
3	toroidal energy
4	axisymmetric poloidal energy
5	axisymmetric toroidal energy

rot.*tag* Output concerning the rotation of inner core and mantle. This file is written by `s_write_rot.f`.

No. of column	Contents
1	time
2	IC rotation rate
3	Lorentz torque on IC
4	viscouse torque on IC
5	mantle rotation rate
6	Lorentz torque on mantle
7	viscous torque on mantle
8	poloidal magnetic field component (l=1,m=*minc*), real part
9	poloidal magnetic filed component (l=1,m=*minc*), imag. part
10	total angular momentum
11	IC angular momentum
12	OC angular momentum
13	mantle angular momentum

misc.*tag* This file constains time sequences of the Nusselt number at the inner and outer boundary. The name has been chose because more output might be written into this file in later version..

No. of column	Contents
1	time
2	Nusselt number at the inner-core boundary
3	Nusselt number at the core-mantle boundary

mag_spec#.*tag* Magnetic energy spectra.

No. of column	Contents
1	degree / order
2	OC poloidal energy per degree

No. of column	Contents
3	OC poloidal energy per order
4	OC toroidal energy per degree
5	OC toroidal energy per oder
6	IC poloidal energy per degree
7	IC poloidal energy per order
8	IC toroidal energy per degree
9	IC toroidal energy per order
10	CMB (poloidal) energy per degree
11	CMB (poloidal) energy per order

kin_spec#.*tag* Kinetic energy spectra.

No. of column	Contents
1	degree / order
2	OC poloidal energy per degree
3	OC poloidal energy per order
4	OC toroidal energy per degree
5	OC toroidal energy per oder
6	CMB energy per degree
7	CMB energy per order

rst#.*tag* Unformatted file that contains all fields and other information to restart or continue a run. Written by subroutine `s_store.f`.

G#.*tag*/ g#.*tag* Graphic output files. Names of unformatted (formatted) files start with capital G (little g). Whether the output is formatted or unfomated output is controlled by *`ngform`*.
The files start with a header containing first a string informing about the graphical output version --- the most recents are: Graphout_Version_4 for formatted output and Graphout_Version_5 for unformatted output --- followed by the *`runid`*, plus information about the truncation and dimensionless parameters.

The fields are written in batches for a radial level and a colatitude block. If the code runs on a prallel computer, these batches may not be in the correct order, but can easily put into order with the information in line `n`. Here *n* numbers the output batches. The mangetic field is only written for a dynamo run.

Position	Contents
line 1	String, version information
line 2	`runid`
line 3	time, n_r_max, n_theta_max, n_phi_tot, ngrad, ngcolat, nglon, n_r_ic_max-1, ngrad_ic, minc, nblock, ra, ek, pr, prmag, radratio, sigma_ratio
line 4	list of colatitudes in graphic output
line n	No. of radial level in graphic output radius normalised to r_o, no. of start colatitude in block, no. of last colatitude in block
line n+1	entropy for all longitudes and all latitudes in the latitude block
line n+2	radial velocity for all longitudes and all latitudes in the latitude block
line n+3	latitudinal velocity for all longitudes and all latitudes in the latitude block
line n+4	zonal velocity for all longitudes and all latitudes in the latitude block
line n+5	radial mangetic field for all longitudes and all latitudes in the latitude block
line n+6	latitudinal magnetic field for all longitudes and all latitudes in the latitude block
line n+7	zonal magnetic field for all longitudes and all latitudes in the latitude block

dipole.*tag* Information about the magnetic dipole.

No. of column	Contents
1	time
2	colatitude of dipole-pole
3	longitude of dipole-pole
4	relative energy of axisymmetric dipole CMB
5	energy of axisymmetric dipole CMB normalised with total energy up to degree and order 11
6	relative energy of dipole in outer core
7	relative energy of dipole at CMB
8	energy of dipole at CMB normalised with total energy up to degree and order 11

B_coeff_cmb.*tag* Contains time sequence of poloidal magnetic field coeffitients at the CMB up to degree and order 8. The file is unformatted and follows a special order to save space and time. It is written by s_write_b_cmb.f.
Please use s_read_b_cmb.f to extract the time sequences.

?_mov.*tag* The question mark stands for the name that characterizes the movie.
This name contains information about the fields and the surface (see above). First comes the field information: Br_, Bt_, Bp, Bh_, B_ (total field), Vr_, Vt_, Vp_, Vh_, V_, T_, FL_ (fieldlines), or TAS (toroidal axisymmetric field). Information about the surface is second: CMB, SUR, R=C, T=C, P=C. No information on the surface is needed or the axisymetric fieldlines or the axisymmetric toroidal field.
Each movie file consists of a head with information on the movie type, grid, and parameters.
(See s_write_movie_frame.f and s_get_movie_type.f for details.)
The head is followed by the frames, each containing of a line with frame number and time plus one line for each field.
I am using an IDL script to extract and display the frames. Please contact me, if you plan to write your own graphic routine and need additional information on the structure of the movie-files.

4 Subroutines

Prefixes are used to identify different types of routines. Generally main programs have no prefix while subroutines have the prefix s_ while function strart with f_ . In addition all files containing commen blocks to be included in the routines are marked with the prefix c_ .
Subroutines from two JW-libraries are used:
l_char_manip.f and l_usefull.f.

4.1 Main code

magic.f

Calls:
s_prep.f, s_write_namelists.f, s_get_e_kin.f, s_get_e_mag.f, s_write_rot, s_step_time.f

Task:
Main program.

f_random.f

Called by:
s_init_s.f

Task:
Random generator for initialisation of entropy field.

s_cgesl.f

Called by:
s_update_b.f, s_update_s.f, s_update_wp.f, s_update_z.f

Task:
Backsubstitution of LU-decomposed matricies for linear equation system for one RHS.

s_cgesl2.f

Called by:
s_update_b.f, s_update_s.f, s_update_wp.f, s_update_z.f

Task:
Same as s_cgesl.f but for two RHSs.

s_cheb_x _map_e.f

Called by:
s_rad_func.f

Task:
Returns the radial grid points in the actual intervall $[r_i, r_o]$ and the associated points in the Chebychev intervall $-(1, 1)$.

s_check_time _hits.f

Called by:
s_step_time.f

Task:
Checks whether a desired time t_a (f.e. for a graphic output) falls into the intervall $[t, t+\delta t]$, where t is the current stepping time, and δt is the time step. If this is true δt is changed so that the next time step will hit t_a exactly.

s_copy_data.f

Called by:
s_map_data.f

Calls:
s_init_costf1.f, s_costf1.f

Task:
Copies data from a strart field with a different into the desired grid.

s_copy_data _ic.f

Called by:
s_map_data_ic.f

Calls:
s_init_costf1.f, s_costf1.f

Task:
Same as s_copy_data.f but for inner core.

s_costf1.f

Called by:
s_copy_data.f, s_copy_data_ic.f, s_get_dr.f, s_get_ddr.f, s_get_dddr.f, s_get_ddr_even.f, s_get_e_kin.f, s_get_e_mag.f, s_j_cond.f, s_s_cond.f, s_spectrum.f, s_update_b.f, s_update_s.f, s_update_wp.f, s_update_z.f, s_write_rot.f

Calls:
check_dim.f (in l_usefull.f), s_fft_fac.f

Task:
Performs a multiple cosine transform of the first kind (see f.e. numerical recipes).

s_costf2.f

Called by:
s_get_ddr_even.f, s_update_b.f

Calls:
check_dim.f (in l_usefull.f), s_fft_fac.f

Task:
Performs a multiple cosine transform of the second kind (see f.e. numerical recipes).

s_courant.f

Called by:
s_step_time.f

Task:
Calculates advection length and length associated with the modified Alfen-velocity necessary to check the Courant condition (see s_dt_courant.f).

s_default _namelists.f

Called by:
s_input.f

Task:
Sets default namelists to default values.

s_dt_ courant.f

Called by:
s_step_time.f

Task:
Checks whether time step should be changed because of voilation of the courant criteria. Time step is modified accordigly, checking whethe it is in the allowed intervall of $[dtmin, dtmax]$ (see input parameters).

s_factorise.f

Called by:
s_init_costf1.f, s_init_costf2.f

Task:
Factorises the number of radial grid points for the cos-transform.

s_fft_fac.f

Called by:
s_costf1.f, s_costf2.f

Task:
Performs a complex Fourier transform.

s_fft_init _own.f

Called by:
s_theta_phi_func.f

Task:
Initialises the Fourier transform used to go back and forth from *m*-space (spherical order) to ϕ-space (longitude).

s_fields _average.f

Called by:
s_step_time.f

Calls:
s_spectrum.f, s_graph_out.f, s_graph_out_ic.f, s_vb_comp_trans.f

Task:
Performs on the fly averaging of velocity, magnetic and entroy field in *(r,l,m)*-space if $l\_averag = .true.$ After the last time step or after a kill -30 command the averaged fields in physical space (r, θ, ϕ) are caculated and writte into a graphic output file. In addition the averaged spectra and

energies are calculated.

s_filter.f

Called by:
s_graph_out.f

Task:
Filters the magnetic field at the CMB according the the filter chosen by the respective input parameters to mimic the filtering of the magnetic field by the Earth´s mantle. (see above, namelist *&output_contrl*)

s_fourtf_ic_own.f

Called by:
s_graph_out_ic.f

Calls:
fft99a.f, fft99b.f, wpass2.f, wpass3.f, wpass4.f, wpass5.f

Task:
Performs several simulatious FFTs for the back and forth transform from *m*-space (spherical order) to ϕ-space (longitude).

s_fourtf_own.f

Called by:
s_graph_out.f

Calls:
fft99a.f, fft99b.f, wpass2.f, wpass3.f, wpass4.f, wpass5.f

Contains:
ft99a.f, fft99b.f, wpass2.f, wpass3.f, wpass4.f, wpass5.f

Task:
Same as s_fourtf_ic_own.f, differs only in size of internal work-array.

s_gauleg.f

Called by:
s_theta_phi_func.f

Task:
Returns colatitudes and weights for the Gauss-Legendre transformation in colatitude direction.

s_get_b_ic.f

Called by:
s_graph_out_ic.f, s_store_movie_frame_ic.f

Calls:
s_fourtf_ic.f

Task:
Calculates the magnetic field components in the inner core.

s_get_b _surface.f

Called by:
s_store_movie_frame.f

Calls:
s_fourtf.f

Task:
Calculates the magnetic field components at the Earth´s surface.

s_get_bmat.f

Called by:
s_step_time.f

Calls:
s_sgefa.f

Task:
Calculates the implicit time step matrix for the poloidal and toroidal magnetic field potentials and returns their LU-decomposition.

s_get_br_v _cmb.f

Called by:
s_step_time.f

Calls:
s_fourtf.f, s_legtf_2.f

Task:
Calculates terms for the nonlinear magnetic boundary condition in the approximation for a small mantle conductance. The actual BC is finished by s_get_mag_nl_cmb.f (see below).

s_get_chebs.f

Called by:
s_rad_func.f

Task:
Calculates Chebychev polynomials and their first, second and third derivative at the points returned by s_cheb_x_map_e.f.

s_get_chebs_even.f

Called by:
s_step_time.f

Task:
Same as s_get_chebs.f but calculates only the even polynomials.

s_get_dcheb.f
s_get_ddcheb.f
s_get_dddcheb.f
s_get_ddcheb_even.f

Called by:
s_get_dr.f, s_get_ddr.f, s_get_dddr.f, s_get_ddr_even.f, s_update_b.f, s_update_s.f, s_update_wp.f, s_update_z.f

Task:
Routines return the Chebychev-coeffitients of the radial derivative of a function given itself in Chebychev-coeffitients. The maximum degree of the derivative calculated is given by the number of ds infront of the cheb in the subroutine name. Again s_get_ddcheb_even.f only calculates the derivatives of an even function, but note that the first derivatives are odd while the second derivatives are again even.

s_get_dr.f
s_get_ddr.f
s_get_dddr.f
s_get_ddr_even.f

Called by:
s_get_dr_dt.f, s_init_b.f, s_init_s.f, s_init_v.f, s_read_start_file.f

Calls:
s_costf1.f, s_costf2.f, s_get_dcheb.f, s_get_ddcheb.f, s_get_dddcheb.f, s_get_ddcheb_even.f

Task:
Similar to s_get_dcheb.f .... but here the radial derivatives are calculated in radial grid space using a Chebychev-transform.

s_get_br_dt.f

Called by:
s_step_time.f

Calls:
s_get_dr.f

Task:
Calculates radial derivative of special nonlinear terms and adds explicit time step contributions.

s_get_e_kin.f

Called by:
magic.f, s_step_time.f

Calls:
s_costf1.f, pos_at_end.f (in l_usefull.f)

Task:
Calculates the poloidal and toroidal kinetic energies and writes these into the appropriate output file.

s_get_e_mag.f

Called by:
magic.f, s_step_time.f

Calls:
s_costf1.f, pos_at_end.f (in l_usefull.f)

Task:
Calculates the poloidal and toroidal magnetic energies and writes these into the appropriate output file.

s_get_fl.f
s_get_fl_ic.f

Called by:
s_store_movie_frame.f, s_store_movie_frame_ic.f

Task:
Calculate scalar field necessary to draw fieldlines of the axisymmetric poloidal magnetic field in a meridional cut.

s_get_hit
_times.f

Called by:
s_prep.f

Task:
Calculates the times for the different output types.

s_get_movie_type.f

Called by:
magic.f

Task:
Indentifies the desired movies from the input strin array *movie()* and sets the necessary parameters for the run.

s_get_lorentz_torque.f

Called by:
s_step_time.f

Task:
Calculates Lorentz-torque contribution (of the colatitude block) on inner core or mantle.

s_get_mag_nl_cmb.f

Called by:
s_step_time.f

Task:
Using the terms prepared by s_get_br_v_cmb.f this subroutine calculates the nonlinear magnetic boundary condition at the CMB for a mantle with a small conductance.

s_gel_nl.f

Called by:
s_step_time.f

Task:
Finishes the calculation of nonlinear terms in the outer core.

s_gel_nl_ic.f

Called by:
s_step_time.f

Task:
Finishes the calculation of nonlinear terms in the inner core.

s_get_smat.f

Called by:
s_step_time.f

Calls:
s_sgefa.f

Task:
Calculates the implicit time step matrix for the entropy field and returns the LU-decomposition.

s_get_td.f

Called by:
s_step_time.f

Task:
Calculates first part of the explicit time step terms.

s_get_viscous _torque.f

Called by:
s_write_rot.f

Task:
Calculates viscouse torque on inner core or mantle.

s_get_vmat.f

Called by:
s_step_time.f

Calls:
s_sgefa.f

Task:
Calculates the implicit time step matrix for velocity field potentials and returns their LU-decomposition.

s_get _z10mat.f

Called by:
s_step_time.f

Calls:
s_sgefa.f

Task:
Calculates the implicit time step matrix for the toroidal velocity potential (l=1,m=0) if rigid boundary conditions have been chosen and returns the LU-decomposition.

s_graph_out.f

Called by:
s_step_time.f

Calls:
s_graph_write.f, s_filter.f, s_horiz_trans.f

Task:
Master for graphic output of outer core data.

s_graph_out_ic.f

Called by:
s_step_time.f

Calls:
s_fourtf_ic.f, s_graph_write.f

Task:
Master for graphic output of inner core data.

s_graph_write.f

Called by:
s_graph_out.f, s_graph_out_ic.f

Task:
Writes graphic output into appropriate output file.

s_horiz_trans_r.f

Called by:
s_graph_out.f

Calls:
s_fourtf.f

Task:
Given the poloidal field potential in (r, l, m)-space this routine performs a

transform to (r, θ, ϕ) -space for the radial component (factor *l(l+1)* used).

s_init_b.f

Called by:
s_prep.f

Calls:
s_j_cond.f, s_get_ddr.f, s_get_ddr_even.f

Task:
Initializes magnetic field (see above).

s_init_costf1.f

Called by:
s_rad_func.f

Task:
Initializes cos-transform of the first kind.

s_init_costf2.f

Called by:
s_rad_func.f

Task:
Initializes cos-transform of the second kind.

s_init_s.f

Called by:
s_prep.f

Calls:
s_s_cond.f, s_get_ddr.f

Task:
Initializes entropy field (see above).

s_init_v.f

Called by:
s_prep.f

Calls:
s_get_dr.f, s_get_ddr.f

Task:
Initializes velocity field (see above).

s_input.f

Called by:
s_prep.f

Calls:
s_write_namelists.f

Task:
Reads namelists, sets variables according to namelists, and writes namelists into STDOUT.

s_j_cond.f

Called by:
s_init_b.f

Calls:
s_costf1.f, s_sgefa.f, s_cgesl.f

Task:
Solves diffusion equation for the initialized magnetic toroidal field (see above).

s_leg_prep.f

Called by:
s_step_time.f

Task:
Calculates help arrays for the calculation and Legendre transform of the different fields.

s_legtf.f

Called by:
s_step_time.f

Task:
Given the results from s_leg_prep.f this routine calculates all fields and the necessary radial derivatives in (r, θ, m)-space.

s_legtf _nomag.f

Called by:
s_step_time.f

Task:
Same as s_legtf.f but does not treat the magnetic field.

s_legtf_2.f

Called by:
s_step_time.f

Task:
Performs simultaneous Legendre-transform for two arrays from (r, θ, m)-space to (r, l, m)-space.

s_legtf_3.f

Called by:
s_step_time.f

Task:
Performs simultaneous Legendre-transform for three arrays from (r, θ, m)-space to (r, l, m)-space.

s_map_data.f

Called by:
s_read_start_file.f

Calls:
s_copy_data.f

Task:
Reads fields from startfile and call s_copy_data.f to mapp to a different grid if required.

s_map_data_ic.f

Called by:
s_read_start_file.f

Calls:
s_copy_data_ic.f

Task:
Same as s_map_data.f but for inner core.

s_plm_theta.f

Called by:
s_theta_phi_func.f

Task:
Calculates associated Legendre function $P_{lm}(\cos\theta)$ and the derivative $\sin\theta\ \partial_\theta P_{lm}(\cos\theta)$ at the required colatitude points upto degree and order $l_{\max}$.

s_prep.f

Called by:
magic.f

Calls:
s_input.f, s_rad_func.f, s_theta_phi_func.f, s_read_start_file.f, s_init_s.f, s_init_b.f, s_init_v.f, s_zero_terms.f

Task:
Preforms preperatory work to be done before the time stepping like defining the initial field and calculating functions and constants needed later.

s_print_graph.f

Called by:
Daemon on interrupt kill -40

Task:
Causes graphic output in current time step.

s_rad_func.f

Called by:
s_prep.f

Calls:
s_cheb_x_map_e.f, s_get_chebs.f, s_get_chebs_even.f, s_init_costf1.f, s_init_costf2.f, check_dim.f (contained in l_usefull.f),

Task:
Prepares radial functions.

s_read_start_file.f

Called by:
s_prep.f

Calls:
s_map_data.f, s_map_data_ic.f, s_get_dr.f, s_get_ddr.f, s_get_ddr_even.f

Task:
Reads startfile (calls subroutines doing this) and prepares the input field for the first time step, i.e. scales it, calculates radial derivatives...

s_s_cond.f

Called by:
s_init_s.f

Calls:
s_sgefa.f, s_cgesl.f, s_costf1.f

Task:
Solves diffusion equation for the horizontally homogeneous entropy contribution (*l=0,m=0*) according to the heat source chosen.

s_sgefa.f

Called by:
s_get_bmat.f, s_get_smat.f, s_get_vmat.f, s_get_z10mat.f, s_j_cond.f, s_s_cond.f

Calls:
s_cheb_x_map_e.f, s_get_chebs.f, s_get_chebs_even.f, s_init_costf1.f, s_init_costf2.f, check_dim.f (contained in l_usefull.f),

Task:
Performs LU-decomposition via Gaussian elemiation. Goes back to the LINPACK.

s_sgesl.f

Called by:
s_update_s.f

Task:
Backsubstitution of LU-decomposed matricies for linear equation system for one real RHS.

s_spectrum.f

Called by:
s_step_time.f

Calls:
s_costf1.f

Task:
Calculates spectra of kinetic and magnetic energy and writes these to the appropriate output files.

s_step_time.f

Called by:
magic.f

Calls:
signal.f (unix-routine), s_get_vmat.f, s_get_smat.f, s_get_bmat.f, s_graph_out.f, s_graph_out_ic.f, s_leg_prep.f, s_legtf.f, s_legtf_nomag.f, s_fourtf.f, s_v_rigid_boundary.f, s_get_nl.f, s_get_nl_ic.f, s_legtf_2.f, s_legtf_3.f, s_get_br_v_cmb.f, s_get_lorentz_torque.f, s_courant.f, s_vbmax.f, s_lpsave.f, s_get_td.f, s_get_dr_td.f, s_get_mag_nl_cmb.f, s_dt_courant.f, s_check_time_hits.f, s_update_b.f, s_update_s.f, s_update_wp.f, s_update_z.f, s_update_omega.f, s_get_e_mag.f, s_get_e_kin.f, pos_at_end.f (contained in l_usefull.f), s_store.f, s_spectrum.f, s_fields_average.f

Task:
This subroutine performs the actual time stepping. Nearly all action including the output branches off from this routine.

s_stop _iteration.f

Called by:
Daemon on interrupt kill -30

Task:
Sets a flag that causes the iteration to performe a soft stop upon `kill -30` (see above). A soft stop means that the current time step is finished properly and a restart plus a graphic file are written.

s_store.f

Called by:
s_step_time.f

Task:
Writes all necessary information into a restart file (see above).

s_store_movie _frame.f
s_store_movie _frame_ic.f

Called by:
s_step_time.f

Task:
Store all fields required for the current frame of all movies in work array.

s_theta_phi _func.f

Called by:
s_prep.f

Calls:
s_gauleg.f, s_plm_theta.f, s_fftinit.f

Task:
Prepares horizotal functions and functions of spherical degree *l* and/or order *m*.

s_update_b.f

Called by:
s_step_time.f

Calls:
s_costf1.f, s_costf2.f, s_cgesl.f, s_cgesl2.f, s_get_ddcheb.f, s_get_ddcheb_even.f

Task:
Updates the magnetic field potentials of inner and outer core by performing the actual time step. Next explicit time step is prepared.

s_update _omega.f

Called by:
s_step_time.f

Task:
Used to update inner core and mantle rotation rates.

s_update_s.f

Called by:
s_step_time.f

Calls:
s_costf1.f, s_cgesl.f, s_cgesl2.f, s_get_ddcheb.f

Task:
Updates the entropy field by performing the actual time step. Next explicit time step is prepared.

s_update _wp.f

Called by:
s_step_time.f

Calls:
s_costf1.f, s_cgesl.f, s_cgesl2.f, s_get_dcheb.f, s_get_dddcheb.f

Task:
Updates the poloidal velocity potential and the pressure by performing the actual time step. Next explicit time step is prepared.

s_update_z.f

Called by:
s_step_time.f

Calls:
s_costf1.f, s_cgesl.f, s_cgesl2.f, s_get_ddcheb.f

Task:
Updates the toroidal velocity potential by performing the actual time step.

Next explicit time step is prepared.

s_v_rigid _boundary.f

Called by:
s_step_time.f

Task:
Sets velocities at the boundaries if rigid boundary condition have been chosen.

s_vb_comp _trans.f

Called by:
s_fields_average.f

Calls:
s_fourtf.f

Task:
Given the magnetic and velocity potentials in (r, l, m)-space this routine calculates the magnetic and velocity fields in (r, θ, ϕ)-space.

s_vbmax.f

Called by:
s_step_time.f

Task:
Calculates maxima of horizontal and radial velocity and Alfen-velocity needed to check the courant criteria.

s_write_b _cmb.f

Called by:
s_step_time.f

Task:
Writes coeffs of the poloidal magnetic field at the CMB into the CMB output-file.

s_write _movie _frame.f

Called by:
s_step_time.f

Task:
Writes the stored movie frames (s_store_movie_frame.f) into the respecti-ve movie-files.

s_write _namelists.f

Called by:
magic.f, s_input.f, input.f

Task:
Writes the namelists into an output file.

s_write_rot.f

Called by:
s_step_time.f

Calls:
s_get_viscous_torque.f

Task:
Writes rotation-information into the appropriate file (see above).

s_zero _terms.f

Called by:
s_prep.f

Task:
Sets appropriate terms to zero before the start of the time integration.

4.2 Common-blocks

Generally (nearly) all common are supplied in files with the prefix c_ . They are included in the code at compilation.

c_blocking
Contains blocking information used to speed up the run.

c_const.f
Contains several constants like π, moments of inertia of mantle and inner

core ...

c_fft_own.f
Information needed for fourier transform.

c_fields.f
All fields including their radial derivatives.

c_horizontal.f
Precomputed functions of colatitude and longitude plus functions of the spherical degree and order.

c_init_fields.f
Contains information about the start fields.

c_logic.f
Contains several logical flags that control the run.

c_mat.f
Contrains all LU-decomposed inversion matricies for the implicit time step

c_num_param.f
Contains numerical parameters.

c_output.f
Contains information on output.

c_phys_param.f
Contains physical parameters.

c_radial.f
Contains precomputed radial functions.

c_work.f
Contains work arrays.

4.3 Supplementary routines

input.f **Calls:**

s_default_namelists.f, s_write_namelists.f, s_extract_par.f, f_length_to_blank.f

Task:

Constructs script to run magic.exe.
Input.f has to be compiled and liked by running the respective makefile:

```
make -f input.make
```

This will result in the executable input.exe which is used by the script input. The paths in input.make have to be addapted to the local enviroment.
The shell-script input can be used to print another shell-script run_script for running the code (see below):

```
input tag1 [tag2]
```

The first tag will be used to identify the run --- all output files will have the extension *tag1* --- while the second tag identifies an existing result. If the second tag is supplied, the script will continue to time step the last restart file identified by *tag2* (See below for the meaning of tags.).
The executable input.exe can be used directly by supplying the two tags via STDIN.