The spherical MHD code MagIC
Fundamentals

Thomas Gastine

Institut de Physique du Globe de Paris

6th July 2017
Outline

1. Introduction
   - What for? How?
   - Introducing MagIC

2. MHD problem

3. Installing and running the code

4. Postprocessing
What for?

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

Earth’s mantle

Solar convective zone

Crameri (2014)
VERIS (2013)
Spherical geometry is more natural for studying rotating convection in astrophysical and geophysical objects!
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 $\Omega$
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
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
Chandrasekhar (1960s): poloidal/toroidal decomposition, onset of convection in spherical shells

Orszag (1970s): spectral methods in computational fluid dynamics

Young (1974): finite-amplitude convection in a Boussinesq spherical shell using a fully spectral code (roughly $\ell = m = 8$)

Glatzmaier & Gilman (1980): onset of compressible convection in a spherical shell

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 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)
Structure of the code
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

\[ \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \]

Navier Stokes equation:

\[ \rho \left( \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + 2\Omega \times \mathbf{u} \right) = -\nabla p + \rho \mathbf{g} + \frac{1}{\mu_0} (\nabla \times \mathbf{B}) \times \mathbf{B} + \nabla \cdot \mathbf{S} \]

with the rate-of-strain tensor expressed by

\[ S_{ij} = \nu \rho \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \delta_{ij} \nabla \cdot \mathbf{u} \right) \]
Energy equation for a compressible fluid

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

with the viscous heating \( \Phi_\nu \) expressed by

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

If in addition to that, compositional changes are also considered another equation for the chemical composition \( \xi \) reads

\[ \rho \left( \frac{\partial \xi}{\partial t} + u \cdot \nabla \xi \right) = \nabla \cdot \left( k_\xi \nabla \xi \right) + \epsilon_\xi \]
Induction equation

Non-relativistic Maxwell equations provide

\[ \frac{\partial B}{\partial t} = \nabla \times (u \times B - \lambda \nabla \times B) \]

with \( \nabla \cdot B = 0 \)

When \( \lambda \) is homogeneous, one simply gets

\[ \frac{\partial B}{\partial t} = \nabla \times (u \times B) + \lambda \Delta B \]
In general:

\[ p = f(\rho, T, \xi) \]

or

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

where

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

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

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

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

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

\[ \nabla \tilde{p} = \tilde{\rho} \mathbf{g}; \quad \nabla \tilde{T} = \frac{\alpha}{c_p} \tilde{T} \mathbf{g}; \quad \nabla \tilde{\xi} = 0 \]
Anelastic continuity equation

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

$$\frac{\partial \tilde{\rho}}{\partial t} + \frac{\partial \rho'}{\partial t} + \nabla \cdot (\tilde{\rho} u) + \nabla \cdot (\rho' u) = 0$$

Estimate of the ratio

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

The first order anelastic equation thus reads

$$\nabla \cdot (\tilde{\rho} u) = 0$$
Anelastic equations

Navier-Stokes equation:

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

Energy equation:

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

Induction equation:

\[
\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{u} \times \mathbf{B} - \lambda \nabla \times \mathbf{B})
\]
Boundary conditions

- **Mechanical boundary conditions:**
  
  Stress-free: \( n \times (S \cdot n) = 0 \), or no-slip: \( u = 0, \quad r = r_i, r_o \)

- **Magnetic boundary conditions:**
  
  Vacuum: \( \Delta B = 0, \quad r = r_i, r_o \)

- **Thermal boundary conditions:**
  
  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 $\nu$, thermal diffusivity $\kappa$ and magnetic diffusivity $\lambda$, one gets:

\[
\nabla \cdot (\tilde{\rho} \mathbf{u}) = 0 \\
\nabla \cdot \mathbf{B} = 0
\]

\[
\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{2}{E} \mathbf{e}_z \times \mathbf{u} = -\nabla \frac{p'}{\tilde{\rho}} + \frac{Ra}{Pr} g(r) s' \mathbf{e}_r + \frac{1}{\tilde{\rho} E Pm} (\nabla \times \mathbf{B}) \times \mathbf{B} + \frac{1}{\tilde{\rho}} \nabla \cdot \mathbf{S}
\]

\[
\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{u} \times \mathbf{B}) + \frac{1}{Pm} \Delta \mathbf{B}
\]

\[
\tilde{\rho} \tilde{T} \left( \frac{\partial s'}{\partial t} + \mathbf{u} \cdot \nabla s' \right) = \frac{1}{Pr} \nabla \cdot (\tilde{\rho} \nabla T') + \frac{Di Pr}{Ra} \left[ \Phi_{\nu} + \frac{1}{Pm^2 E} (\nabla \times \mathbf{B})^2 \right]
\]

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

\[
\begin{align*}
\nabla \cdot u &= 0 \\
\nabla \cdot B &= 0 \\
\frac{\partial u}{\partial t} + u \cdot \nabla u + \frac{2}{E} e_z \times u &= -\nabla p' + \frac{Ra}{Pr} g(r) T'e_r + \frac{1}{EPm} (\nabla \times B) \times B + \Delta u \\
\frac{\partial B}{\partial t} &= \nabla \times (u \times B) + \frac{1}{Pm} \Delta B \\
\frac{\partial T'}{\partial t} + u \cdot \nabla T' &= \frac{1}{Pr} \Delta T'
\end{align*}
\]
From physical properties to dimensionless numbers

Ekman number: \[ E = \frac{\nu}{\Omega d^2} \]

Rayleigh number: \[ Ra = \frac{\alpha T_o g_o d^3 \Delta s}{c_p \nu \kappa} \]

Prandtl number: \[ Pr = \frac{\nu}{\kappa} \]

Magnetic Prandtl number: \[ Pm = \frac{\nu}{\lambda} \]

Dissipation number: \[ Di = \frac{\alpha T_o g_o}{c_p} \]

Radius ratio: \[ \eta = \frac{r_i}{r_o} \]

N.B. when \( Di \to 0 \), the Boussinesq limit is recovered.
### The (astro/geo)physical regime

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Earth’s core</th>
<th>Giant planets</th>
<th>Sun</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$</td>
<td>$10^{-15}$</td>
<td>$10^{-18}$</td>
<td>$10^{-15}$</td>
</tr>
<tr>
<td>$Ra$</td>
<td>$10^{27}$</td>
<td>$10^{30}$</td>
<td>$10^{24}$</td>
</tr>
<tr>
<td>$Pr$</td>
<td>$0.1$</td>
<td>$0.1$</td>
<td>$10^{-6}$</td>
</tr>
<tr>
<td>$Pm$</td>
<td>$10^{-6}$</td>
<td>$10^{-7}$</td>
<td>$10^{-3}$</td>
</tr>
<tr>
<td>$\Lambda$ (Lorentz/Coriolis)</td>
<td>$1$</td>
<td>$1$</td>
<td>?</td>
</tr>
<tr>
<td>$Ro_\ell$ (Inertia/Coriolis)</td>
<td>$10^{-2}$</td>
<td>$10^{-3}$</td>
<td>$1$</td>
</tr>
<tr>
<td>$Rm$ (adv./diff.)</td>
<td>$1000$</td>
<td>$10^5$</td>
<td>$10^9$</td>
</tr>
<tr>
<td>$Re$ (adv./diff.)</td>
<td>$10^9$</td>
<td>$10^{12}$</td>
<td>$10^{12}$</td>
</tr>
</tbody>
</table>

What does it actually implies? Is it possible to reach these parameters with my numerical dynamo model?
The (astro/geo)physical regime

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Earth’s core</th>
<th>Giant planets</th>
<th>Sun</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$</td>
<td>$10^{-15}$</td>
<td>$10^{-18}$</td>
<td>$10^{-15}$</td>
</tr>
<tr>
<td>$Ra$</td>
<td>$10^{27}$</td>
<td>$10^{30}$</td>
<td>$10^{24}$</td>
</tr>
<tr>
<td>$Pr$</td>
<td>0.1</td>
<td>0.1</td>
<td>$10^{-6}$</td>
</tr>
<tr>
<td>$Pm$</td>
<td>$10^{-6}$</td>
<td>$10^{-7}$</td>
<td>$10^{-3}$</td>
</tr>
<tr>
<td>$\Lambda$ (Lorentz/Coriolis)</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>$Ro_\ell$ (Inertia/Coriolis)</td>
<td>$10^{-2}$</td>
<td>$10^{-3}$</td>
<td>1</td>
</tr>
<tr>
<td>$Rm$ (adv./diff.)</td>
<td>1000</td>
<td>$10^5$</td>
<td>$10^9$</td>
</tr>
<tr>
<td>$Re$ (adv./diff.)</td>
<td>$10^9$</td>
<td>$10^{12}$</td>
<td>$10^{12}$</td>
</tr>
</tbody>
</table>

What does it actually implies? Is it possible to reach these parameters with my numerical dynamo model?
Reynolds number: the range of length-scale

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

\[ \ell_d = \frac{d}{Re} \]
Reynolds number: the range of length-scale

\[ Re = \frac{u_{\text{rms}} d}{\nu} = \frac{d}{\ell_d} \quad \text{where} \quad \ell_d = \frac{\nu}{u_{\text{rms}}} \]

\[ \ell_d = \frac{d}{Re} \]

- 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 \( Re_{\text{mesh}} = 1 \).
Ekman number: the range of time-scales

\[ E = \frac{\nu}{\Omega d^2} = \frac{P_{\text{rot}}}{\tau_{\nu}} \text{ where } \tau_{\nu} = \frac{d^2}{\nu} \]

\( \tau_{\nu} \) is the viscous diffusion time, \( P_{\text{rot}} \) the rotation period.

\[ \tau_{\nu} = \frac{P_{\text{rot}}}{E} \]
Ekman number: the range of time-scales

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

\( \tau_{\nu} \) is the viscous diffusion time, \( P_{\text{rot}} \) the rotation period.

- In natural objects, \( \tau_{\nu} \sim 10^{15} \ P_{\text{rot}} \)
- 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Earth’s core</th>
<th>Tractable</th>
<th>Hard limit (2015)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$</td>
<td>$10^{-15}$</td>
<td>$\geq 10^{-6}$</td>
<td>$10^{-7}$</td>
</tr>
<tr>
<td>$Ra$</td>
<td>$10^{27}$</td>
<td>$\leq 10^{12}$</td>
<td>$10^{13}$</td>
</tr>
<tr>
<td>$Pr$</td>
<td>0.1</td>
<td>0.1 – 10</td>
<td>1</td>
</tr>
<tr>
<td>$Pm$</td>
<td>$10^{-6}$</td>
<td>0.1</td>
<td>$6 \times 10^{-2}$</td>
</tr>
<tr>
<td>$\Lambda$ (Lorentz/Coriolis)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$Ro_\ell$ (Inertia/Coriolis)</td>
<td>$10^{-2}$</td>
<td>$10^{-3} – 10^{-1}$</td>
<td>$10^{-1}$</td>
</tr>
<tr>
<td>$Rm$ (adv./diff.)</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>$Re$ (adv./diff.)</td>
<td>$10^{9}$</td>
<td>100 – 1000</td>
<td>7000</td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mandatory</strong></td>
<td>Fortran and C compilers</td>
</tr>
<tr>
<td><strong>Suggested</strong></td>
<td>git (<a href="https://git-scm.com/">https://git-scm.com/</a>) to clone the code repository</td>
</tr>
<tr>
<td><strong>Suggested</strong></td>
<td>CMake (<a href="https://cmake.org">https://cmake.org</a>) to build the code</td>
</tr>
<tr>
<td><strong>Suggested</strong></td>
<td>MPI library: rather use intelMPI or MPICH for full support for hybrid MPI/OpenMP</td>
</tr>
<tr>
<td><strong>Optional</strong></td>
<td>LAPACK or MKL</td>
</tr>
<tr>
<td><strong>Optional</strong></td>
<td>SHTns for spherical harmonics transforms</td>
</tr>
</tbody>
</table>
Data visualisation and post processing

Requirements

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

<table>
<thead>
<tr>
<th>Python libraries required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mandatory</strong> matplotlib (<a href="https://matplotlib.org">https://matplotlib.org</a>): plotting functions</td>
</tr>
<tr>
<td><strong>Mandatory</strong> scipy (<a href="https://www.scipy.org">https://www.scipy.org</a>): scientific libraries</td>
</tr>
<tr>
<td><strong>Suggested</strong> ipython (<a href="https://ipython.org">https://ipython.org</a>): interactive shell</td>
</tr>
<tr>
<td><strong>Optional</strong> basemap (<a href="https://matplotlib.org/basemap/">https://matplotlib.org/basemap/</a>): additional map projections</td>
</tr>
</tbody>
</table>
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 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!
&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,
&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
/

&mantine
  nRotMa     =0
/

&inner_core
  sigma_ratio =1.d0,  ! Conducting inner-core
  nRotIC      =1,    ! Rotating inner core
/
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 ...
e_kin.TAG is always produced. It contains the time evolution of kinetic energy. To plot it:

- Open ipython and load the python modules
  ```python
  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
  ```python
  >>> ts = MagicTs(field='e_mag_oc') # Read e_mag_oc.TAG file in $PWD
  ```

- Manipulate the data
  ```python
  >>> print(ts.time, ts.emagoc_pol)
  ```
G_#.TAG files contain 3-D arrays on the grid:

- Load the G_1.TAG file:
  ```python
  >>> from magic import *
  >>> s = Surf(ivar=1)
  ```
- Plot the radial velocity $u_r$ in the equatorial plane:
  ```python
  >>> s.equat(field='vr')
  ```
- Plot the $\phi$-averaged azimuthal flow $u_\phi$:
  ```python
  >>> s.avg(field='vp', cm='seismic', levels=33)
  ```
- Plot the radial cut of $B_r$ at $r = 0.75 r_0$:
  ```python
  >>> s.surf(field='Br', r=0.75) # Hammer projection
  ```
Data visualisation and post processing

Additional outputs

- Plot spectra $\text{kin\_spec\_1\_TAG}$
  
  ```python
  >>> # Plot kin_spec_1.TAG
  >>> sp = MagicSpectrum(field='kin', ispec=1)
  ```

- Plot the time-averaged radial profile of magnetic energy $\text{eMagR.TAG}$
  
  ```python
  >>> # Plot eMagR.TAG
  >>> r = MagicRadial(field='eMagR')
  ```

- And more...
  
  ```python
  >>> # Movie files (time evolution of 2D slices)
  >>> m = Movie()
  ```
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