The spherical MHD code MagIC

Fundamentals

Thomas Gastine

Institut de Physique du Globe de Paris

6th July 2017

Outline

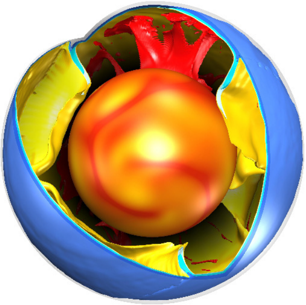
1 Introduction
- What for? How?
- Introducting MagIC

2 MHD problem

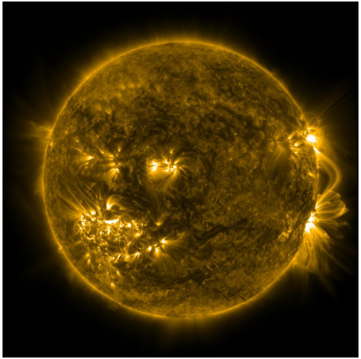
3 Installing and running the code

4 Postprocessing

What for?



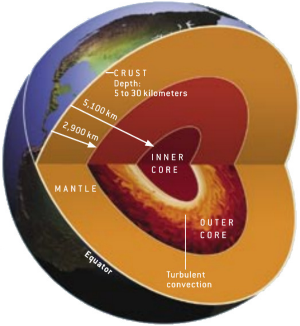
Crameri (2014)



VERIS (2013)

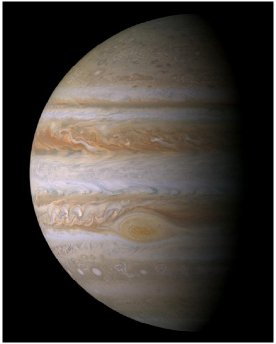
Earth's mantle **Solar convective zone**

What for?



Glatzmaier & Olson (2005)

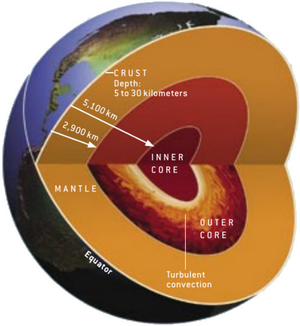
Earth's core



Cassini

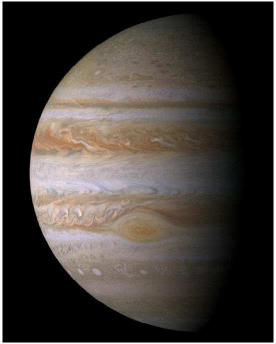
Jupiter

What for?



Glatzmaier & Olson (2005)

Earth's core

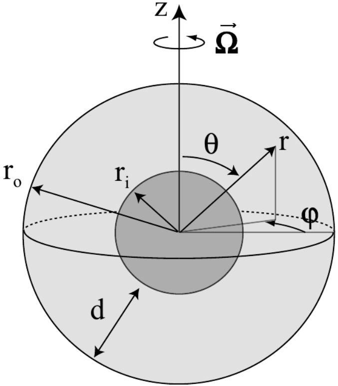


Cassini

Jupiter

Spherical geometry is more natural for studying rotating convection in astrophysical and geophysical objects!

The setup



Rotating spherical shell

Frame of reference rotating with system rotation Ω

How?

Local methods = finite differences, volume, elements?

- **PROS**: easier to implement, more straightforward to parallelise, grid refinements possible
- **CONS**: anisotropic grids, pole instability, problem with vacuum magnetic boundary condition, more points required to get same accuracy

How?

Spectral methods =expansion as complete sets of functions?

- **PROS**: derivatives easier to calculate with high accuracy, magnetic boundary condition is straightforward, lower number of grid points required
- **CONS**: parallelisation harder to implement and more communications

How?

To date spectral methods are more suitable!

"Local methods [...] need longer elapsed times than spectral methods to achieve the same accuracy with the same number of processors. Spherical harmonic expansion methods [...] offer the best assurance of efficiency for geodynamo simulations" (Matsui *et al.* 2016)

Some milestones...

1. **Chandrasekhar (1960s)**: poloidal/toroidal decomposition, onset of convection in spherical shells

Some milestones...

1. **Chandrasekhar (1960s)**: poloidal/toroidal decomposition, onset of convection in spherical shells
2. **Orszag (1970s)**: spectral methods in computational fluid dynamics

Some milestones...

1. **Chandrasekhar (1960s)**: poloidal/toroidal decomposition, onset of convection in spherical shells
2. **Orszag (1970s)**: spectral methods in computational fluid dynamics
3. **Young (1974)**: finite-amplitude convection in a Boussinesq spherical shell using a fully spectral code (roughly $\ell = m = 8$)

Some milestones...

1. **Chandrasekhar (1960s)**: poloidal/toroidal decomposition, onset of convection in spherical shells
2. **Orszag (1970s)**: spectral methods in computational fluid dynamics
3. **Young (1974)**: finite-amplitude convection in a Boussinesq spherical shell using a fully spectral code (roughly $\ell = m = 8$)
4. **Glatzmaier & Gilman (1980)**: onset of compressible convection in a spherical shell

Some milestones...

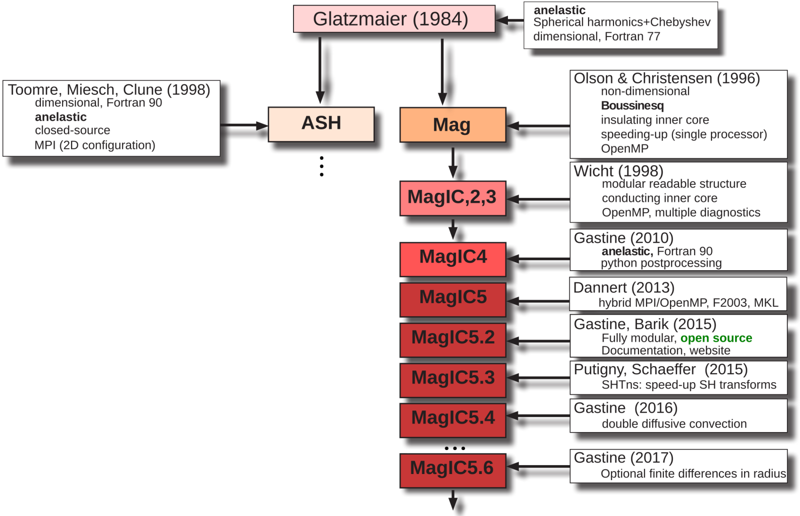
1. **Chandrasekhar (1960s)**: poloidal/toroidal decomposition, onset of convection in spherical shells
2. **Orszag (1970s)**: spectral methods in computational fluid dynamics
3. **Young (1974)**: finite-amplitude convection in a Boussinesq spherical shell using a fully spectral code (roughly $\ell = m = 8$)
4. **Glatzmaier & Gilman (1980)**: onset of compressible convection in a spherical shell
5. **Glatzmaier (1984)**: **pseudo-spectral MHD code in a spherical shell geometry**

Pseudo-spectral? What does it mean?

Pseudo-spectral codes

- **The linear terms are expanded as complete sets of functions** (e.g. spherical harmonics, Chebyshev polynomials, Fourier functions, ...)
- **Nonlinear terms treated in grid space** rather than spectral space = numerical transformations between spectral and spatial representations

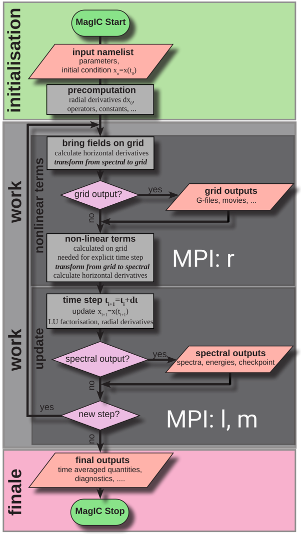
MagIC heritage



MagIC in words

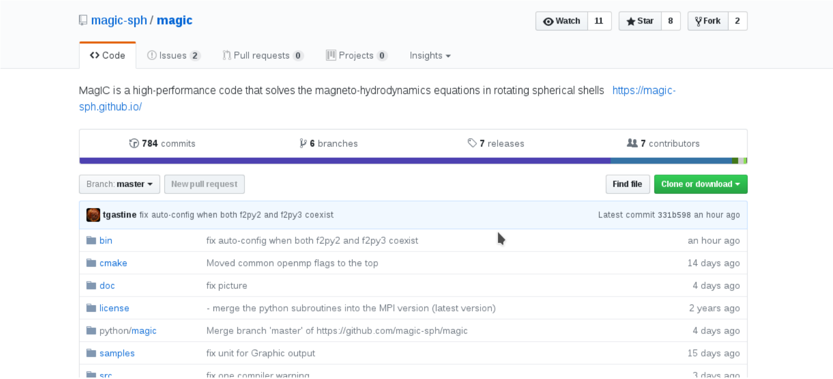
- MagIC simulates rotating fluid dynamics in a spherical shell
- It solves for the coupled evolution of Navier-Stokes equation, MHD equation, temperature (or entropy) equation and an equation for chemical composition under both the anelastic and the Boussinesq approximations
- A dimensionless formulation of the equations is assumed
- MagIC is a free software (GPL), written in Fortran
- Post-processing relies on python libraries
- Poloidal/toroidal decomposition is employed
- MagIC uses spherical harmonic decomposition in the angular directions
- Chebyshev polynomials or finite differences are employed in the radial direction
- MagIC uses a mixed implicit/explicit time stepping scheme
- The code relies on a hybrid parallelisation scheme (MPI/OpenMP)

Stucture of the code



Website and documentation

- Since 2015: MagIC is a hosted on https://github.com/magic-sph/magic



- **Online documentation**: https://magic-sph.github.io

Outline

1 Introduction

2 MHD problem
- Fully compressible equations
- From fully compressible to anelastic
- Dimensionless anelastic equations

3 Installing and running the code

4 Postprocessing

- MagIC simulates rotating fluid dynamics in a spherical shell
- It solves for the coupled evolution of Navier-Stokes equation, MHD equation, temperature (or entropy) equation and an equation for chemical composition under both the anelastic and the Boussinesq approximations
- A dimensionless formulation of the equations is assumed
- MagIC is a free software (GPL), written in Fortran
- Post-processing relies on python libraries
- Poloidal/toroidal decomposition is employed
- MagIC uses spherical harmonic decomposition in the angular directions
- Chebyshev polynomials or finite differences are employed in the radial direction
- MagIC uses a mixed implicit/explicit time stepping scheme
- The code relies on a hybrid parallelisation scheme (MPI/OpenMP)

Equation of motion for a compressible fluid

Continuity equation

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{u}) = 0$$

Navier Stokes equation:

$$\rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u} + 2\boldsymbol{\Omega} \times \boldsymbol{u} \right) = -\boldsymbol{\nabla} p + \rho \, \boldsymbol{g} + \frac{1}{\mu_0} (\boldsymbol{\nabla} \times \boldsymbol{B}) \times \boldsymbol{B} + \boldsymbol{\nabla} \cdot \mathsf{S}$$

with the rate-of-strain tensor expressed by

$$\mathsf{S}_{ij} = \nu \rho \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \delta_{ij} \, \boldsymbol{\nabla} \cdot \boldsymbol{u} \right)$$

Energy equation for a compressible fluid

$$\rho T \left(\frac{\partial s}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} s \right) = \boldsymbol{\nabla} \cdot (k_T \boldsymbol{\nabla} T) + \Phi_\nu + \lambda (\boldsymbol{\nabla} \times B)^2 + \epsilon_T$$

with the viscous heating Φ_ν expressed by

$$\Phi_\nu = 2\rho \left[e_{ij} e_{ji} - \frac{1}{3} (\boldsymbol{\nabla} \cdot \boldsymbol{u})^2 \right]$$

If in addition to that, compositional changes are also considered another equation for the chemical composition ξ reads

$$\rho \left(\frac{\partial \xi}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} \xi \right) = \boldsymbol{\nabla} \cdot (k_\xi \boldsymbol{\nabla} \xi) + \epsilon_\xi$$

Induction equation

Non-relativistic Maxwell equations provide

$$\frac{\partial \boldsymbol{B}}{\partial t} = \boldsymbol{\nabla} \times (\boldsymbol{u} \times \boldsymbol{B} - \lambda \boldsymbol{\nabla} \times \boldsymbol{B})$$

with $\boldsymbol{\nabla} \cdot \boldsymbol{B} = 0$

When λ is homogeneous, one simply gets

$$\frac{\partial \boldsymbol{B}}{\partial t} = \boldsymbol{\nabla} \times (\boldsymbol{u} \times \boldsymbol{B}) + \lambda \boldsymbol{\Delta} \boldsymbol{B}$$

Equation of state

In general:

$$p = f(\rho, T, \xi)$$

or

$$\frac{1}{\rho}\partial\rho = -\alpha\,\partial T + \beta\,\partial p + \delta\partial\xi$$

where

$$\text{Thermal expansivity: } \alpha = -\frac{1}{\rho}\left(\frac{\partial\rho}{\partial T}\right)_{\xi,p}$$

$$\text{Compressibillity: } \beta = \frac{1}{\rho}\left(\frac{\partial\rho}{\partial p}\right)_{\xi,\rho}$$

$$\text{Chemical coefficient: } \delta = \frac{1}{\rho}\left(\frac{\partial\rho}{\partial\xi}\right)_{p,\rho}$$

From fully compressible to anelastic and Boussinesq

Reference state

MHD equations

MagIC either uses the **anelastic** or the **Boussinesq** approximation of the Navier Stokes equation

Anelastic approximation = small disturbance (prime) around an **adiabatic** reference state (tilde):

$$\epsilon \sim \frac{s'}{c_p} \sim \frac{T'}{\tilde{T}} \sim \frac{\rho'}{\tilde{\rho}} \sim \frac{p'}{\tilde{p}} \sim \frac{\xi'}{\tilde{\xi}}$$

The reference state is **hydrostatic, adiabatic, and non magnetic**:

$$\boldsymbol{\nabla} \tilde{p} = \tilde{\rho}\, \boldsymbol{g}; \quad \boldsymbol{\nabla} \tilde{T} = \frac{\alpha \tilde{T}}{c_p} \boldsymbol{g}; \quad \boldsymbol{\nabla} \tilde{\xi} = 0$$

Anelastic continuity equation

Using $\rho = \tilde{\rho} + \rho'$ yields

$$\underbrace{\frac{\partial \tilde{\rho}}{\partial t}}_{=0} + \frac{\partial \rho'}{\partial t} + \boldsymbol{\nabla} \cdot (\tilde{\rho} \boldsymbol{u}) + \underbrace{\boldsymbol{\nabla} \cdot (\rho' \boldsymbol{u})}_{\mathcal{O}(\epsilon^2)} = 0$$

Estimate of the ratio

$$\frac{\partial \rho' / \partial t}{\boldsymbol{\nabla} \cdot (\tilde{\rho} \boldsymbol{u})} \sim \frac{\rho'}{\tilde{\rho}} \sim \epsilon$$

The first order anelastic equation thus reads

$$\boxed{\boldsymbol{\nabla} \cdot (\tilde{\rho} \boldsymbol{u}) = 0}$$

Anelastic equations

Navier-Stokes equation:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u} + 2\boldsymbol{\Omega} \times \boldsymbol{u} = -\boldsymbol{\nabla} \frac{p'}{\tilde{\rho}} - \frac{\tilde{\alpha} \tilde{T}}{c_p} s' \boldsymbol{g} + \frac{1}{\mu_0 \tilde{\rho}} (\boldsymbol{\nabla} \times \boldsymbol{B}) \times \boldsymbol{B} + \frac{1}{\tilde{\rho}} \boldsymbol{\nabla} \cdot \mathsf{S}$$

Energy equation:

$$\tilde{\rho} \tilde{T} \left(\frac{\partial s'}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} s' \right) = \boldsymbol{\nabla} \cdot \left(k_T \boldsymbol{\nabla} T' \right) + \Phi_\nu + \lambda \left(\boldsymbol{\nabla} \times B \right)^2 + \epsilon_T$$

Induction equation:

$$\frac{\partial \boldsymbol{B}}{\partial t} = \boldsymbol{\nabla} \times \left(\boldsymbol{u} \times \boldsymbol{B} - \lambda \boldsymbol{\nabla} \times \boldsymbol{B} \right)$$

Boundary conditions

- **Mechanical boundary conditions**:

$$\text{Stress-free: } \boldsymbol{n} \times (\mathsf{S} \cdot \boldsymbol{n}) = \boldsymbol{0}, \quad \text{or no-slip: } \boldsymbol{u} = \boldsymbol{0}, \quad r = r_i, r_o$$

- **Magnetic boundary conditions**:

$$\text{Vacuum: } \boldsymbol{\Delta B} = \boldsymbol{0}, \quad r = r_i, r_o$$

- **Thermal boundary conditions**:

$$\text{Flux: } \frac{\partial T'}{\partial r} = 0, \quad \text{or temperature: } T' = 0, \quad r = r_i, r_o$$

- MagIC simulates rotating fluid dynamics in a spherical shell
- It solves for the coupled evolution of Navier-Stokes equation, MHD equation, temperature (or entropy) equation and an equation for chemical composition under both the anelastic and the Boussinesq approximations
- **A dimensionless formulation of the equations is assumed**
- MagIC is a free software (GPL), written in Fortran
- Post-processing relies on python libraries
- Poloidal/toroidal decomposition is employed
- MagIC uses spherical harmonic decomposition in the angular directions
- Chebyshev polynomials or finite differences are employed in the radial direction
- MagIC uses a mixed implicit/explicit time stepping scheme
- The code relies on a hybrid parallelisation scheme (MPI/OpenMP)

A dimensionless formulation of the anelastic MHD equation

MHD equations

MagIC uses a **dimensionless form** of the anelastic MHD equations

In MagIC, the viscous diffusion time is assumed to be the **reference timescale** and the spherical shell gap the **reference lengthscale**:

$$[\tilde{\rho}] = \tilde{\rho}(r = r_o); \quad [\tilde{T}] = \tilde{T}(r = r_o); \quad [r] = r_o - r_i;$$
$$[t] = \frac{d^2}{\nu}; \quad [u] = \frac{\nu}{d}; \quad [B] = \sqrt{\mu_0 \lambda \tilde{\rho} \Omega}; \quad [p'] = \tilde{\rho}(r = r_o)\frac{\nu^2}{d^2}$$

This implies that the velocity is expressed in **Reynolds number** unit, and the magnetic field in **Elsasser number** unit.

Dimensionless anelastic MHD equations

In the case of an ideal gas with homogeneous kinematic viscosity ν, thermal diffusivity κ and magnetic diffusivity λ, one gets:

$$
\begin{aligned}
\boldsymbol{\nabla} \cdot (\tilde{\rho} \boldsymbol{u}) &= 0 \\
\boldsymbol{\nabla} \cdot \boldsymbol{B} &= 0 \\
\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u} + \frac{2}{E} \boldsymbol{e_z} \times \boldsymbol{u} &= -\boldsymbol{\nabla} \frac{p'}{\tilde{\rho}} + \frac{Ra}{Pr} g(r)\, s' \boldsymbol{e_r} + \frac{1}{\tilde{\rho}\, E\, Pm} (\boldsymbol{\nabla} \times \boldsymbol{B}) \times \boldsymbol{B} + \frac{1}{\tilde{\rho}} \boldsymbol{\nabla} \cdot \mathsf{S} \\
\frac{\partial \boldsymbol{B}}{\partial t} &= \boldsymbol{\nabla} \times (\boldsymbol{u} \times \boldsymbol{B}) + \frac{1}{Pm} \boldsymbol{\Delta} \boldsymbol{B} \\
\tilde{\rho} \tilde{T} \left(\frac{\partial s'}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} s' \right) &= \frac{1}{Pr} \boldsymbol{\nabla} \cdot (\tilde{\rho} \boldsymbol{\nabla} T') + \frac{Di\, Pr}{Ra} \left[\Phi_\nu + \frac{1}{Pm^2\, E} (\boldsymbol{\nabla} \times B)^2 \right]
\end{aligned}
$$

N.B. In case of compositional convection, another equation and two additional control parameters are required.

Dimensionless Boussinesq MHD equations

In the Boussinesq limit, $Di \to 0$, then

$$\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0$$

$$\boldsymbol{\nabla} \cdot \boldsymbol{B} = 0$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u} + \frac{2}{E} \boldsymbol{e_z} \times \boldsymbol{u} = -\boldsymbol{\nabla} p' + \frac{Ra}{Pr} g(r)\, T' \boldsymbol{e_r} + \frac{1}{E\ Pm} (\boldsymbol{\nabla} \times \boldsymbol{B}) \times \boldsymbol{B} + \boldsymbol{\Delta} \boldsymbol{u}$$

$$\frac{\partial \boldsymbol{B}}{\partial t} = \boldsymbol{\nabla} \times (\boldsymbol{u} \times \boldsymbol{B}) + \frac{1}{Pm} \boldsymbol{\Delta} \boldsymbol{B}$$

$$\frac{\partial T'}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} T' = \frac{1}{Pr} \boldsymbol{\Delta} T'$$

From physical properties to dimensionless numbers

Ekman number: $E = \dfrac{\nu}{\Omega\, d^2}$

Rayleigh number: $Ra = \dfrac{\alpha T_o g_o d^3 \Delta s}{c_p \nu \kappa}$

Prandtl number: $Pr = \dfrac{\nu}{\kappa}$

Magnetic Prandtl number: $Pm = \dfrac{\nu}{\lambda}$

Dissipation number: $Di = \dfrac{\alpha T_o g_o}{c_p}$

Radius ratio: $\eta = \dfrac{r_i}{r_o}$

N.B. when $Di \rightarrow 0$, the Boussinesq limit is recovered.

The (astro/geo)physical regime

Parameter	Earth's core	Giant planets	Sun
E	10^{-15}	10^{-18}	10^{-15}
Ra	10^{27}	10^{30}	10^{24}
Pr	0.1	0.1	10^{-6}
Pm	10^{-6}	10^{-7}	10^{-3}
Λ (Lorentz/Coriolis)	1	1	?
Ro_ℓ (Inertia/Coriolis)	10^{-2}	10^{-3}	1
Rm (adv./diff.)	1000	10^{5}	10^{9}
Re (adv./diff.)	10^{9}	10^{12}	10^{12}

The (astro/geo)physical regime

Parameter	Earth's core	Giant planets	Sun
E	10^{-15}	10^{-18}	10^{-15}
Ra	10^{27}	10^{30}	10^{24}
Pr	0.1	0.1	10^{-6}
Pm	10^{-6}	10^{-7}	10^{-3}
Λ (Lorentz/Coriolis)	1	1	?
Ro_ℓ (Inertia/Coriolis)	10^{-2}	10^{-3}	1
Rm (adv./diff.)	1000	10^{5}	10^{9}
Re (adv./diff.)	10^{9}	10^{12}	10^{12}

What does it actually implies? Is it possible to reach these parameters with my numerical dynamo model?

Reynolds number: the range of length-scale

$$Re = \frac{u_{rms}\, d}{\nu} = \frac{d}{\ell_d} \quad \text{where} \quad \ell_d = \frac{\nu}{u_{rms}}$$

$$\ell_d = \frac{d}{Re}$$

Reynolds number: the range of length-scale

$$Re = \frac{u_{rms}\, d}{\nu} = \frac{d}{\ell_d} \quad \text{where} \quad \ell_d = \frac{\nu}{u_{rms}}$$

$$\ell_d = \frac{d}{Re}$$

- In natural objects, $l_d \sim 10^{-9}\, d$
- In other words, the ratio of the bigger length-scale to the smallest one is 10^9.
- **You might need 10^9 grid points in each direction**. This implies $Re_{mesh} = 1$.

Ekman number: the range of time-scales

$$E = \frac{\nu}{\Omega\, d^2} = \frac{P_{rot}}{\tau_\nu} \quad \text{where} \quad \tau_\nu = \frac{d^2}{\nu}$$

τ_ν is the viscous diffusion time, P_{rot} the rotation period.

$$\tau_\nu = \frac{P_{rot}}{E}$$

Ekman number: the range of time-scales

$$E = \frac{\nu}{\Omega\, d^2} = \frac{P_{rot}}{\tau_\nu} \quad \text{where} \quad \tau_\nu = \frac{d^2}{\nu}$$

τ_ν is the viscous diffusion time, P_{rot} the rotation period.

$$\tau_\nu = \frac{P_{rot}}{E}$$

- In natural objects, $\tau_\nu \sim 10^{15}\, P_{rot}$
- In other words, the ratio of the longest time-scale to the smallest one is 10^{15}!
- **You might need 10^{15} time steps to model the problem**

Summary

Parameter	Earth's core	Tractable	Hard limit (2015)
E	10^{-15}	$\geq 10^{-6}$	10^{-7}
Ra	10^{27}	$\leq 10^{12}$	10^{13}
Pr	0.1	$0.1 - 10$	1
Pm	10^{-6}	0.1	6×10^{-2}
Λ (Lorentz/Coriolis)	1	1	1
Ro_ℓ (Inertia/Coriolis)	10^{-2}	$10^{-3} - 10^{-1}$	10^{-1}
Rm (adv./diff.)	1000	1000	1000
Re (adv./diff.)	10^{9}	$100 - 1000$	7000

Two complementary approaches

- In the **“tractable” regime**: parameter studies are possible
- In the **“hard-limit” regime**, only one single run is possible

Outline

1 Introduction

2 MHD problem

3 Installing and running the code
- Requirements and compilation
- Executing MagIC

4 Postprocessing

- MagIC simulates rotating fluid dynamics in a spherical shell
- It solves for the coupled evolution of Navier-Stokes equation, MHD equation, temperature (or entropy) equation and an equation for chemical composition under both the anelastic and the Boussinesq approximations
- A dimensionless formulation of the equations is assumed
- MagIC is a free software (GPL), written in Fortran
- Post-processing relies on python libraries
- Poloidal/toroidal decomposition is employed
- MagIC uses spherical harmonic decomposition in the angular directions
- Chebyshev polynomials or finite differences are employed in the radial direction
- MagIC uses a mixed implicit/explicit time stepping scheme
- The code relies on a hybrid parallelisation scheme (MPI/OpenMP)

Requirements to compile MagIC

Requirements

Mandatory Fortran and C compilers

Suggested git (https://git-scm.com/) to clone the code repository

Suggested CMake (https://cmake.org) to build the code

Suggested MPI library: rather use intelMPI or MPICH for full support for hybrid MPI/OpenMP

Optional LAPACK or MKL

Optional SHTns for spherical harmonics transforms

Data visualisation and post processing

Requirements

Post-processing functions are python based. You need to install the following libraries:

Python libraries required

Mandatory `matplotlib` (https://matplotlib.org): plotting functions

Mandatory `scipy` (https://www.scipy.org): scientific libraries

Suggested `ipython` (https://ipython.org): interactive shell

Optional `basemap` (https://matplotlib.org/basemap/): additional map projections

Get the code and compile it

1. **Install requirements**:

```
$ module load gcc-6 gfortran-6 libopenmpi cmake git
$ module load python27 python-scipy  ipython  python-matplotlib
```

2. **Clone the code from** `github`

```
$ git clone https://github.com/magic-sph/magic.git
```

3. **Set-up the environment variables**

```
$ cd magic
$ source sourceme.sh # (or sourceme.csh)
```

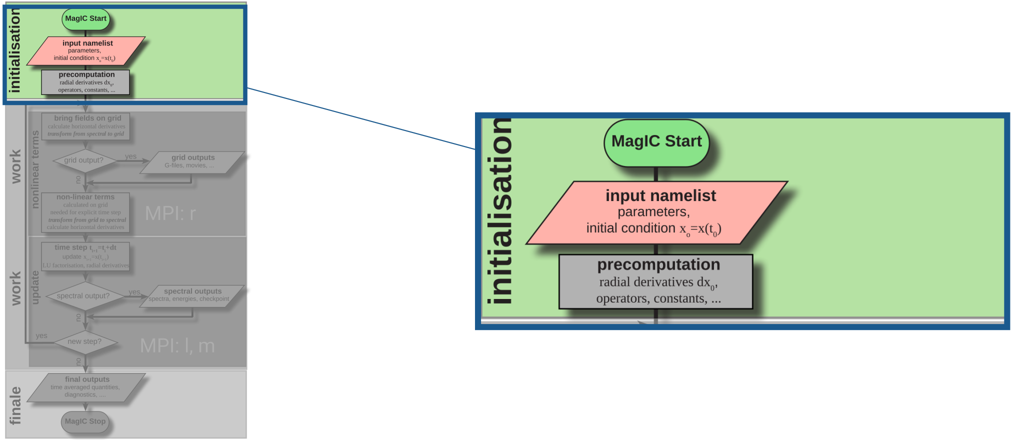
4. **Define the Fortran and C compilers**

```
$ export FC=mpif90 # replace by your compiler
$ export CC=mpicc
```

5. **Create a build directory and compile**

```
$ mkdir build; cd build
$ cmake $MAGIC_HOME -DUSE_MPI=yes -DUSE_OMP=no
$ make -j
```

MagIC structure



Run MagIC

Run with 8 CPUs:

```
$ export OMP_NUM_THREADS=1
$ mpiexec -n 8 magic.exe input.nml
```

`input.nml` contains all the input informations required to run the code!

Documentation

Input namelist (1/3)

```
&grid
 n_r_max      =33,  ! Radial resolution
 n_cheb_max   =31,  ! Number of Chebyshev polynomials
 n_phi_tot    =192, ! Azimuthal resolution
 minc         =1,   ! Azimuthal symmetry
/
&control
 mode         =0,       ! Magnetic, non-magnetic, ...
 tag          ="test",  ! Extension of the output files
 n_time_steps=40000,    ! Number of timesteps
 dtmax        =1.0D-4,  ! Maximum timestep
 runHours     =02,      ! Run-time
 runMinutes   =00,
/
```

Input namelist (2/3)

```
&phys_param
 ra          =1.1D5,  ! Rayleigh number
 ek          =1.0D-3, ! Ekman number
 pr          =1.0D0,  ! Prandtl number
 prmag       =5.0D0   ! Magnetic Prandtl number
 radratio    =0.35D0, ! Radius ratio r_i/r_o
 ktops       =1,      ! BC: fixed-temperature at the top
 ktopv       =2,      ! BC: rigid wall at the top
/
&start_field
 l_start_file=.false.,                ! Start from a check point?
 start_file  ="checkpoint_end.start", ! Name of the check point
 init_b1     =3,                      ! Init. mag. field: dipole
 amp_b1      =1,                      ! Amplitude \Lambda=1
 init_s1     =0404,                   ! Init. temperature perturbation
 amp_s1      =0.03,                   ! Amplitude of the init. pert.
/
```

Input namelist (3/3)

```
&output_control
 n_log_step  =50,    ! Output every n_log_step
 n_graphs    =3,     ! Number of graphic files
 n_rsts      =1,     ! Number of restart files
 n_stores    =0,
 n_specs     =1,     ! Number of spectra
/
&mantle
 nRotMa      =0
/
&inner_core
 sigma_ratio =1.d0,   ! Conducting inner-core
 nRotIC      =1,      ! Rotating inner core
/
```

Outline

1 Introduction

2 MHD problem

3 Installing and running the code

4 Postprocessing

log.TAG file

`log.TAG` **provides all the important information about the run**:

- All parameters and other inputs including default values
- Information on parallelization, run time etc
- Log of important events: important output files, changing time step, ...
- Some important time averaged quantities, measures ...

Documentation

Plotting time series

`e_kin.TAG` **is always produced. It contains the time evolution of kinetic energy. To plot it:**

- Open ipython and load the python modules

```
ipython --matplotlib=gtk (or ipython --pylab)
>>> from magic import *
>>> ts = MagicTs(field='e_kin') # Read e_kin.TAG file in $PWD
>>> pdoc MagicTs # Gives you the documentation
```

- Plot the time evolution of magnetic energy

```
>>> ts = MagicTs(field='e_mag_oc') # Read e_mag_oc.TAG file in $PWD
```

- Manipulate the data

```
>>> print(ts.time, ts.emagoc_pol)
```

▸ Documentation

Loading and plotting snapshots

`G_#.TAG` **files contain 3-D arrays on the grid**:

- Load the `G_1.TAG` file:

```
>>> from magic import *
>>> s = Surf(ivar=1)
```

- Plot the radial velocity u_r in the equatorial plane:

```
>>> s.equat(field='vr')
```

- Plot the ϕ-averaged azimuthal flow u_ϕ:

```
>>> s.avg(field='vp', cm='seismic', levels=33)
```

- Plot the radial cut of B_r at $r = 0.75\, r_o$:

```
>>> s.surf(field='Br', r=0.75) # Hammer projection
```

▸ Documentation

Data visualisation and post processing

Additional outputs

- Plot spectra kin_spec_1.TAG

```
>>> # Plot kin_spec_1.TAG
>>> sp = MagicSpectrum(field='kin', ispec=1)
```

Documentation

- Plot the time-averaged radial profile of magnetic energy eMagR.TAG

```
>>> # Plot eMagR.TAG
>>> r = MagicRadial(field='eMagR')
```

Documentation

- And more...

```
>>> # Movie files (time evolution of 2D slices)
>>> m = Movie()
```

Documentation

Data visualisation and post processing

3-D visualisation with paraview

Requirements

Install a vtk-friendly software: here `paraview` but `VisIt` or `mayavi` should also work fine.

1. Read the graphic file you want to convert

```
>>> from magic import MagicGraph
>>> gr = MagicGraph(ivar=1) # Load G_1.TAG
```

2. Convert it to a file format readable by `paraview`

```
>>> # Produce output.vts
>>> Graph2Vtk(gr, filename='output')
```

3. Load `output.vts` with `paraview`

```
$ paraview output.vts
```

Documentation