



MagIC

# Introduction to MagIC

Johannes Wicht and Thomas Gastine (IPGP)

# magic-sph on gitHub

magic-sph / magic

Search

Type / to search

<> Code

Issues 2

Pull requests 2

Actions

Projects

Wiki

Security

Insights

Settings

magic

Public

Edit Pins

Unwatch 23

Fork 35

Star 87

master 14 Branches 15 Tags

Go to file

Add file

<> Code

About

AnkitBarik

Fix dsigma for nVarCond=1

9cab422 · 2 months ago

2,271 Commits

.github/workflows	rather make use of the package of the distribution instead o...	last year
bin	Cleanup, restore older simple check	3 months ago
cmake	raise minimum Cmake version to 3.5 to remove warnings	last year
doc	update conf.py	9 months ago
license	- merge the python subroutines into the MPI version (latest ...	10 years ago
python/magic	Fix for matplotlib > 3.8	3 months ago
samples	fix directory cleanups in unitTest(s).py	9 months ago
src	Fix dsigma for nVarCond=1	2 months ago
submitscripts	clean resub.py	2 years ago
.gitignore	update gitignore files	last year

MagIC is a high-performance code that solves the magneto-hydrodynamics equations in rotating spherical shells

[magic-sph.github.io/](#)

cfddconvectionmhd

computational-fluid-dynamics

magnetohydrodynamicsplanetary-science

stellar-astrophysics

Readme

Activity

Custom properties

87 stars

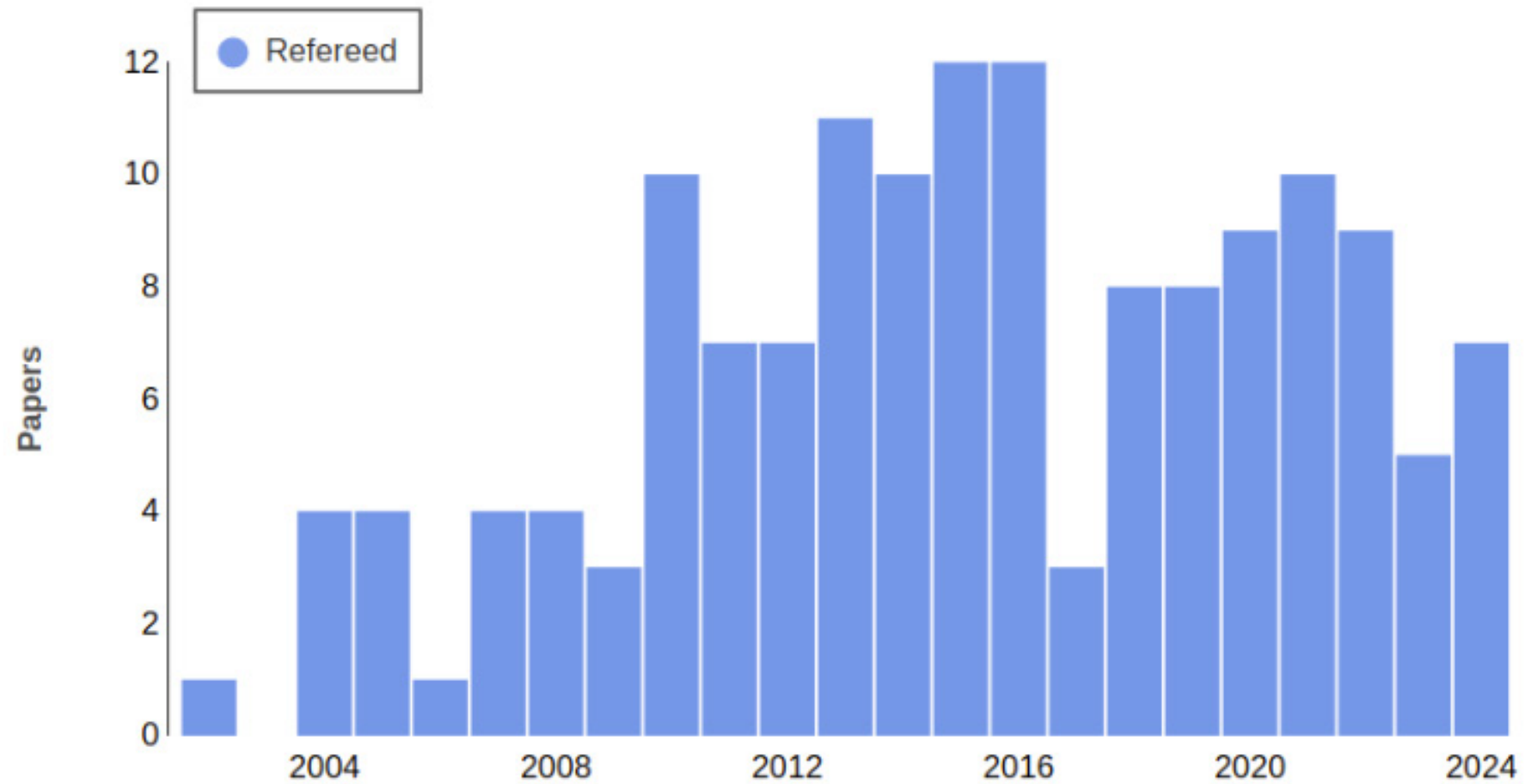
23 watching

35 forks

Report repository

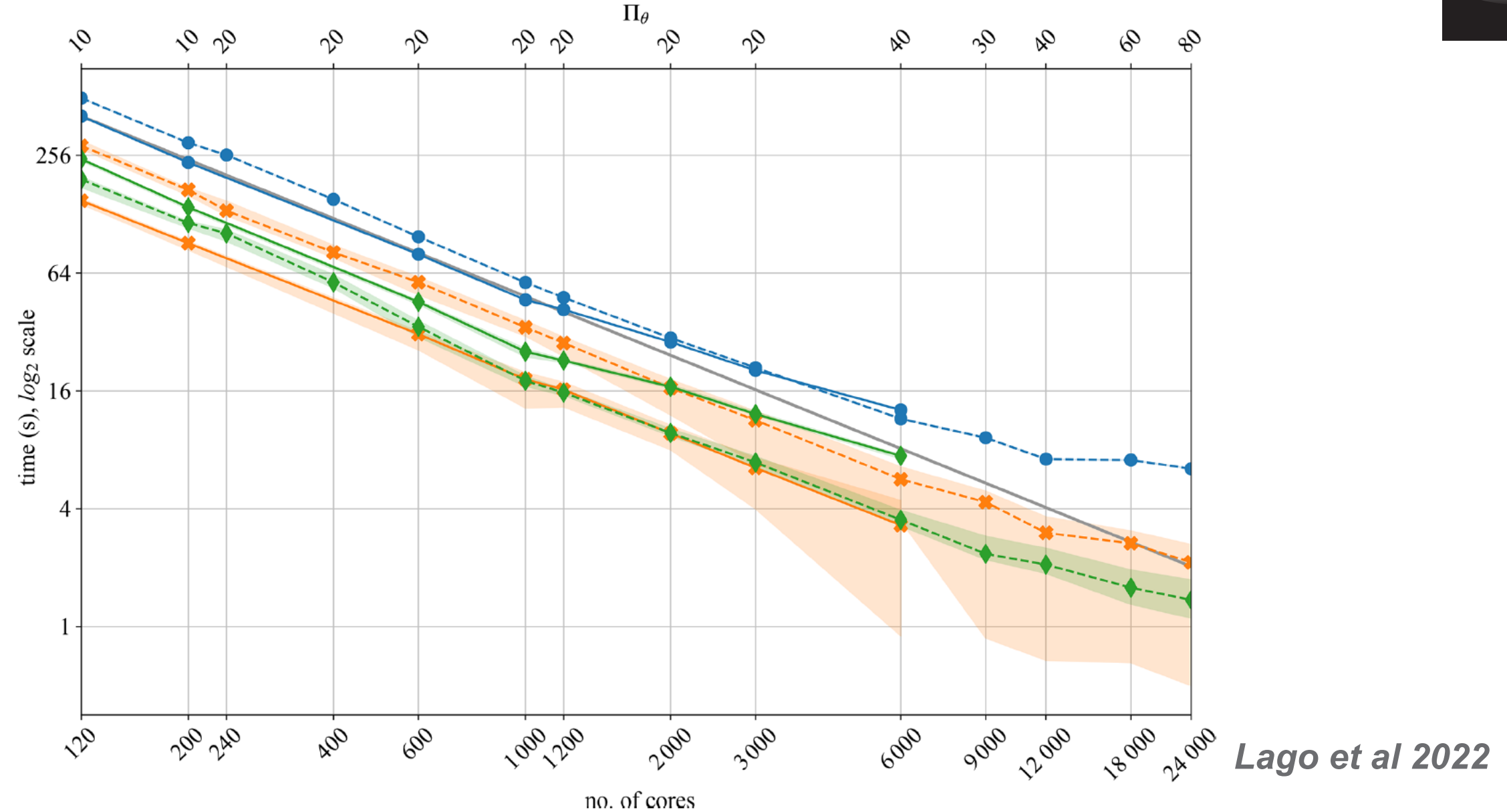
Available on gitHub.

# Publication Record



- More than 150 peer-reviewed publications!
- Used for many different problems.

# Efficient Parallelization

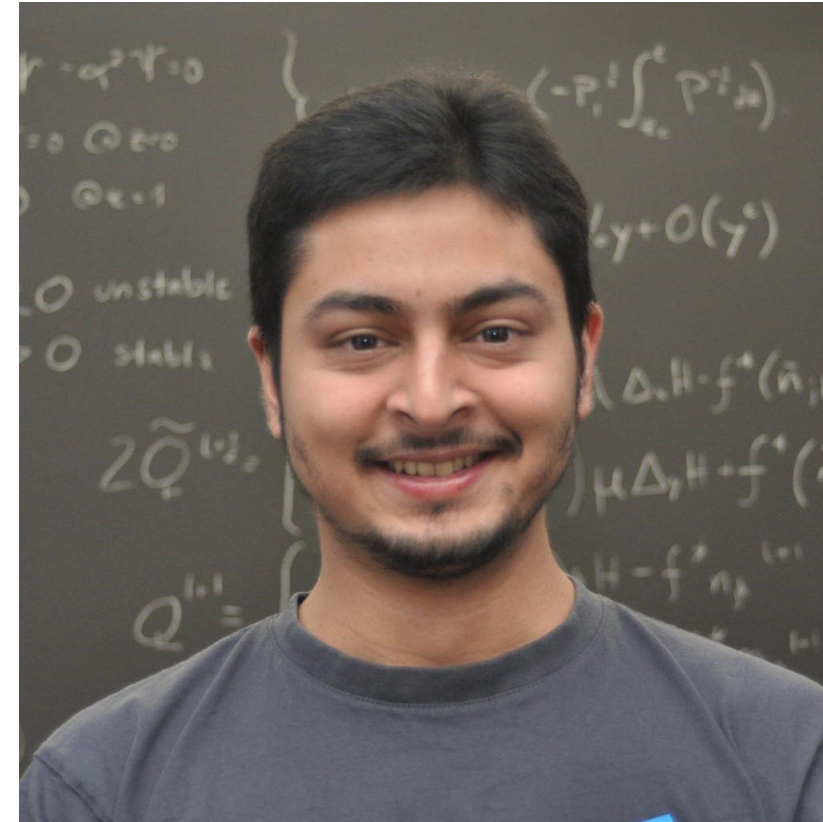


- Fast, highly parallelized, runs on super computers but also notebooks.
- Efficient use of several thousand cores for large problems.

# Well Maintained



*Thomas Gastine*



*Ankit Barit*

- Maintained by experienced scientists.

# Documentation



## Table of Contents

[Contents](#)  
[Indices and tables](#)

## Next topic

[Introduction](#)

## This Page

[Show Source](#)

## Quick search

## Contents

- [Get MagIC and run it](#)
  - [Download the code](#)
  - [Setting up the environment variables](#)
  - [Install SHTns \(\*\*recommended\*\*\)](#)
  - [Setting up compiler options and compiling](#)
  - [Preparing a production run](#)
- [Formulation of the \(magneto\)-hydrodynamics problem](#)
  - [The reference state](#)
  - [Boussinesq approximation](#)
  - [Anelastic approximation](#)
  - [Dimensionless control parameters](#)
  - [Boundary conditions and treatment of inner core](#)
- [Numerical technique](#)
  - [Poloidal/toroidal decomposition](#)
  - [Spherical harmonic representation](#)
  - [Radial representation](#)
  - [Spectral equations](#)
  - [Time-stepping schemes](#)
  - [Coriolis force and nonlinear terms](#)
  - [Boundary conditions and inner core](#)
- [Contributing to the code](#)
  - [Checking the consistency of the code](#)
  - [Advices when contributing to the code](#)
  - [Building the documentation and contributing to it](#)

● Extensive online documentation.







HomeGet it/Run itContribute!Numerical methodsContents

Magic 6.3 documentation » Data visualisation and post-processing » Support for G\_#.TAG files

previous | next | modules | fortran modules | index

Table of Contents

Support for G\_#.TAG files

- MagicGraph
  - MagicGraph.\_\_init\_\_()
  - MagicGraph.read\_record\_markers()
  - MagicGraph.read\_stream()
  - MagicGraph.rearrangeLat()
- Surf
  - Surf.\_\_init\_\_()
  - Surf.\_\_weakref\_\_
  - Surf.avg()
  - Surf.equat()
  - Surf.slice()
  - Surf.surf()

Support for checkpoint\_#.TAG files

- Graph2Rst()
- MagicCheckpoint
  - MagicCheckpoint.\_\_init\_\_()

## Support for G\_#.TAG files

```
class magic.MagicGraph(ivar=None, datadir='.', quiet=True, ave=False, tag=None, precision=<class 'numpy.float32'>)
```

[\[source\]](#)

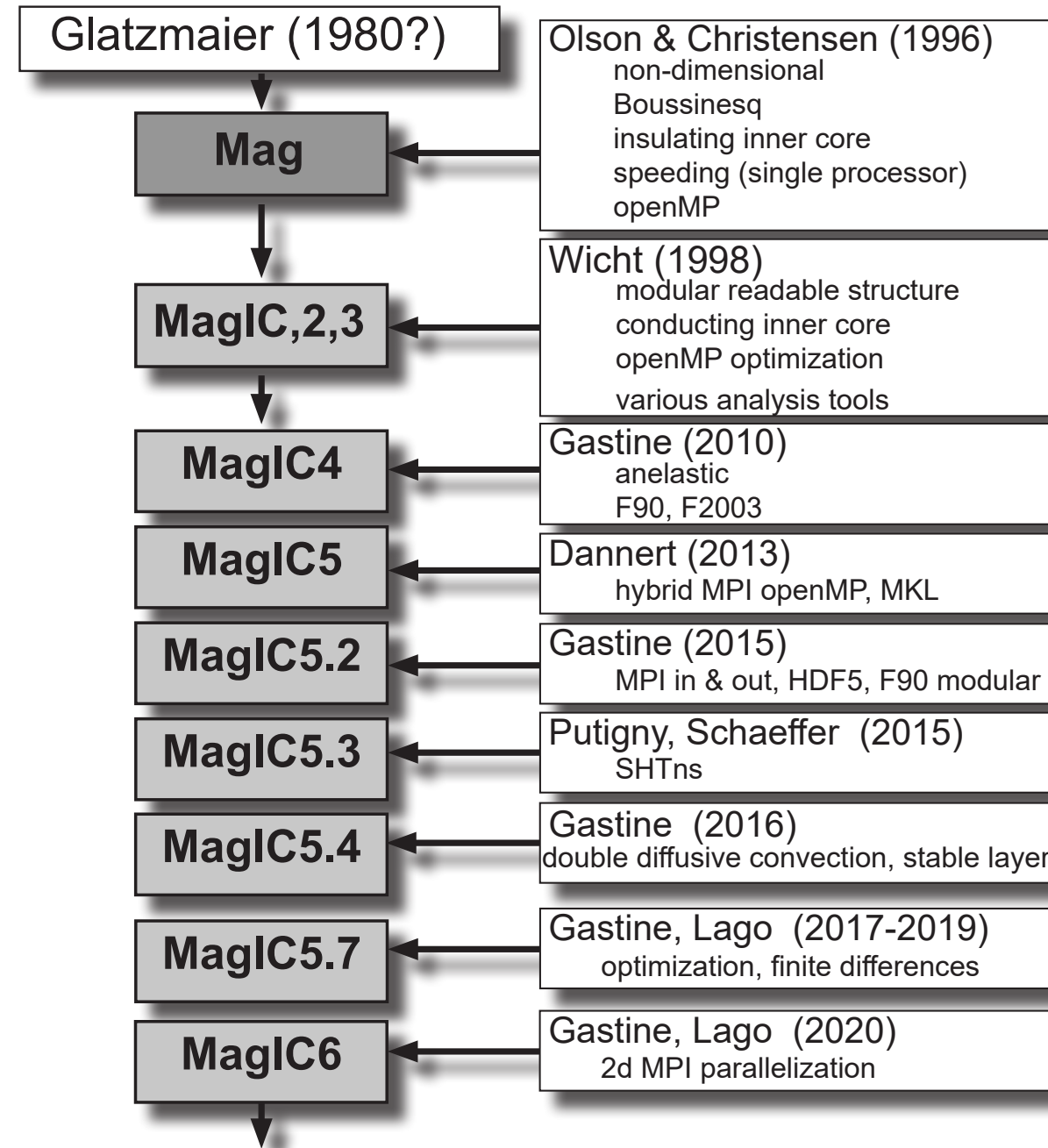
This class allows to read the 3-D graphic outputs of the MagIC code (G\_#.TAG and G\_ave.TAG) files. Those are binary unformatted outputs, there are therefore two ways to load them:

- If buildLib=True in magic.cfg and the fortran libraries were correctly built, then the reader uses a fortran program that is expected to be much faster than the pure python routine.
- If buildLib=False, then a pure python program is used to read the G files.

```
>>> # Regular G files
>>> gr = MagicGraph(ivar=1, tag='N0m2a')
>>> print(gr.vr.shape) # shape of vr
>>> print(gr.ek) # print ekman number
>>> print(gr.minc) # azimuthal symmetry
>>> # Averaged G file with double precision
>>> gr = MagicGraph(ave=True, tag='N0m2', precision=np.float64)
```

● Readily available visualization tools using Python.

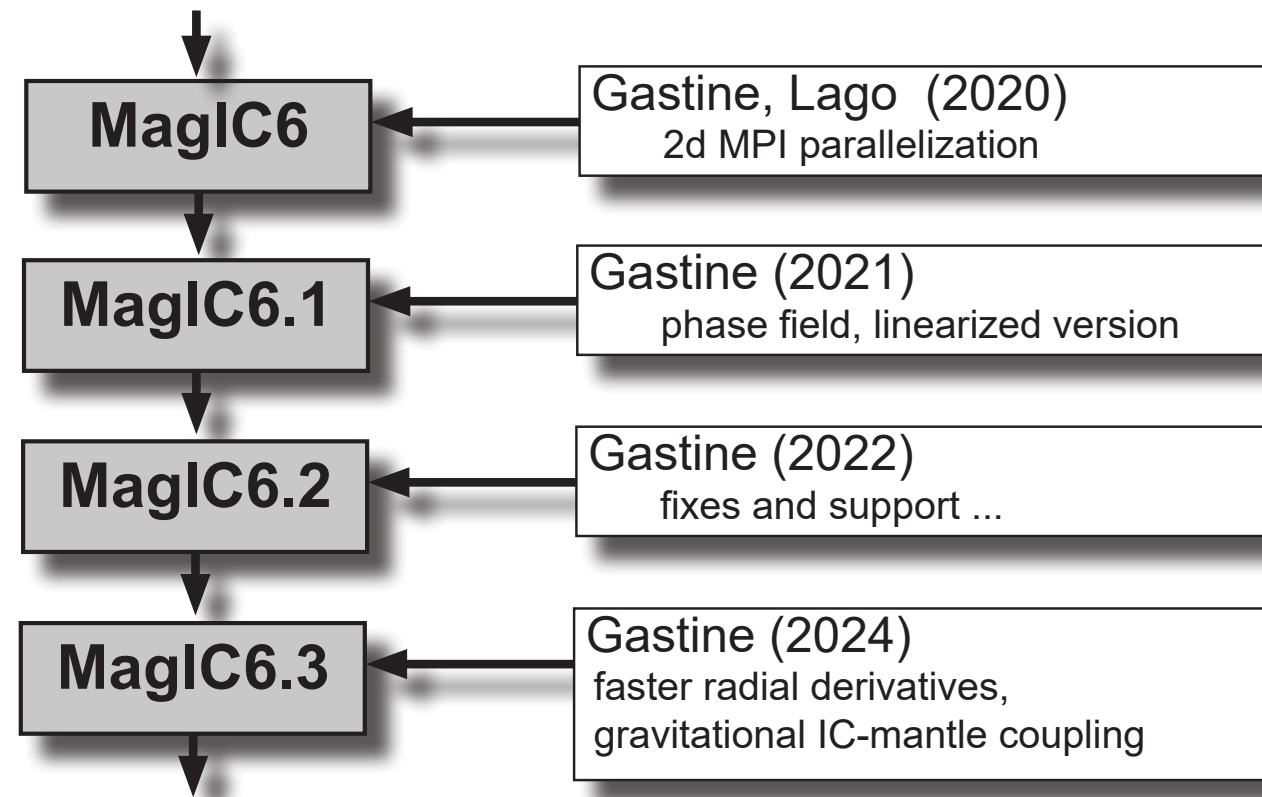
# MagIC Heritage



● Other codes: Ash, Rayleigh, Parody (JA), Xshells, UK-Mhd, Pencil



# MagIC Heritage



● We all rely on Thomas Gastine.

# MagIC in Words 1

- Open access
- Extensive manual
- Extensive post-processing including visualization
- High degree of parallelization
- Solves for convection and magnetic field generation
- Rotating frame of reference
- Spherical shell or full sphere
- Dimensionless formulation

# Poloidal/Toroidal Decomposition

- Poloidal/toroidal representation for flow and magnetic field:

$$\mathbf{U} = \nabla \times \nabla \times \hat{\mathbf{r}} w + \nabla \times \hat{\mathbf{r}} z$$

$$\mathbf{B} = \nabla \times \nabla \times \hat{\mathbf{r}} b + \nabla \times \hat{\mathbf{r}} j$$

- Poloidal potentials  $w$  and  $b$ , toroidal potentials  $z$  and  $j$
- Continuity equations are automatically fulfilled.

$$\nabla \cdot \mathbf{U} = 0 \quad \nabla \cdot \mathbf{B} = 0$$

- This procedure allows to go from 9 equations with 8 unknowns to 6 equations for 6 unknowns.

NOTE:

$$\mathbf{U} = -\nabla^2 w \hat{\mathbf{r}} + \nabla_H \frac{\partial}{\partial r} w + \nabla \times \hat{\mathbf{r}} z$$

Radial component is purely poloidal.

Horizontal poloidal component depends on radial derivative.

# Poloidal/Toroidal Equations

- Poloidal/toroidal equations:

$$\begin{aligned}\hat{\mathbf{r}} \cdot \frac{\partial}{\partial t} \mathbf{U} &= -\nabla_H^2 \frac{\partial}{\partial t} w \\ \hat{\mathbf{r}} \cdot \nabla \times \frac{\partial}{\partial t} \mathbf{U} &= -\nabla_H^2 \frac{\partial}{\partial t} z\end{aligned}$$

with horizontal Laplacian

$$\nabla_H^2 = \frac{1}{r^2 \sin \theta} \frac{\partial}{\partial \theta} \left( \sin \theta \frac{\partial}{\partial \theta} \right) + \frac{1}{r^2 \sin^2 \theta} \frac{\partial^2}{\partial \phi^2}$$

- Spherical surface harmonics are Eigen function:

$$\nabla_H^2 Y_{\ell m} = -\frac{\ell(\ell+1)}{r^2} Y_{\ell m}$$

- Pressure equation:

$$\nabla_H \frac{\partial}{\partial t} \mathbf{U} = \nabla_H^2 \frac{\partial}{\partial t} \frac{\partial}{\partial r} w$$

## 7 Equations for 7 Unknowns

- Radial component of NS equation for poloidal flow potential  $w$  .
- Radial component of the curl of the NS equation for toroidal flow potential  $z$  .
- Horizontal divergence of the NS equation for pressure  $p$ .
- Heat equation for temperature  $T$ .
- Radial component of induction equation for poloidal field potential  $b$  .
- Radial component of the curl of the induction equation for toroidal field potential  $j$  .
- Continuity equations automatically fulfilled!



# Boundary Conditions

- Flow:

either rigid boundaries with vanishing flow

$$w = \frac{\partial}{\partial r} w = z = 0$$

or stress-free

$$w = \left( \frac{\partial^2}{\partial r^2} - \frac{2}{r} \right) w = \left( \frac{\partial}{\partial r} - \frac{2}{r} \right) z = 0$$

- Magnetic field is matched to a potential field at the outer boundary and to a simplified solution for a rigid inner core in the inner boundary.

- Matching condition to a potential field at the CMB reads:

$$b_{\ell m} + \frac{r_o}{\ell} \frac{\partial}{\partial r} b_{\ell m} = 0$$

- Temperature boundary conditions are either fixed temperature or fixed heat flux (radial gradient of temperature).

# Pseudospectral Approach

- Derivatives are best calculated in spectral space (recurrence relations).
- The magnetic boundary condition is easily formulated in spectral space ( $Y_{lm}$ ).
- The  $Y_{lm}$  are Eigen functions of the Laplace operator (dissipation).
- Output wanted in physical grid.
- Non-linear terms are best calculated on physical grid.
- We thus need a spectral and a grid representations and transformations between the two.

# Numerical Grid

- Equidistant points in longitude for FFT
- Gauss-Legendre grid points in latitude for Gauss-Legendre transform
- Special grid points in radius to use FFT for Chebychev polynomials  $C_n$ :



# Horizontal Representation

- Spherical surface harmonics on longitude and colatitude:

$$Y_{\ell}^m(\theta, \phi) = P_{\ell}^m(\cos \theta) e^{im\phi}$$
$$\int_0^{2\pi} d\phi \int_0^{\pi} \sin \theta d\theta Y_{\ell}^m(\theta, \phi) Y_{\ell'}^{m'}(\theta, \phi) = \delta_{\ell\ell'} \delta^{mm'}$$

- Grid representation with degree and order up to  $\ell_{max}$

$$g(r, \theta, \phi) = \sum_{\ell=0}^{\ell_{max}} \sum_{m=-\ell}^{\ell} g_{\ell m}(r) Y_{\ell}^m(\theta, \phi)$$

- Spectral representation:

$$g_m(r, \theta) = \frac{1}{2\pi} \int_0^{2\pi} d\phi g(r, \theta, \phi) e^{-im\phi}.$$

$$g_{\ell m}(r) = \frac{1}{\pi} \int_0^{\pi} d\theta \sin \theta g_m(r, \theta) P_{\ell}^m(\cos \theta)$$

# Horizontal Transforms

- FFT for  $\phi \rightarrow m$  and  $m \rightarrow \phi$ , with  $N_\phi = 2 N_\theta$  grid points
- Gauss-Legendre integration for  $\theta \rightarrow \ell$

$$g_{\ell m}(r) = \frac{1}{N_\theta} \sum_{j=1}^{N_\theta} w_j g_m(r, \theta_j) P_\ell^m(\cos \theta_j)$$

- Clever vector multiplication for  $\ell \rightarrow \theta$
- Full dealiasing with  $\ell_{max} = [\min(2N_\theta, N_\phi) - 1]/3$
- Both transforms faster with SHTns



# Horizontal Derivatives

- Horizontal Laplacian:

$$\Delta_H Y_\ell^m = -\frac{\ell(\ell+1)}{r^2} Y_\ell^m$$

- $\theta$  derivative

$$\vartheta_2 = \sin \theta \frac{\partial}{\partial \theta}$$

$$\vartheta_2 P_\ell^m(\cos \theta) = \ell c_{\ell+1}^m P_{\ell+1}^m(\cos \theta) - (\ell+1) c_\ell^m P_{\ell-1}^m(\cos \theta)$$

with

$$c_\ell^m = \sqrt{\frac{(\ell+m)(\ell-m)}{(2\ell-1)(2\ell+1)}}$$

so that

$$\int_0^\pi d\theta \sin \theta P_\ell^m \vartheta_2 \sum_{\ell'} f_{\ell'}^m P_{\ell'}^m = (\ell-1) c_\ell^m f_{\ell-1}^m - (\ell+2) c_{\ell+1}^m f_{\ell+1}^m$$

# Radial Representation

- Chebychev polynomials up to degree N

$$g_{\ell m}(r) = \sum_{n=0}^N g_{\ell mn} \mathcal{C}_n(r)$$

$$\mathcal{C}_n(x) = \cos[n \arccos(x)] \quad -1 \leq x \leq 1$$

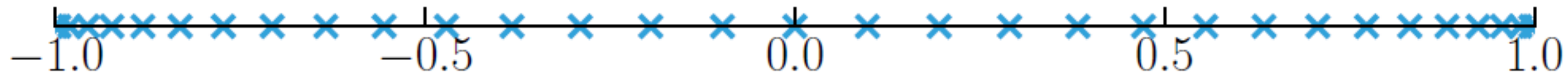
- Grid points are the N extrema of  $\mathcal{C}_{N_r-1}$

$$x_k = \cos\left(\pi \frac{(k-1)}{N_r-1}\right), \quad k = 1, 2, \dots, N_r$$

so that

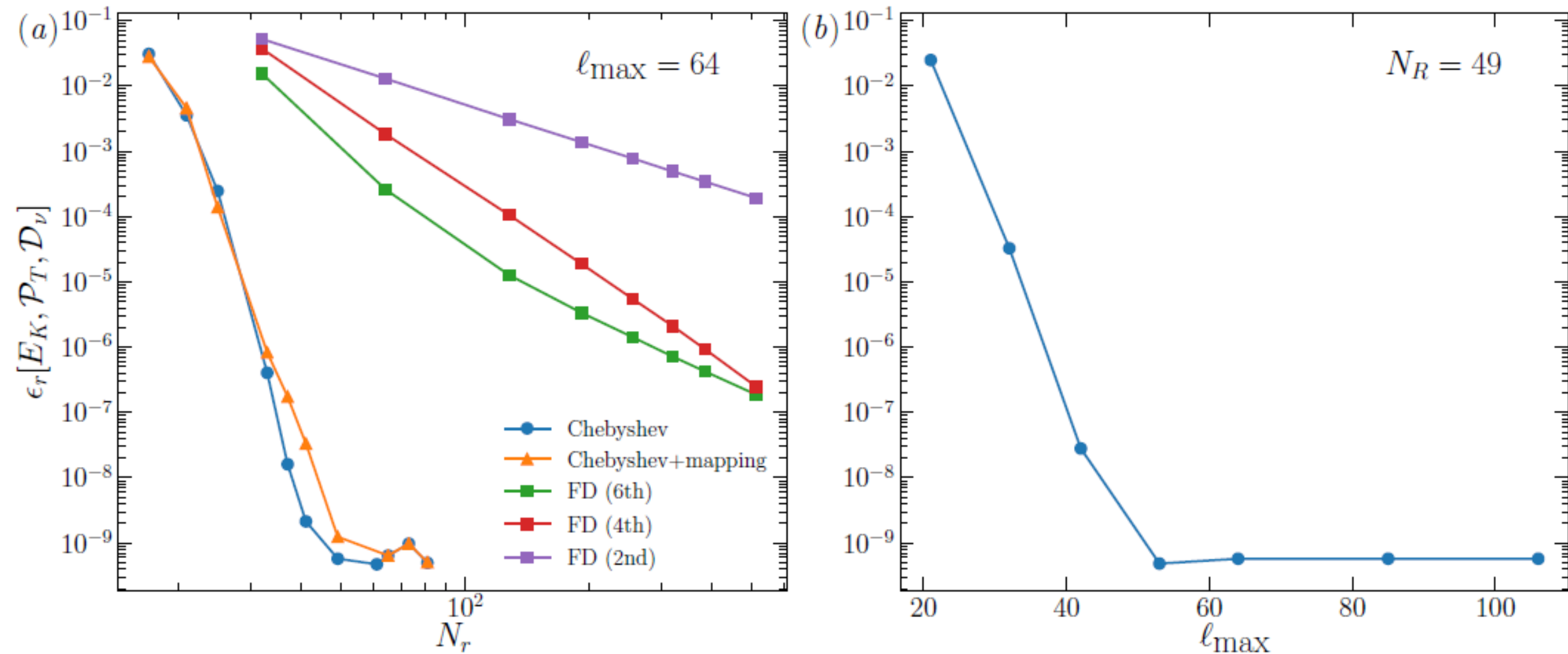
$$\mathcal{C}_{nk} = \mathcal{C}_n(x_k) = \cos\left(\pi \frac{n(k-1)}{N_r-1}\right)$$

- This allows us to use FFT in radius.
- Derivatives are calculated with recurrence relations.
- Finite difference schemes are also available!



# Collocation versus Finite Differences

A comparison for the benchmark BM0 (Christensen *et al.* 2001)



# Calculating the Non-Linear Term

- Calculate horizontal components of EMF  $\mathcal{F} = \mathbf{u} \times \mathbf{B}$  on grid

$$\mathcal{F}_\theta = u_\phi B_r - u_r B_\phi, \quad \mathcal{F}_\phi = u_r B_\theta - u_\theta B_r$$

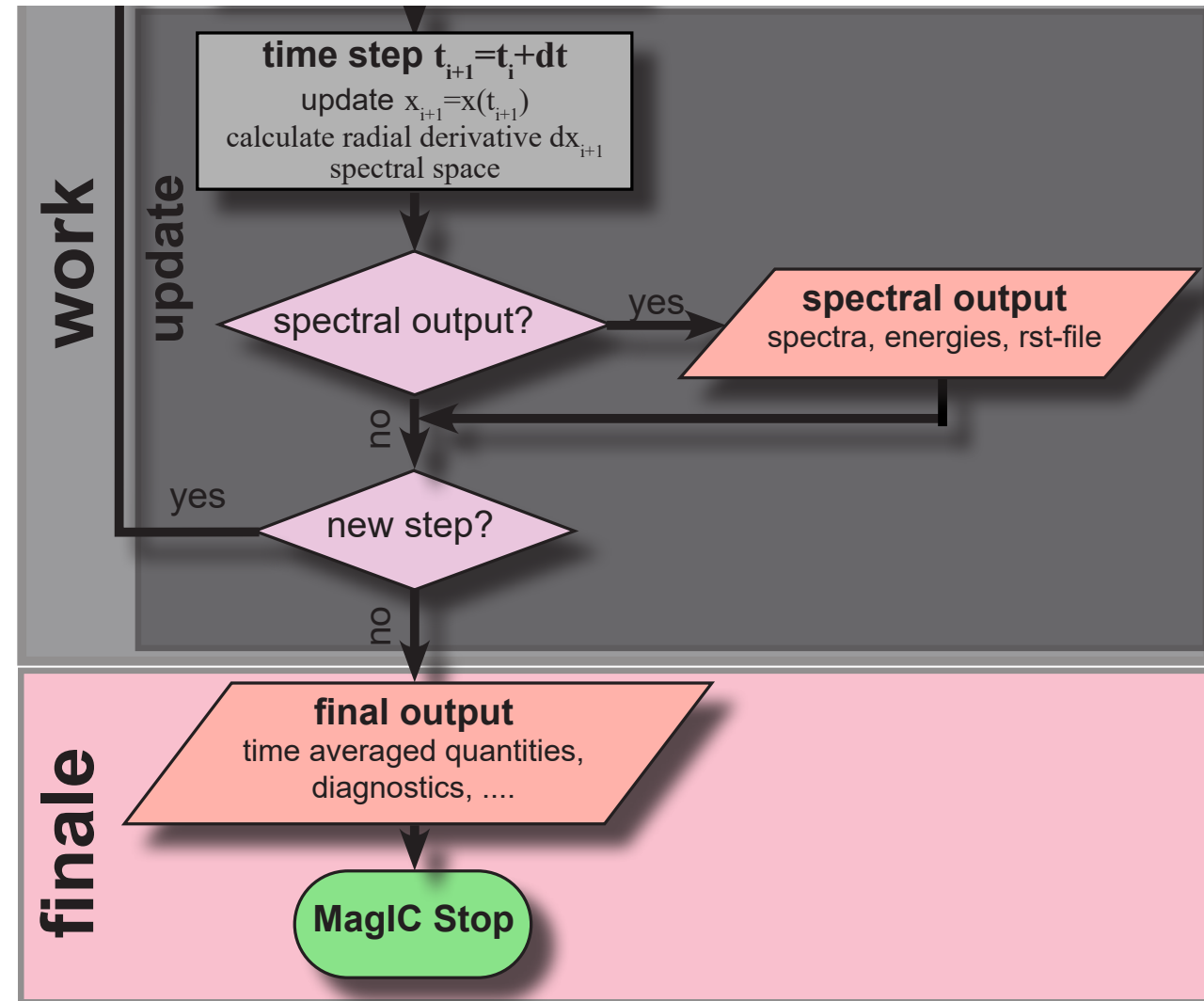
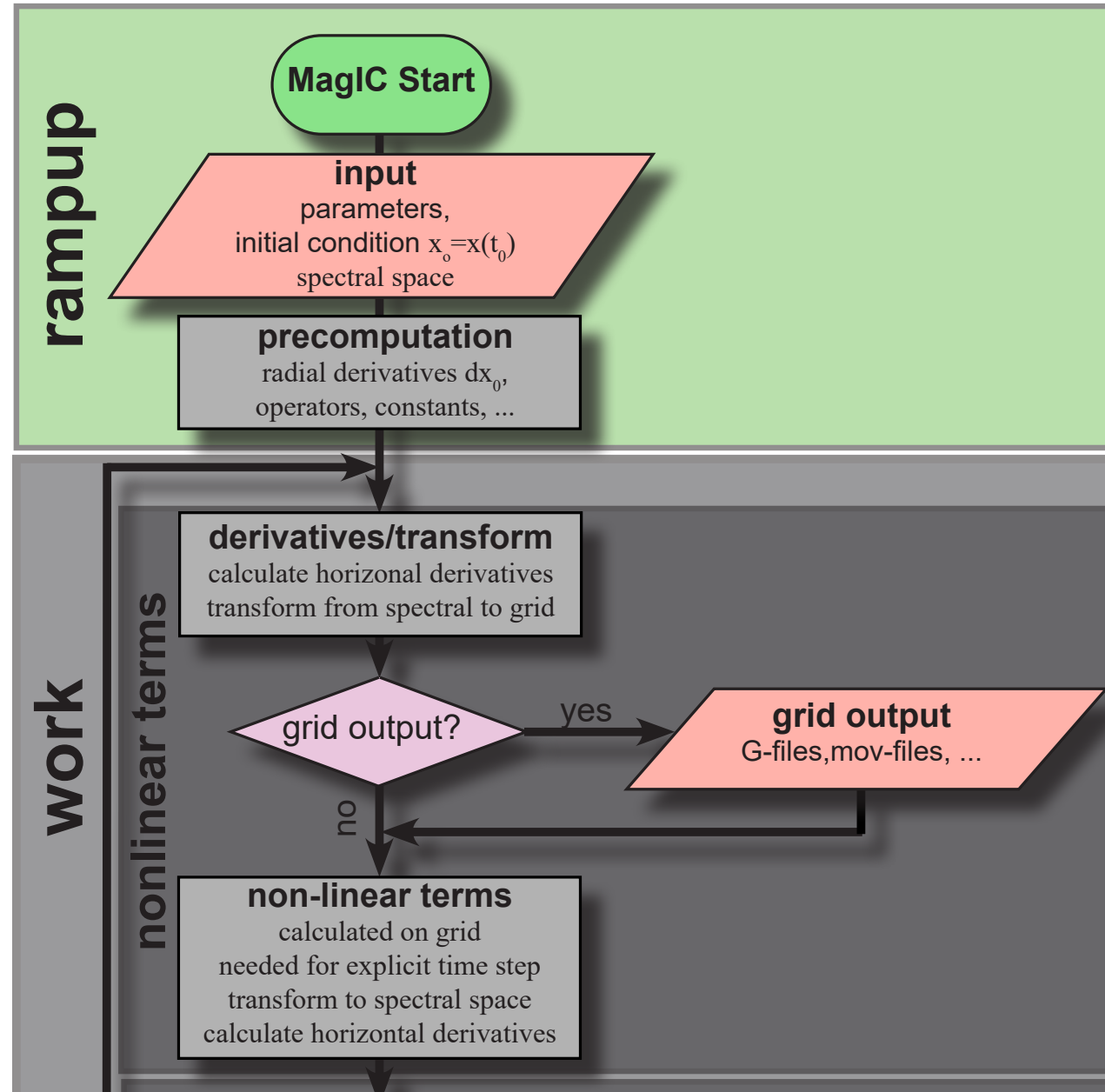
- Then 
$$\mathcal{N}^g = \mathbf{e}_r \cdot [\nabla \times (\mathbf{u} \times \mathbf{B})] = \frac{1}{r \sin \theta} \left[ \frac{\partial (\sin \theta \mathcal{F}_\phi)}{\partial \theta} - \frac{\partial \mathcal{F}_\theta}{\partial \phi} \right]$$

- Transform components to spectral space:  $\mathcal{F}_{\theta\ell}^m \quad \mathcal{F}_{\phi\ell}^m$

- Then calculate derivative in spectral space:

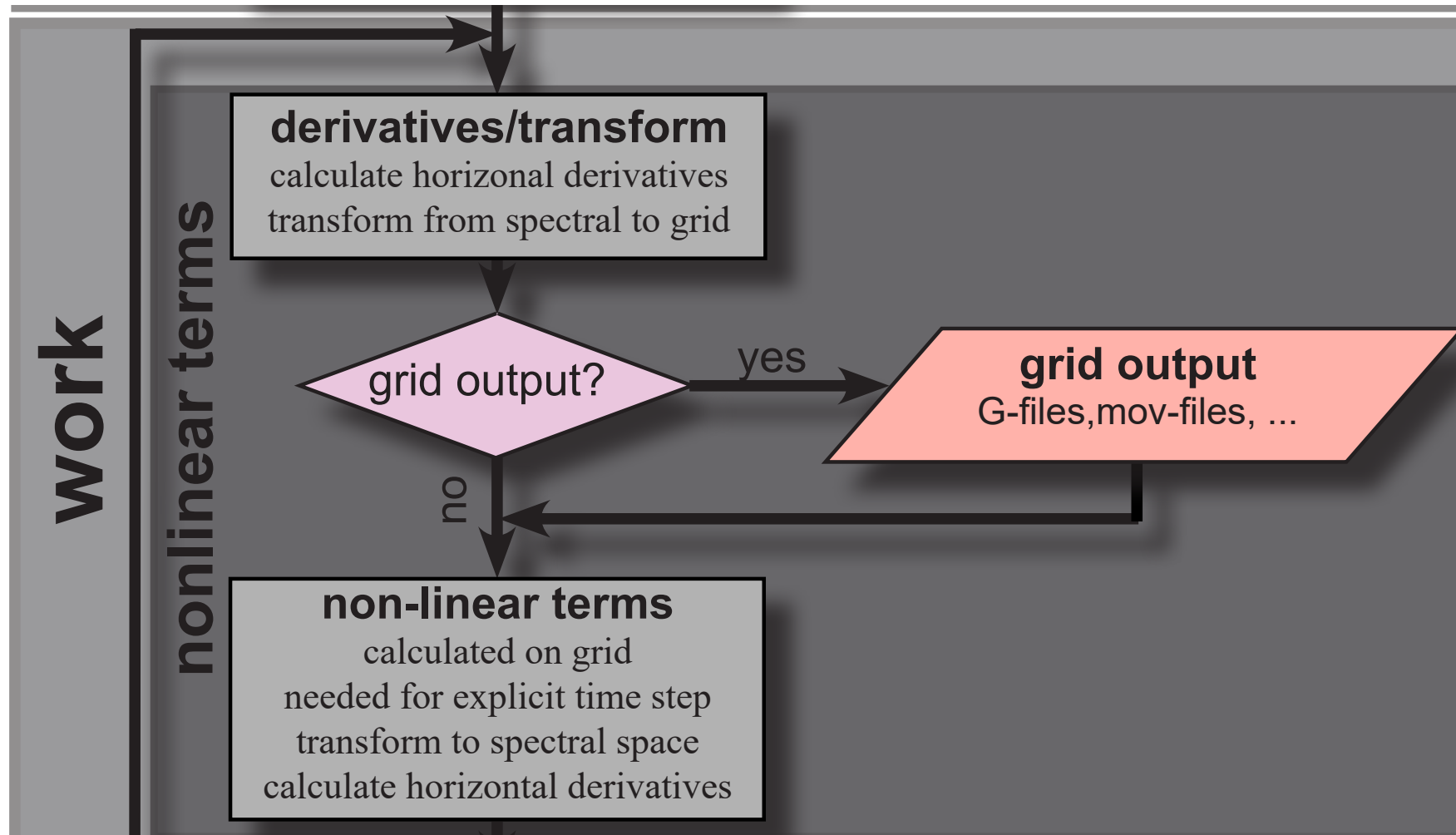
$$\mathcal{N}_{\ell m}^g = (\ell + 1) c_\ell^m \mathcal{F}_{\phi\ell-1}^m - \ell c_{\ell+1}^m \mathcal{F}_{\phi\ell+1}^m - im \mathcal{F}_{\theta\ell}^m$$

# MagIC Structure





# Radial Loop



$$b(r, \ell, m)$$



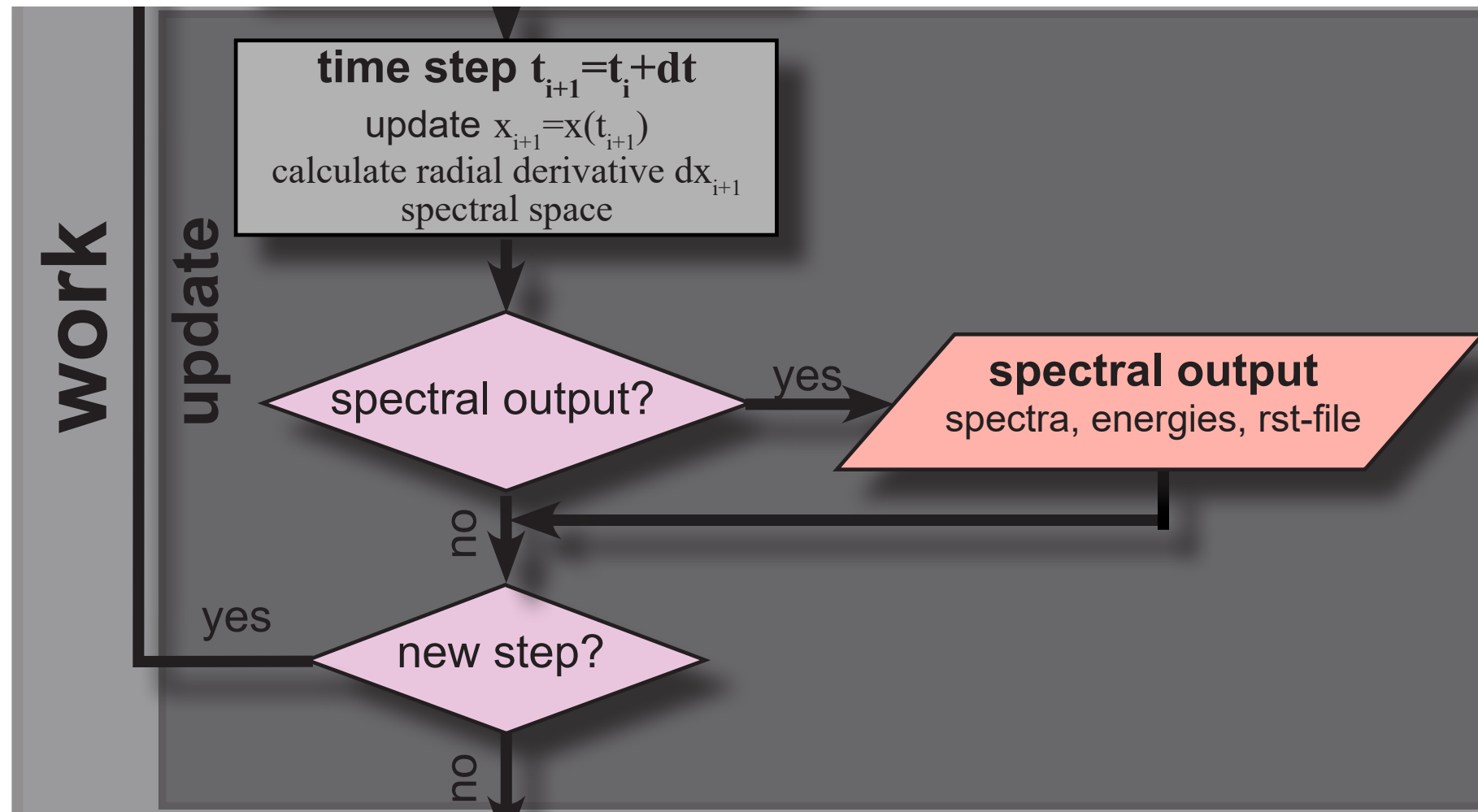
$$b(r, \theta, \phi)$$



$$b(r, \ell, m)$$

- Solving the non-linear terms takes most of the time. They are calculated in grid space and then transformed to spectral space.

# LM Loop



$$b(t, r, \ell_i, m_i) \rightarrow b(t + \delta t, r, \ell_i, m_i)$$

- The time step is performed for each spherical harmonic mode, solving a linear system of equations, one equation per radial grid point. Boundary equations are replaced by the boundary conditions.

# Time Discretization

- Generic evolution equation with:

terms  $\mathcal{I}(x, t)$  to be treated implicitly and  
terms  $\mathcal{E}(x, t)$  to be treated explicitly

$$\frac{\partial}{\partial t} x + \mathcal{I}(x, t) = \mathcal{E}(x, t)$$

- Implicit Crank-Nicolson time step:

$$\left( \frac{x(t + \delta t) - x(t)}{\delta t} \right)_I = -\alpha \mathcal{I}(x, t + \delta t) - (1 - \alpha) \mathcal{I}(x, t)$$

- Explicit Adams-Bashforth time step:

$$\left( \frac{x(t + \delta t) - x(t)}{\delta t} \right)_E = \frac{3}{2} \mathcal{E}(x, t) - \frac{1}{2} \mathcal{E}(x, t - \delta t)$$

- Combined mixed time step yields algebraic equation

$$\frac{x(t + \delta t)}{\delta t} + \alpha \mathcal{I}(x, t + \delta t) = \frac{x(t)}{\delta t} - (1 - \alpha) \mathcal{I}(x, t) + \frac{3}{2} \mathcal{E}(x, t) - \frac{1}{2} \mathcal{E}(x, t - \delta t)$$

- Time step is checked with a modified Courant-Friedrich-Levy criterion.

- Note: Several other higher order methods have been implemented.

# Example Algebraic Equation for Poloidal Magnetic Field

$$(\mathcal{A}_{kn} + \alpha \mathcal{I}_{kn}) g_{\ell mn}(t + \delta t) = (\mathcal{A}_{kn} - (1 - \alpha) \mathcal{I}_{kn}) g_{\ell mn}(t) + \frac{3}{2} \mathcal{E}_{k\ell m}(t) - \frac{1}{2} \mathcal{E}_{k\ell m}(t - \delta t)$$

with

$$\mathcal{A}_{kn} = \frac{\ell(\ell + 1)}{r_k^2} \frac{1}{\delta t} \mathcal{C}_{nk}$$

$$\mathcal{I}_{kn} = \frac{\ell(\ell + 1)}{r_k^2} \frac{1}{Pm} \left( \frac{\ell(\ell + 1)}{r_k^2} \mathcal{C}_{nk} - \mathcal{C}_{nk}'' \right)$$

$$\mathcal{E}_{k\ell m}(t) = \mathcal{N}_{k\ell m}^g = \int d\Omega Y_\ell^{m*} \mathbf{e}_r \cdot \mathbf{D}(t, r_k, \theta, \phi)$$

- For each spectral mode we solve a system of algebraic linear equations, one equation for each radial grid point.
- Equations for outer and inner radial grid points are replaced with boundary conditions.

# MagIC in Words 2

- Open access
- Extensive manual
- Extensive post-processing including visualization
- High degree of parallelization
- Solves for convection and magnetic field generation
- Rotating frame of reference
- Spherical shell or full sphere
- Dimensionless formulation
- Poloidal/toroidal decomposition
- Pseudo-spectral approach
- Choice of pseudospectral or finite differences in radius
- Choice of different time stepping schemes
- Automatic check of adapted Courant-Friedrich-Levy criterium



# Spectral Poloidal Dynamo Equation

- Radial component of induction equation

$$\mathbf{e}_r \cdot \left( \frac{\partial \mathbf{B}}{\partial t} \right) = \frac{\partial}{\partial t} (\mathbf{e}_r \cdot \mathbf{B}) = -\Delta_H \frac{\partial g}{\partial t}$$

- Plug in spectral representation of  $g$ , multiply with spherical harmonic, integrate over spherical surface, use orthogonality relations.  
This yields the spectral form:

$$\begin{aligned} \frac{\ell(\ell+1)}{r^2} \left[ \left( \frac{\partial}{\partial t} + \frac{1}{Pm} \lambda \frac{\ell(\ell+1)}{r^2} \right) C_n - \frac{1}{Pm} \lambda C_n'' \right] g_{\ell mn} \\ = \mathcal{N}_{\ell m}^g = \int d\Omega Y_\ell^{m*} \mathcal{N}^g = \int d\Omega Y_\ell^{m*} \mathbf{e}_r \cdot \mathbf{D} \end{aligned}$$

- $\mathbf{D}$  is the radial induction term calculated in grid space

# Final Spectral Equations

- This yields a purely spectral equation:

$$\frac{\ell(\ell + 1)}{r^2} \left[ \left( \frac{\partial}{\partial t} + \frac{1}{Pm} \lambda \frac{\ell(\ell + 1)}{r^2} \right) c_n - \frac{1}{Pm} \lambda c_n'' \right] g_{\ell mn} = \mathcal{N}_{\ell m}^g$$

- Treating the other equations similarly means that all spatial derivatives are taken care off!
- What do we do with the time derivative?