



The Spherical MHD Code MagIC, Fundamentals

Johannes Wicht, Thomas Gasting, Ankit Barik
MPS Katlenburg-Lindau

MagIC in Words

- MagIC solves for thermodynamic evolution, fluid flow and magnet field generation
- Domain = spherical shell
- Region below and above domain treated as boundary conditions or parametrized
- Frame of reference rotating with system rotation Ω
- MagIC uses a dimensionless formulation
- Poloidal/toroidal decomposition in employed
- MagIC is a pseudo spectral code
- MagIC uses a mixed impicite, explicite time stepping

MagIC github

This repository Search Pull requests Issues Gist

magic-sph / magic Unwatch 8 Star 4 Fork 0

MagIC is a high-performance code that solves the magneto-hydrodynamics equations in rotating spherical shells
<https://magic-sph.github.io/>

323 commits 4 branches 3 releases 6 contributors

Branch: master magic / +

tgastine fix g file header when WITH_MPI=False Latest commit c9de562 18 hours ago

bin	Auto-detect GWDG and set ccompiler for f2py	a month ago
cmake	make cmake backward compatible for 2.6	22 hours ago
doc	replace USE_MKL by USE_LAPACKLIB, add lapack	22 hours ago
license	- merge the python subroutines into the MPI version (latest version)	2 months ago
paraview	Create a new branch for temporarily store the mpi version. Will be	2 years ago
python/magic	Fixed butterfly.py file. It was actually taking the values at some lo...	22 hours ago
samples	fix USE_OMP flag in auto-test script	6 days ago
src	fix g file header when WITH_MPI=False	18 hours ago
subscripts	NEW hybrid MPI/OpenMP version.	2 years ago
.gitignore	mv magic.cfg to magic.cfg.default	a month ago
CMakeLists.txt	make cmake backward compatible for 2.6	22 hours ago
README.md	fix architecture + improve magic_checks.pl	7 days ago

Code

Issues 1

Pull requests 0

Wiki

Pulse

Graphs

HTTPS clone URL

<https://github.cc>

You can clone with HTTPS, SSH, or Subversion.

Clone in Desktop

Download ZIP

MagIC homepage



[Home](#) [Get it/Run it](#) [Contribute!](#) [Numerical methods](#) [Contents](#)

Table Of Contents

Welcome

- [Quickly starting using MagIC](#)
- [Documentation](#)
- [Contributing to the code](#)
- [Giving credit](#)

This Page

[Show Source](#)

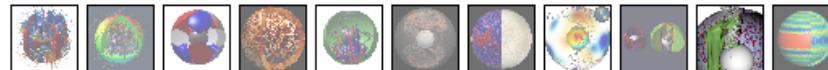
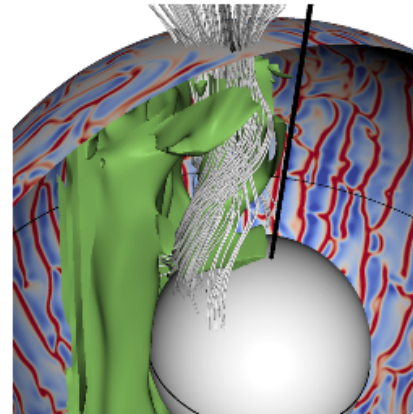
Quick search

Enter search terms or a module, class or function name.

Welcome

Formation of polar spots in a fully-convective star model

Yadav, R. et al., A&A, 2015



MagIC is a numerical code that can simulate fluid dynamics in a spherical shell. MagIC solves for the Navier-Stokes equation including Coriolis force, optionally coupled with an induction equation for Magneto-Hydro Dynamics (MHD) and a temperature (or entropy) equation under both the anelastic and the Boussinesq approximations.

MagIC uses Chebyshev polynomials in the radial direction and spherical harmonic decomposition in the azimuthal and latitudinal directions. The time-stepping scheme relies on a semi-implicit Crank-Nicolson for the linear terms of the MHD equations and a Adams-Bashforth scheme for the non-linear terms and the Coriolis force.

MagIC is written in Fortran and designed to be used on supercomputing clusters. It thus relies on a hybrid

Spectral and Grid Representation

[Home](#)[Get it/Run it](#)[Contribute!](#)[Numerical methods](#)[Contents](#)[Magic 5.2 documentation »](#)[next](#) | [modules](#) | [fortran modules](#) | [index](#)

Table Of Contents

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

Next topic

[Introduction](#)

This Page

[Show Source](#)

Quick search

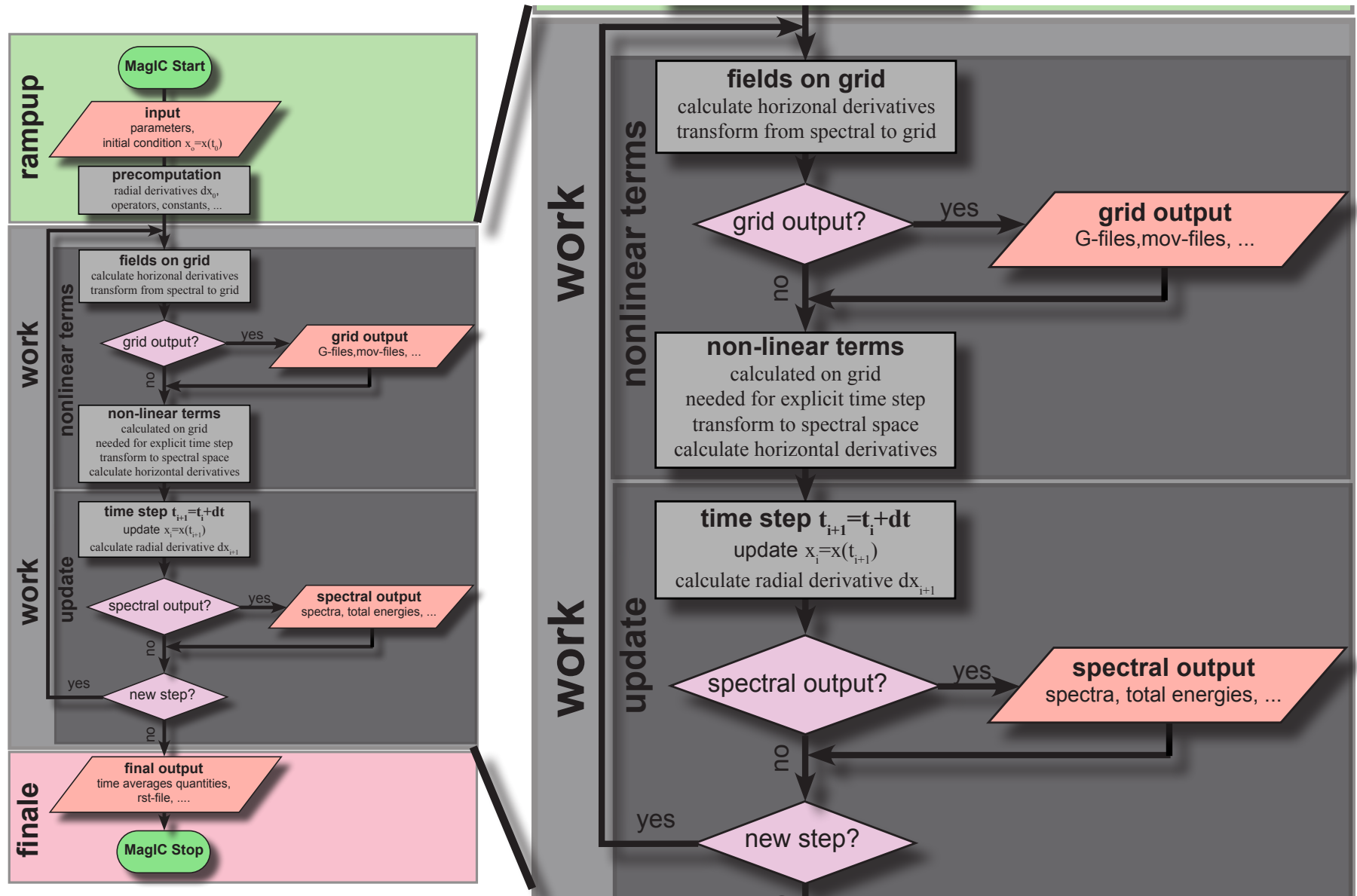
Enter search terms or a module,
class or function name.

Contents

- [Get MagIC and run it](#)
 - [Download the code](#)
 - [Setting up the environment variables](#)
 - [Setting up compiler options and compiling](#)
 - [Preparing a production run](#)
- [Formulation of the \(magneto\)-hydrodynamics problem](#)
 - [The anelastic approximation](#)
 - [An adiabatic reference state](#)
 - [MHD equations](#)
 - [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](#)

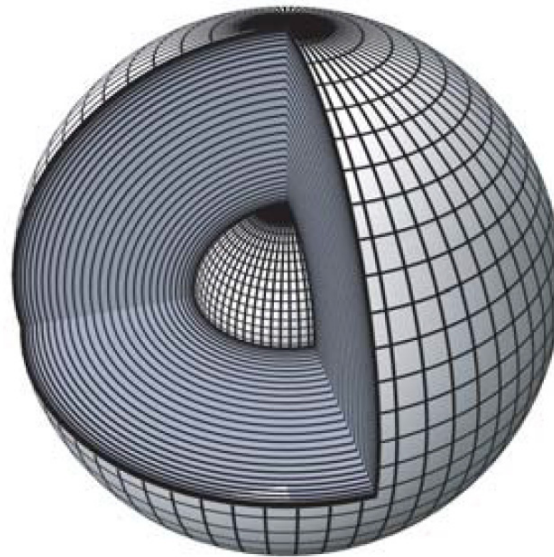
[Checking the consistency of the code](#)

MagIC Structure



Numerical Grid

- Non-linear terms calculated on local grid



- Gauss-Legendre grid points in latitude
- evenly spaced grid points in longitude for FFT
- special grid points in radius for FFT

$$C_{nk} = C_n(x_k) = \cos \left(\pi \frac{n(k-1)}{N_r - 1} \right)$$

Horizontal Representation

- Spherical surface harmonics in longitude and latitude:

$$Y_\ell^m(\theta, \phi) = P_\ell^m(\cos \theta) e^{im\phi}$$

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

- Grid representation:

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

With degree and order up to ℓ_{max}

- Spectral representation:

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

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

Horizontal Transforms

1) FFT for $\phi \rightarrow m$ and $m \rightarrow \phi$.
with $N_\phi = 2N_\theta$ azimuthal grid points

2) Gauss-Legendre integration for $\theta \rightarrow \ell$.

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

with N_θ azimuthal grid points

Clever vector multiplication $\ell \rightarrow \theta$.

Recently both azimuthal transforms changed to SHTns.

3) Full dealiasing with $\ell_{max} = [\min(2N_\theta, N_\phi) - 1]/3$

Horizontal Derivatives, two Examples

1) Horizontal Laplacian:

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

2) Latitudinal 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

1) Chebychev polynomials up to degree

$$g_{lm}(r) = \sum_{n=0}^N g_{lmn} \mathcal{C}_n(r)$$

with

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

2) Grid points are the N_r 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

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

3) Thus fast cosine transforms can be

4) Dealiasing for $N_r > N$.

Spectral Poloidal Dynamo Equation

1) Take radial component of the dynamo equation

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

2) Plug in horizontal spectral representation of g . Multiply with spherical harmonic, integrate over spherical surface and use orthogonal relations:

radial diffusion

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

Solving the Nonlinear Term

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

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

so that

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

2) Transform these to spectral space: $\mathcal{F}_{\theta\ell}^m \quad \mathcal{F}_{\phi\ell}^m$

3) Calculate curl in spectral space using recurrence relations:

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

How does that look like in the code?

Calculating Nonlinear terms

0) Master: rIterThetaBlocking_seq.f90

1) transform_to_grid_space in
rIterThetaBlocking.f90

2) Legendre transform:

legTFG in legendre_spec_to_grid.f90

```
do mc=1,n_m_max
  dm =D_mc2m(mc)
  lmS=lStop(mc)
  vrES =zero
  vrEA =zero
  dvrdtES=zero
  dvrdtEA=zero
  cbrES =zero
  cbrEA =zero
  !--- 8 add/mult, 29 dble words
  do lm=lStart(mc),lmS-1,2
    vrES =vrES + leg_helper%dLhw(lm)* Plm(lm,nThetaNHS)
    dvrdtEA =dvrdtEA + leg_helper%dLhw(lm)* dPlm(lm,nThetaNHS)
```

3) FFT (internal or MKL) called in rIterThetaBlocking.f90

```
if ( this%nBc == 0 ) then
  call fft_thetab(gsa%vrc,1)
  call fft_thetab(gsa%vrc,1)
```

Calculating Nonlinear terms

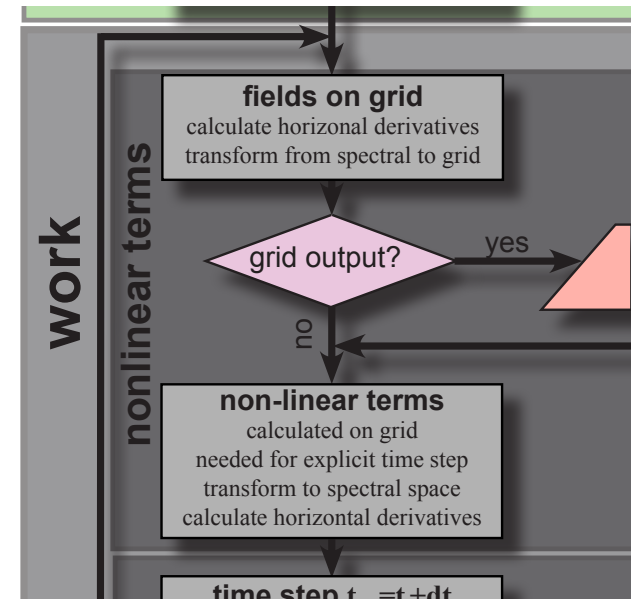
4) Calculate nonlinear terms on grid:

get_n1.f90

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

```
nTheta=nThetaLast
do nThetaB=1,sizeThetaB
  nTheta =nTheta+1
  nThetaNHS=(nTheta+1)/2|
  or4sn2=or4(nR)*osn2(nThetaNHS)

do nPhi=1,n_phi_max
  this%VxBt(nPhi,nThetaB)= orho1(nR)*or4sn2 * (
    this%vpc(nPhi,nThetaB)*this%brc(nPhi,nThetaB) - &
    this%vrc(nPhi,nThetaB)*this%bpc(nPhi,nThetaB) )
end do
```



Back Transform to Spectral Space

5) `transfrom_to_lm_space` in
`rIterThetaBlocking.f90`

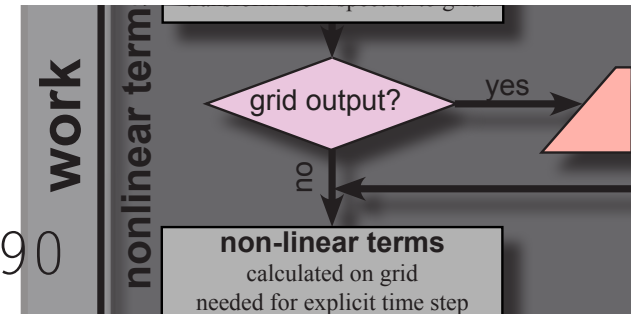
6) Gauss-Legendre back transform:
`legTFG` in `legendre_spec_to_grid.f90`

```
do nThetaB1=nThetaMin, sizeThetaB/2, 2
  nThetaB2=nThetaB1+1
  nTheta1 =nTheta1+2
  nTheta2 =nTheta1+1

do mc=1, n_m_max
  lmS=lStopP(mc)
  f1ES1=f1ES(mc, nThetaB1)
  f1ES2=f1ES(mc, nThetaB2)
  f1EA1=f1EA(mc, nThetaB1)
  f1EA2=f1EA(mc, nThetaB2)
  do lm=lStartP(mc), lmS-1, 2
    f1LM(lm) =f1LM(lm) +f1ES1*wPlm(lm, nTheta1)+f1ES2*wPlm(lm, nTheta2)
    f1LM(lm+1)=f1LM(lm+1)+f1EA1*wPlm(lm+1, nTheta1)+f1EA2*wPlm(lm+1, nTheta2)
  end do
```

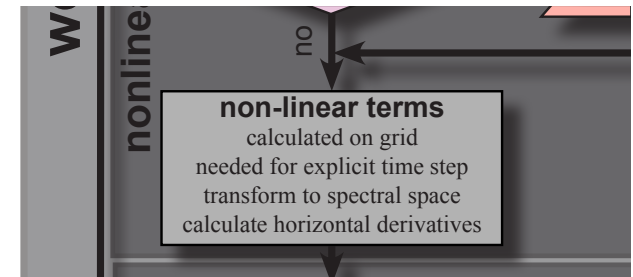
7) FFT back transform (internal or MKL) called in
`rIterThetaBlocking.f90`

```
call fft_thetab(gsa%VxBt, -1)
call fft_thetab(gsa%VxBp, -1)
```



Finish off Induction Term in Spectral Space

8) Calculate extra derivatives in
get_td.F90 called by
rIterThetaBlocking.f90



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

```
do lm=1, lm max
  if ( l > m ) then
    dbdt(lm) = dTheta1S(lm) * this%VxBpLM(lmPS) &
              & -dTheta1A(lm) * this%VxBpLM(lmPA) &
              & -dPhi(lm) * this%VxBtLM(lmP)
  else if ( l == m ) then
    dbdt(lm) = -dTheta1A(lm) * this%VxBpLM(lmPA) &
              & -dPhi(lm) * this%VxBtLM(lmP)
  end if
```

9) Output is called `dbdt(k, lm)` where `k` numbers the radial grid points and `lm` the spherical harmonic modes.

Everything in Spherical Harmonic Space!

Equations for each radial grid point and spherical harmonic mode!
No spatial derivatives left!

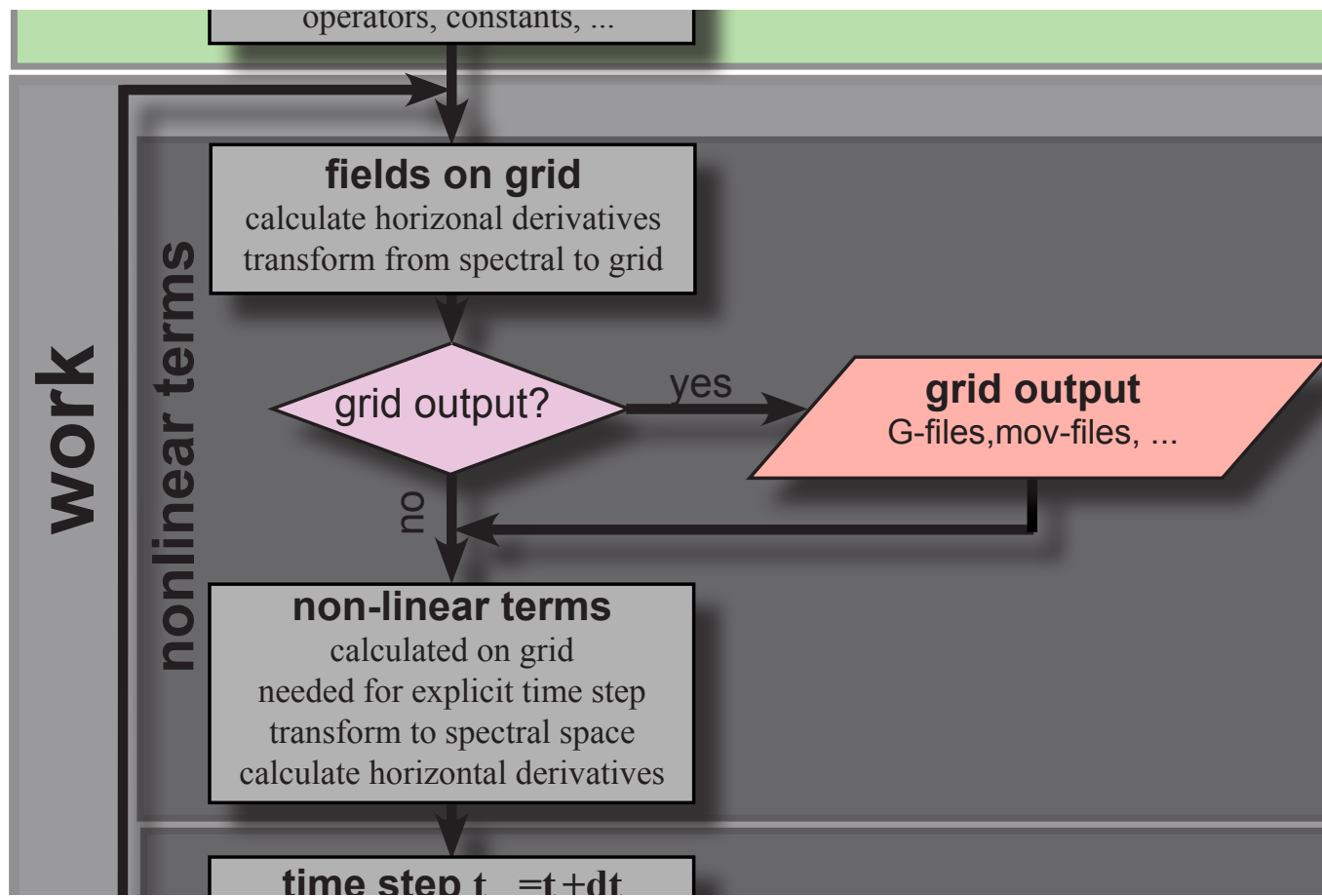
$$\frac{\ell(\ell + 1)}{r^2} \left[\left(\frac{\partial}{\partial t} + \frac{1}{Pm} \lambda \frac{\ell(\ell + 1)}{r^2} \right) C_n - \frac{1}{Pm} \lambda C_n'' \right] g_{lmn}$$
$$= \mathcal{N}_{\ell m}^g = \int d\Omega Y_\ell^{m*} \mathcal{N}^g = \int d\Omega Y_\ell^{m*} \mathbf{e}_r \cdot \mathbf{D}$$

nonlinear dynamo term

How to deal with the time integration, i.e. discretization in time?

MagIC Structure

The most time consuming part of the code!



Time Stepping Scheme

Generic evolutions equation with
terms $\mathcal{I}(x, t)$ to be treated implicitly and
terms $\mathcal{E}(x, t)$ to be treated explicitly.

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

Implicite Crank-Nicolson type time step:

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

Explicite Adams-Bashforth type time step:

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

Mixed time step:

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

Poloidal Magnetic Potential Time Stepping

$$(\mathcal{A}_{kn} + \alpha \mathcal{I}_{kn}) g_{lmn}(t + \delta t) = (\mathcal{A}_{kn} - (1 - \alpha) \mathcal{I}_{kn}) g_{lmn}(t) + \frac{3}{2} \mathcal{E}_{klm}(t) - \frac{1}{2} \mathcal{E}_{klm}(t - \delta t)$$

with

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

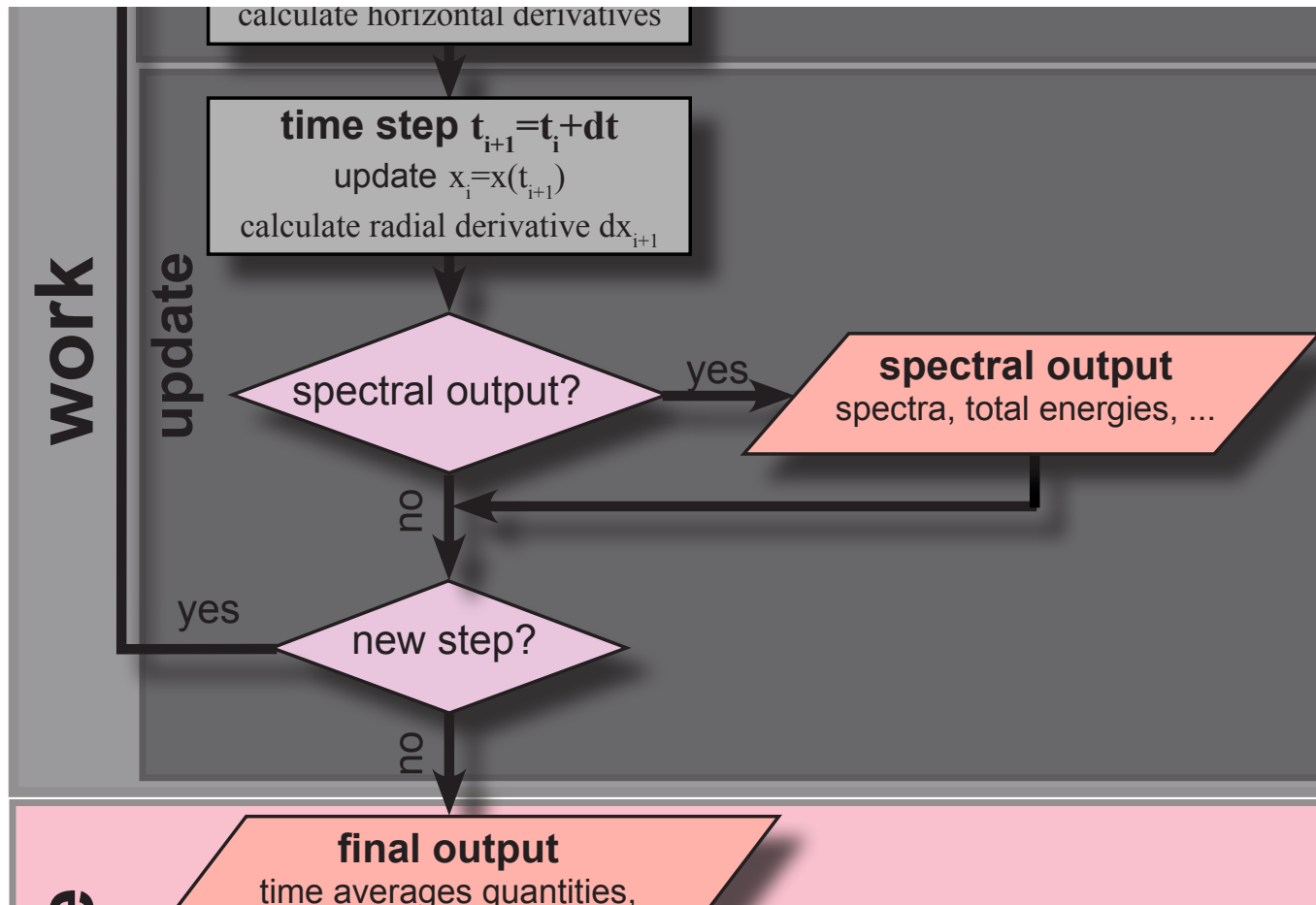
$$\mathcal{I}_{kn} = \frac{\ell(\ell + 1)}{r_k^2} \frac{1}{Pm} \left(\frac{\ell(\ell + 1)}{r_k^2} \mathcal{C}_{nk} - \mathcal{C}_{nk}'' \right)$$

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

How does that look like in the code?

MagIC Structure

The most important part of the code!



Poloidal Magnetic Potential Time Stepping

We already know how the nonlinear terms is calculated

$$\text{dbdt}(k, lm) = \mathcal{E}_{klm}$$

The time stepping matrix is

$$\text{bmat}(k, n, l) = A_{kn} + \alpha \mathcal{I}_{kn}$$

Information from previous time steps:

$$\text{dbdtLast}(k, lm) = -\frac{1}{2} \mathcal{E}_{klm}(t - \delta t) - (1 - \alpha) \mathcal{I}_{kn} g_{lmn}(t)$$

Also used are the following expressions

$$w1 = -1/2 \frac{\delta t}{\delta t_{old}} , \quad w2 = 1 - w1$$

$$Odt = 1/\delta t , \quad dLh(l) = l(l + 1) , \quad Or2(k) = 1/r_k^2$$

Update Routine

Putting it all together:

$$\begin{aligned} \text{bmat}(k, n, \ell) \star \text{b}_{i+1}(n, \ell m) &= w1 \star \text{dbdt}(k, \ell m) + w2 \star \text{dbdtLast}(k, \ell m) + \\ &\quad \text{Odt} \star \text{dLh}(\ell) \star \text{Or2}(k) \star \text{b}_i(n, \ell m) \\ &= \text{rhs}(k, \ell m) \end{aligned}$$

The time stepping matrix is stored in an LU-decomposed form.

All done in module `updateB.f90`:

1) Defining the rhs:

```
do nR=2,n_r_max-1
  if ( nR<=n_r_LCR ) then
    rhs1(nR,lmB,threadid)=0.0_cp
    rhs2(nR,lmB,threadid)=0.0_cp
  else
    rhs1(nR,lmB,threadid)= ( w1*dbdt(lm1,nR)+w2*dbdtLast(lm1,nR) )
      & + O_dt*dLh(st_map%lm2(l1,m1))*or2(nR)*b(lm1,nR)
    rhs2(nR,lmB,threadid)= ( w1*djdt(lm1,nR)+w2*djdtLast(lm1,nR) )
      & + O_dt*dLh(st_map%lm2(l1,m1))*or2(nR)*aj(lm1,nR)
  end if
end do
```


Update Routine

2) Solving the linear equation system:

```
call cgeslML(bMat(:, :, l1), n_r_tot, n_r_real, &
            bPivot(:, l1), rhs1(:, lmB0+1:lmB, threadid), 2*n_r_max, lmB-lmB0)
```

3) Result is in Cheb-space!

4) Calculate and store radial derivatives in Cheb-space:

```
call get_ddr(b, db, ddb, ulmMag-llmMag+1, start_lm-llmMag+1, &
            & stop_lm-llmMag+1, n_r_max, n_cheb_max, &
            & dbdtLast, djdtLast, chebt_oc, drx, ddrx)
```

5) Calculate and store $dbdtLast(k, \ell m)$ for next time step:

```
do nR=n_r_cmb, n_r_icb-1
  do lm1=lmStart_00, lmStop
    l1=lm2l(lm1)
    m1=lm2m(lm1)
    dbdtLast(lm1, nR) = dbdt(lm1, nR) -
                      coex*opm*lambda(nR)*hdif_B(st_map%lm2(l1, m1))* &
                      dLh(st_map%lm2(l1, m1))*or2(nR) * &
                      ( ddb(lm1, nR) - dLh(st_map%lm2(l1, m1))*or2(nR)*b(lm1, nR) )
```

Mode Methods

- Adaptive time step according to modified Courant-Friedrich-Levy criterium.
- Coriolis force treated explicitly.
- Transform on cylindrical grid for output possible.