EPM Group, Institut für Geophysik, ETH Zürich

Introducing PlesioGeostroPy

a Python realization of the PG model

Min, Jingtao

Oct. 2023

Outline

PG, QG, and all the geostrophic models

Premise: the planet is rotating so fast that the Coriolis effect is dominant in the rotating frame
(=$2\boldsymbol{\Omega} \times \mathbf{u}$ is the dominant term in Navier-Stokes)

Geostrophy: pressure gradient balanced by Coriolis force $2\boldsymbol{\Omega} \times \mathbf{u} + \nabla p = \mathbf{0}$
(this balance produces Taylor columns, i.e. invariance of velocity along the rotation axis)

Conclusion: the planet is "nearly" geostrophic
(=the flow organized into columns)
(=Variation $|\frac{\partial \mathbf{u}}{\partial z}| \ll |\nabla_e \mathbf{u}|$)

Premise somewhat confirmed by numerical computations (see e.g. right picture, from Jault and Finlay (2015)), at least outside the tangent cylinder.

PG, QG, and all the geostrophic models

Strictly geostrophic

$$\mathbf{u} = \nabla \times (\Psi \hat{\mathbf{z}})$$

does not satisfy non-penetrating BC.

How to exploit the "near"-geostrophy?

We can choose an ansatz for the flow that is close to geostrophic flow.

Also other options

PG, QG, and all the geostrophic models

How to exploit the "near"-geostrophy?

We can choose an ansatz for the flow that is close to geostrophic flow.

Also other options

Strictly geostrophic

$$\mathbf{u} = \nabla \times (\Psi \hat{\mathbf{z}})$$

does not satisfy non-penetrating BC.

Geostrophic + vertical flow at boundary

$$\mathbf{u} = \nabla \times (\Psi \hat{\mathbf{z}}) + \mathbf{u}_z$$

does not satisfy divergence-free.

PG, QG, and all the geostrophic models

How to exploit the "near"-geostrophy?

We can choose an ansatz for the flow that is close to geostrophic flow.

Also other options

Strictly geostrophic

$$\mathbf{u} = \nabla \times (\Psi \hat{\mathbf{z}})$$

does not satisfy non-penetrating BC.

Geostrophic + vertical flow at boundary

$$\mathbf{u} = \nabla \times (\Psi \hat{\mathbf{z}}) + \mathbf{u}_z$$

does not satisfy divergence-free.

Schaeffer and Cardin QG ansatz

$$\mathbf{u} = \frac{1}{H} \nabla \times (\Psi \hat{\mathbf{z}}) + \frac{z}{sH^2} \frac{dH}{ds} \frac{\partial \Psi}{\partial \phi} \hat{\mathbf{z}}$$

non-penetration + divergence-free.

PG, QG, and all the geostrophic models

Plesio-Geostrophy (PG) (Jackson and Maffei 2020) is built on the QG ansatz by Schaeffer and Cardin. In the absence of magnetic diffusion, PG converts

Vector fields $\mathbf{u}(\mathbf{r})$ and $\mathbf{B}(\mathbf{r})$ living in 3D space,
along with their evolution equations (Navier-Stokes + magnetic induction eqs)

$\longrightarrow$ exactly to $\longrightarrow$

15 PG quantities: Ψ, $\overline{B_s^2}$, $\overline{B_\phi^2}$, $\overline{B_s B_\phi}$, $\widetilde{B_s B_z}$, $\widetilde{B_\phi B z}$, $\widetilde{z B_s^2}$, $\widetilde{z B_\phi^2}$, $\widetilde{z B s B_\phi}$, B_{es}, $B_{e\phi}$, B_{ez}, $B_{es,z}$, $B_{e\phi,z}$, B_r
all of which live in 2D space, along with their evolution equations.

How did I end up here?

Since the start of my doctoral studies, I

- studied surface operators and tried to derive the boundary terms in the diffusive torsional oscillation (TO) equation for 2 months $\Longrightarrow$ didn't lead anywhere;
- studied torsional Alfvén waves and calculated the 1-D eigenmodes of the torsional oscillation for 2 month $\Longrightarrow$ didn't lead anywhere;
- studied anelastic approximation for 1 month in order to work on the anelastic version of QuICC, which may help the Jupiter simulation $\Longrightarrow$ project was called off and deemed unpromising before I could start to think about the numerics;
- picked up the thread and studied the reflection of Alfvén waves at the fluid-solid interface for 2 months $\Longrightarrow$ wrote a sixty-page document and yet had more questions than before;
- have been mainly working on the PG model since Aug 2023.

Outline

The old implementation - code *Daria*

Mathematica implementation (previous member of the group, Dr. Daria Holdenried-Chernoff)

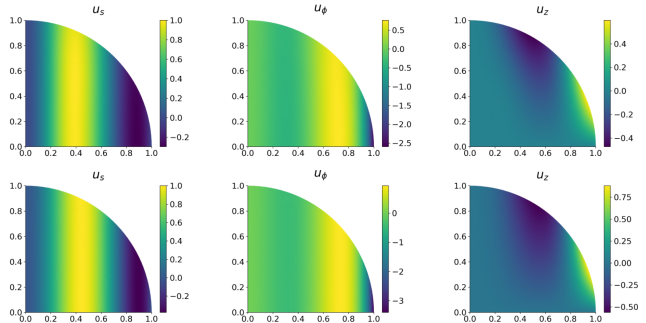


Figure: Eigenmodes calculated using 3-D code *Jiawen* (top) and PG code *Daria*.

Why reinvent the wheels when we already have a functioning implementation?

The old implementation - code *Daria*

Citizens of Gotham, riddle me this... what is this code snippet doing?

```
In[•]:= Les = 1/s D[s Bs2[s, p, t], s] + s^2/H[s] (BsH[s, p, z, t] * BsH[s, p, z, t] + BsmH[s, p, z, t] * BsmH[s, p, z, t]) + 1/s D[BsBp[s, p, t], p] - 1/s Bp2[s, p, t] - BsH[s, p, z, t] * BzH[s, p, z, t] -
    BsmH[s, p, z, t] * BzmH[s, p, z, t];

In[•]:= Lestmp =
    Les /. Bs2[s, p, t] → A0[s, p, t] + x * A[s, p, t] /. D[Bs2[s, p, t], s] → D[A0[s, p, t] + x * A[s, p, t], s] /. D[Bs2[s, p, t], p] → D[A0[s, p, t] + x * A[s, p, t], p] /.
                                Bp2[s, p, t] → B0[s, p, t] + x * B[s, p, t] /. D[Bp2[s, p, t], s] → D[B0[s, p, t] + x * B[s, p, t], s] /. D[Bp2[s, p, t], p] → D[B0[s, p, t] + x * B[s, p, t], p] /.
                            BsBp[s, p, t] → C0[s, p, t] + x * Csp[s, p, t] /. D[BsBp[s, p, t], s] → D[C0[s, p, t] + x * Csp[s, p, t], s] /. D[BsBp[s, p, t], p] → D[C0[s, p, t] + x * Csp[s, p, t], p] /.
                        BsmH[s, p, z, t] → amh0[s, p, z, t] + x * amh[s, p, z, t] /. D[BsmH[s, p, z, t], s] → D[amh0[s, p, z, t], s] + x * D[amh[s, p, z, t], s] /.
                    D[BsmH[s, p, z, t], p] → D[amh0[s, p, z, t], p] + x * D[amh[s, p, z, t], p] /. BsH[s, p, z, t] → ah0[s, p, z, t] + x * ah[s, p, z, t] /.
                D[BsH[s, p, z, t], s] → D[ah0[s, p, z, t], s] + x * D[ah[s, p, z, t], s] /. D[BsH[s, p, z, t], p] → D[ah0[s, p, z, t], p] + x * D[ah[s, p, z, t], p] /.
            BpmH[s, p, z, t] → bmh0[s, p, z, t] + x * bmh[s, p, z, t] /. D[BpmH[s, p, z, t], s] → D[bmh0[s, p, z, t], s] + x * D[bmh[s, p, z, t], s] /.
        D[BpmH[s, p, z, t], p] → D[bmh0[s, p, z, t], p] + x * D[bmh[s, p, z, t], p] /. BpH[s, p, z, t] → bh0[s, p, z, t] + x * bh[s, p, z, t] /.
    D[BpH[s, p, z, t], s] → D[bh0[s, p, z, t], s] + x * D[bh[s, p, z, t], s] /. D[BpH[s, p, z, t], p] → D[bh0[s, p, z, t], p] + x * D[bh[s, p, z, t], p] /.
D[BpH[s, p, z, t], z] → D[bh0[s, p, z, t], z] + x * D[bh[s, p, z, t], z] /. BzmH[s, p, z, t] → cmh0[s, p, z, t] + x * cmh[s, p, z, t] /. BzH[s, p, z, t] → ch0[s, p, z, t] + x * ch[s, p, z, t] /.
D[BzH[s, p, z, t], z] → D[ch0[s, p, z, t], z] + x * D[ch[s, p, z, t], z] /. D[BzH[s, p, z, t], s] → D[ch0[s, p, z, t], s] + x * D[ch[s, p, z, t], s] /.
D[BzmH[s, p, z, t], s] → D[cmh0[s, p, z, t], s] + x * D[cmh[s, p, z, t], s];
```

The old implementation - code *Daria*

Citizens of Gotham, riddle me this... what is this code snippet doing?

```
In[•]:= Les = 1/s D[s Bs2[s, p, t], s] + s^2/H[s] (BsH[s, p, z, t] * BsH[s, p, z, t] + BsmH[s, p, z, t] * BsmH[s, p, z, t]) + 1/s D[BsBp[s, p, t], p] - 1/s Bp2[s, p, t] + BsH[s, p, z, t] * BzH[s, p, z, t] -
    BsmH[s, p, z, t] * BzmH[s, p, z, t];

In[•]:= Lestmp =
    Les /. Bs2[s, p, t] → A0[s, p, t] + x * A[s, p, t] /. D[Bs2[s, p, t], s] → D[A0[s, p, t] + x * A[s, p, t], s] /. D[Bs2[s, p, t], p] → D[A0[s, p, t] + x * A[s, p, t], p] /.
                Bp2[s, p, t] → B0[s, p, t] + x * B[s, p, t] /. D[Bp2[s, p, t], s] → D[B0[s, p, t] + x * B[s, p, t], s] /. D[Bp2[s, p, t], p] → D[B0[s, p, t] + x * B[s, p, t], p] /.
            BsBp[s, p, t] → C0[s, p, t] + x * Csp[s, p, t] /. D[BsBp[s, p, t], s] → D[C0[s, p, t] + x * Csp[s, p, t], s] /. D[BsBp[s, p, t], p] → D[C0[s, p, t] + x * Csp[s, p, t], p] /.
        BsmH[s, p, z, t] → amh0[s, p, z, t] + x * amh[s, p, z, t] /. D[BsmH[s, p, z, t], s] → D[amh0[s, p, z, t], s] + x * D[amh[s, p, z, t], s] /.
    D[BsmH[s, p, z, t], p] → D[amh0[s, p, z, t], p] + x * D[amh[s, p, z, t], p] /. BsH[s, p, z, t] → ah0[s, p, z, t] + x * ah[s, p, z, t] /.
  D[BsH[s, p, z, t], s] → D[ah0[s, p, z, t], s] + x * D[ah[s, p, z, t], s] /. D[BsH[s, p, z, t], p] → D[ah0[s, p, z, t], p] + x * D[ah[s, p, z, t], p] /.
    BpmH[s, p, z, t] → bmh0[s, p, z, t] + x * bmh[s, p, z, t] /. D[BpmH[s, p, z, t], s] → D[bmh0[s, p, z, t], s] + x * D[bmh[s, p, z, t], s] /.
 D[BpmH[s, p, z, t], p] → D[bmh0[s, p, z, t], p] + x * D[bmh[s, p, z, t], p] /. BpH[s, p, z, t] → bh0[s, p, z, t] + x * bh[s, p, z, t] /.
  D[BpH[s, p, z, t], s] → D[bh0[s, p, z, t], s] + x * D[bh[s, p, z, t], s] /. D[BpH[s, p, z, t], p] → D[bh0[s, p, z, t], p] + x * D[bh[s, p, z, t], p] /.
D[BpH[s, p, z, t], z] → D[bh0[s, p, z, t], z] + x * D[bh[s, p, z, t], z] /. BzmH[s, p, z, t] → cmh0[s, p, z, t] + x * cmh[s, p, z, t] /. BzH[s, p, z, t] → ch0[s, p, z, t] + x * ch[s, p, z, t] /.
  D[BzH[s, p, z, t], z] → D[ch0[s, p, z, t], z] + x * D[ch[s, p, z, t], z] /. D[BzH[s, p, z, t], s] → D[ch0[s, p, z, t], s] + x * D[ch[s, p, z, t], s] /.
D[BzmH[s, p, z, t], s] → D[cmh0[s, p, z, t], s] + x * D[cmh[s, p, z, t], s];
```

This rewrites the magnetic quantities in Lorentz force $\overline{L_\phi}$ as background + perturbation for linearization. e.g. `Bp2[s, p, t]->B0[s, p, t] + x*B[s, p, t]` equals to $\overline{B_\phi^2} = \overline{B_{\phi_0}^2} + \epsilon \overline{B_\phi^2}'$

The old implementation - code *Daria*

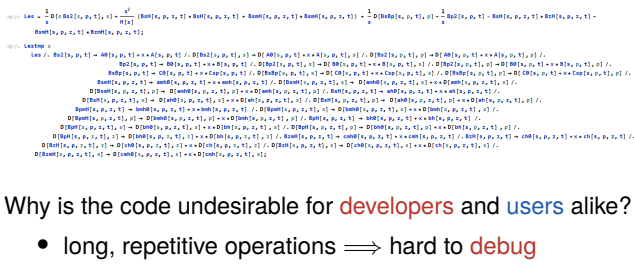
```
In[•]:= Les = 1/s D[s Bs2[s, p, t], s] + s^2/H[s] (BsH[s, p, z, t] * BsH[s, p, z, t] + BsmH[s, p, z, t] * BsmH[s, p, z, t]) + 1/s D[BsBp[s, p, t], p] - 1/s Bp2[s, p, t] + BsH[s, p, z, t] * BzH[s, p, z, t] -
      BsmH[s, p, z, t] * BzmH[s, p, z, t];

In[•]:= Lestmp =
     Les /. Bs2[s, p, t] → A0[s, p, t] + x * A[s, p, t] /. D[Bs2[s, p, t], s] → D[ A0[s, p, t] + x * A[s, p, t], s] /. D[Bs2[s, p, t], p] → D[ A0[s, p, t] + x * A[s, p, t], p] /.
                     Bp2[s, p, t] → B0[s, p, t] + x * B[s, p, t] /. D[Bp2[s, p, t], s] → D[ B0[s, p, t] + x * B[s, p, t], s] /. D[Bp2[s, p, t], p] → D[ B0[s, p, t] + x * B[s, p, t], p] /.
                    BsBp[s, p, t] → C0[s, p, t] + x * Csp[s, p, t] /. D[BsBp[s, p, t], s] → D[ C0[s, p, t] + x * Csp[s, p, t], s] /. D[BsBp[s, p, t], p] → D[ C0[s, p, t] + x * Csp[s, p, t], p] /.
                  BsmH[s, p, z, t] → amh0[s, p, z, t] + x * amh[s, p, z, t] /. D[BsmH[s, p, z, t], s] → D[amh0[s, p, z, t], s] + x * D[amh[s, p, z, t], s] /.
                 D[BsmH[s, p, z, t], p] → D[amh0[s, p, z, t], p] + x * D[amh[s, p, z, t], p] /. BsH[s, p, z, t] → ah0[s, p, z, t] + x * ah[s, p, z, t] /.
                D[BsH[s, p, z, t], s] → D[ah0[s, p, z, t], s] + x * D[ah[s, p, z, t], s] /. D[BsH[s, p, z, t], p] → D[ah0[s, p, z, t], p] + x * D[ah[s, p, z, t], p] /.
              BpmH[s, p, z, t] → bmh0[s, p, z, t] + x * bmh[s, p, z, t] /. D[BpmH[s, p, z, t], s] → D[bmh0[s, p, z, t], s] + x * D[bmh[s, p, z, t] , s] /.
             D[BpmH[s, p, z, t], p] → D[bmh0[s, p, z, t], p] + x * D[bmh[s, p, z, t] , p] /. BpH[s, p, z, t] → bh0[s, p, z, t] + x * bh[s, p, z, t] /.
            D[BpH[s, p, z, t], s] → D[bh0[s, p, z, t], s] + x * D[bh[s, p, z, t] , s] /. D[BpH[s, p, z, t], p] → D[bh0[s, p, z, t], p] + x * D[bh[s, p, z, t] , p] /.
           D[BpH[s, p, z, t], z] → D[bh0[s, p, z, t], z] + x * D[bh[s, p, z, t] , z] /. BzmH[s, p, z, t] → cmh0[s, p, z, t] + x * cmh[s, p, z, t] /. BzH[s, p, z, t] → ch0[s, p, z, t] + x * ch[s, p, z, t] /.
         D[BzH[s, p, z, t], z] → D[ch0[s, p, z, t], z] + x * D[ch[s, p, z, t], z] /. D[BzH[s, p, z, t], s] → D[ch0[s, p, z, t], s] + x * D[ch[s, p, z, t], s] /.
      D[BzmH[s, p, z, t], s] → D[cmh0[s, p, z, t], s] + x * D[cmh[s, p, z, t], s];
```

Why is the code undesirable for developers and users alike?

- long, repetitive operations $\Longrightarrow$ hard to debug
- namings: the background and perturbation fields are named from `A` to `H` $\Longrightarrow$ hard to debug or invoke.
- At some point I even need a correspondence table to decipher the code.

The old implementation - code *Daria*



Why is the code undesirable for developers and users alike?

- long, repetitive operations $\Longrightarrow$ hard to debug
- namings: the background and perturbation fields are named from `A` to `H` $\Longrightarrow$ hard to debug or invoke.
- At some point I even need a correspondence table to decipher the code.

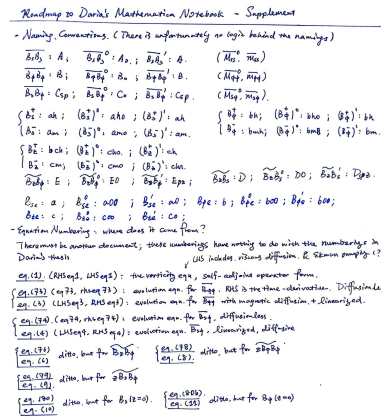


Figure: List of physical variables and their names in the code.

The old implementation - code *Daria*

```
psi[m_, n_] := (1 - s^2)^(3/2) s^m JacobiP[n - 1, 3/2, m, 2 s^2 - 1]

Fm[m_] := Piecewise[{{1, m == 0}, {s, m == 1}, {s^Abs[m]-2, m > 1}}]
βeven[m_] := Piecewise[{{-1/2, m == 0}, {1 - 1/2, m == 1}, {m - 5/2, m > 1}}]

BackgRegFactor = 1; (*s (-1+s^2);*)

BsExp[m_, n_] := BackgRegFactor * (1 - s^2)^1/2 s^Abs[m]-1 JacobiP[n - 1, 1, Abs[m] - 3/2, 2 s^2 - 1]

BpExp[m_, n_] := BackgRegFactor * (1 - s^2)^1/2 s^Abs[m]+1 JacobiP[n - 1, 1, Abs[m] + 1/2, 2 s^2 - 1]

BzExp[m_, n_] := BackgRegFactor * (1 - s^2) s^Abs[m] JacobiP[n - 1, 2, Abs[m] - 1/2, 2 s^2 - 1]

Magnetic moments expansions for toroidal background field

Bsp[m_, n_] := (1 - s^2)^1/2 * s^Abs[m-2] * JacobiP[n - 1, 1, Abs[m - 2] - 1/2, 2 s^2 - 1]

Bpp[m_, n_] := (1 - s^2)^1/2 * Fm[m] * s^2 * JacobiP[n - 1, 1, βeven[m] + 2, 2 s^2 - 1] (*Wherever Bpp

Bpz[m_, n_] := (1 - s^2) s^m-1 JacobiP[n - 1, 2, m - 3/2, 2 s^2 - 1]

Bzpp[m_, n_] := (1 - s^2) Fm[m] * s^2 * JacobiP[n - 1, 2, βeven[m] + 2, 2 s^2 - 1] (*Wherever Bzpp
there needs to be a Bzsp contribution to ensure correct regularity. Note extra H from

Bzsp[m_, n_] := (1 - s^2) s^Abs[m-2] JacobiP[n - 1, 2, Abs[m - 2] - 1/2, 2 s^2 - 1]

bs0[m_, n_] := (1 - s^2) s^Abs[m]-1 JacobiP[n - 1, 2, Abs[m] - 1 - 1/2, 2 s^2 - 1]

bp0[m_, n_] := (1 - s^2) s^Abs[m]+1 JacobiP[n - 1, 2, Abs[m] + 1 - 1/2, 2 s^2 - 1] (*Factor of H ensu
```

Inflexible expansions: the expansion and the construction of matrices are hard coded in each notebook.

To change the expansion, one needs to

- produce a complete new Mathematica notebook;
- manually change the way the code collects the matrix elements.

Especially cumbersome to implement coupling, or if the bases do not coincide with the field quantities (more detail later).

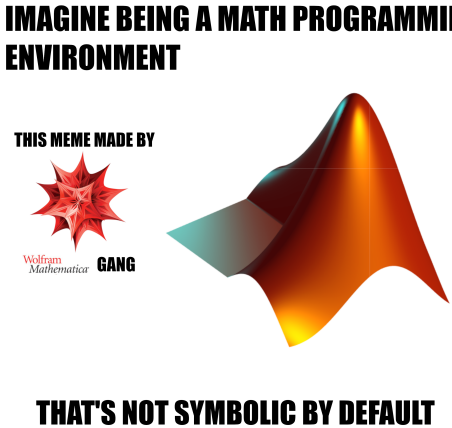
The old implementation

Other problems

- The code is not efficient numerically, nor easily linked to efficient numerical libraries.
- Cryptic Mathematica syntax sugars

```
In[•]:= sol = Solve[JacobiP[colN, α, β, x] == 0, x];
        rlocation = N[x /. # & /@ sol, prec];
```

- When something unexpected happens, difficult to understand the cause
- Or perhaps I just don't like Mathematica



Outline

Introducing PlesioGeostroPy

Make Plesio-Geostrophy model easy to use!

Symbolic manipulation
[`sympy`]

- all PG equations stored, easily accessible
- linearization
- configure expansion
- collect matrix elements

Demo notebook available.

Numerical computation
[`mpmath`,`numpy`/`scipy`]

- compute matrix elements
- quadrature
- eigenvalue solver
- time stepper
- comes in both std (`numpy+scipy` backend) and multi-precision (`mpmath`)

Post-processing
[`matplotlib`]

First batch of result: a glimpse of efficiency

Test problem: eigenvalue problem with Malkus background field, diffusionless.

A not-so-well-controlled test:
Assemble mass and stiffness matrices at the size of 124×124 (PlesioGeostroPy) and 73×73 (Mathematica)

Three versions:

- Code Daria, eval to 32nd digit
- PlesioGeostroPy, multiprecision eval (same precision)
- PlesioGeostroPy, scipy eval (double precision)

Table: Time lapse for computing mass and stiffness matrices

Code (version)	Time $\mathbf{M}$	Time $\mathbf{K}$
Code Daria Mathematica	5s	50s*
PlesioGeostroPy-mp32	16s	35s
PlesioGeostroPy-scipy	400ms	750ms

*may not be representative as this involves other operations

First batch of result: eigenvalues with Malkus background field

Test: eigenvalue problem with Malkus background field, diffusionless.

Eigenvalues (right) are computed for $\mathrm{Le} = 10^{-4}$, the errors are measured relative to the analytical eigenvalue for the PG model.

`PlesioGeostroPy` $\sim$ Code *Daria* mp32 $\approx$ scipy.

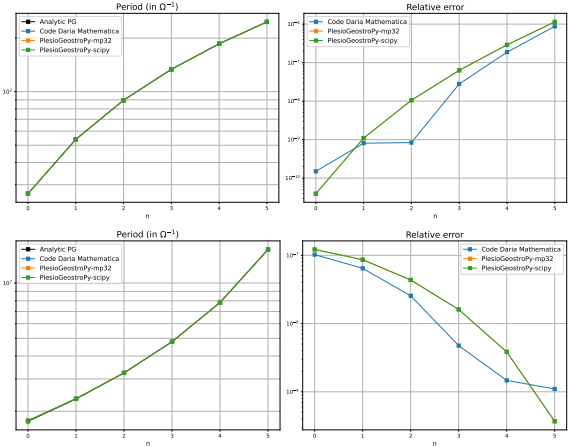
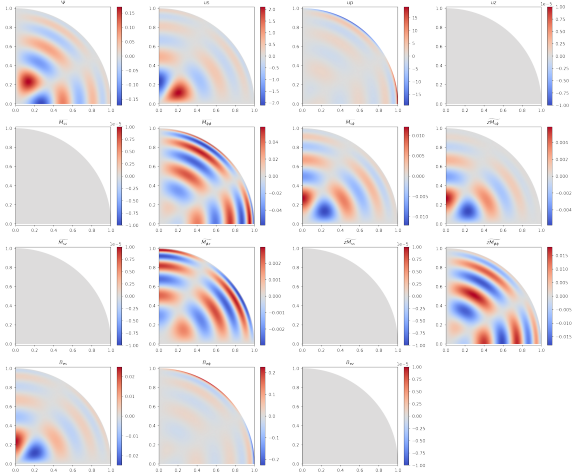


Figure: Periods and errors of the lowest 6 eigenmodes for the fast(top) and slow(bottom) branch.

First batch of result: eigenmodes with Malkus background field

Test: eigenvalue problem with Malkus background field, diffusionless.

Eigenmodes are computed for $\mathrm{Le} = 10^{-4}$.
Right fig. shows the equatorial sections of the 5-th eigenmode for $m = 3$.



First batch of result: eigenmodes with Malkus background field

Test: eigenvalue problem with Malkus background field, diffusionless.

Eigenmodes are computed for $\mathrm{Le} = 10^{-4}$.

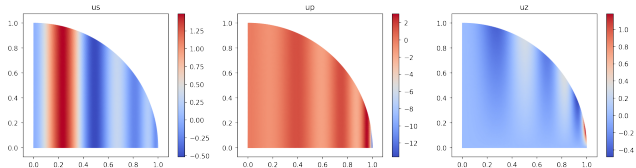


Figure: Meridional sections of the velocity field components for the 5-th eigenmode for $m = 3$.

Outline

To-do list

- Alternative expansions (even alternative evolution equations)
  - Additional coupling revealed (before group retreat), unlikely to implement all low-order coupling in the current way.
  - Need some thoughts + pen-and-paper derivation.
- Reduced systems of the eigenvalue problems
  - The eigenvalue problem with $\mathbf{u}^0 = 0$ can always be reduced to a problem with two variables, instead of 15.
- Treatment of the B_r
  - Circumvented in the eigenvalue problem.
  - Required for time stepping.
  - Possible to incorporate other than insulating BC?
- Time step the system