SWQU Workshop Day 2:
Open-source Flux Transport (OFT)

Ronald M. Caplan, Miko Stulajter, Jon Linker,
Cooper Downs, James Turtle, Lisa Upton, Raphael Attié,
Nick Arge, Carl Henney, and Bibhuti Jha

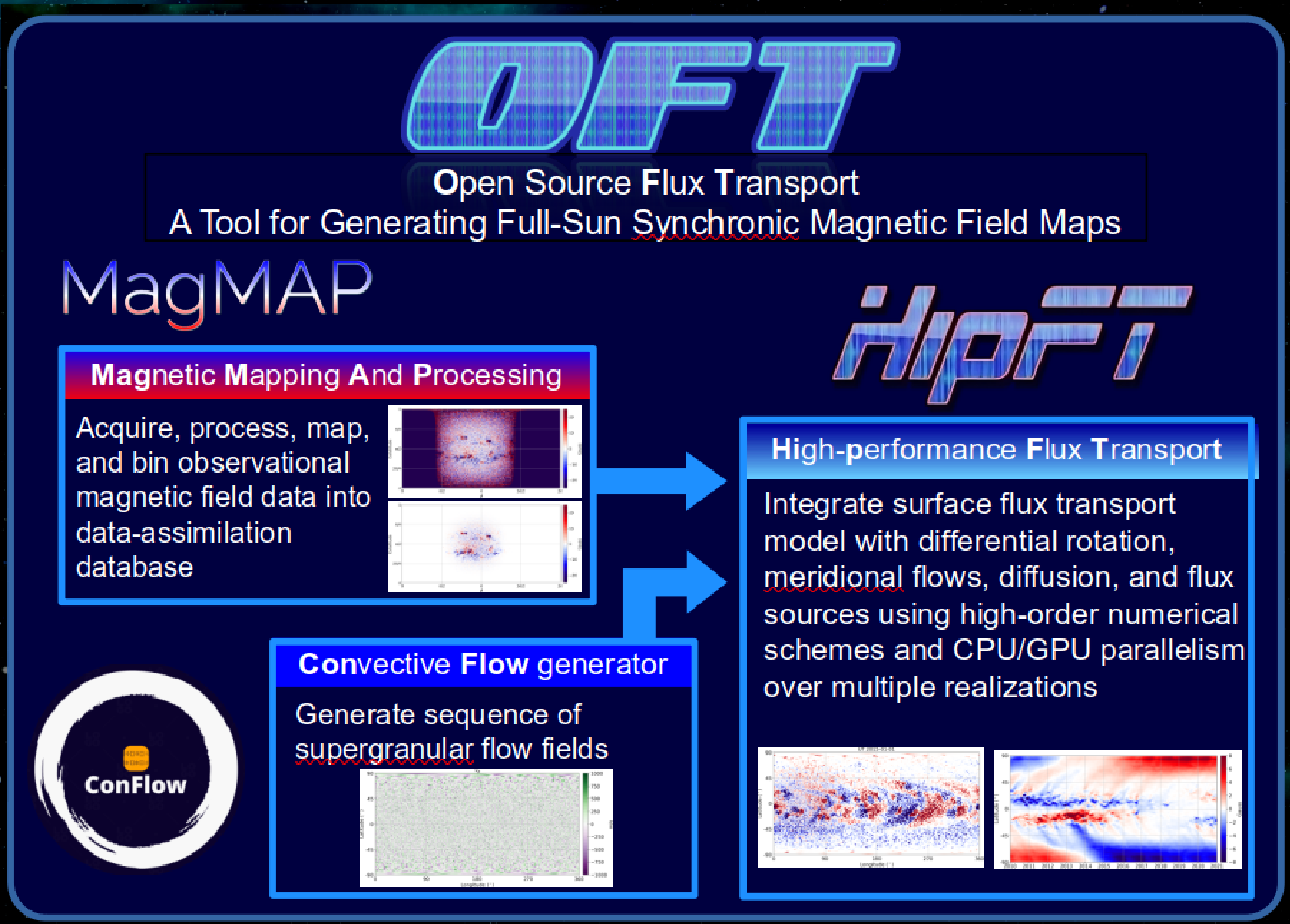
Predictive Science Inc.

NASA NSF

OFT

Outline

- OFT Overview:
  - Models
  - Numerical Methods
  - Code implementations
- BREAK
- How to run OFT
- How to Install OFT
  - Mac (homebrew and macports)
  - Windows (10 or 11 with WSL)
  - Linux
- Assignment
- BREAK



Re-cap of Surface Flux Transport (SFT) Models

- MHD (and other) global models require solar surface magnetic field data as input boundary conditions
- While observed by multiple instruments, routinely only from the Sun-Earth line of sight
- In order to make a global map, old data from the Sun-Earth line can be used (e.g. Carrington/"synoptic" maps), but this is problematic for time-dependent models, especially during solar maximum when the Sun is changing rapidly
- A way to mitigate this problem is to run a data-assimilative surface flux transport model (SFT) that models the Sun's surface flows to transport the field
- Although SFT models miss new far-side flux emergence, they can accurately predict how the most recently assimilated data will change over time on the back of the Sun
- SFTs are also very useful for testing models of the stellar dynamo, solar cycle models, etc.

Open-source Flux Transport (OFT)

SWQU

Space Weather with Quantified Uncertainty

- Developed as part of the “Improving Space Weather Predictions with Data-Driven Models of the Solar Atmosphere and Inner Heliosphere” SWQU project

github.com/predsci/oft

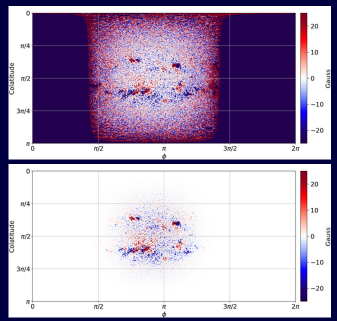
OFT

Open Source **F**lux **T**ransport
A Tool for Generating Full-Sun Synchronic Magnetic Field Maps

MagMAP

Magnetic Mapping And Processing

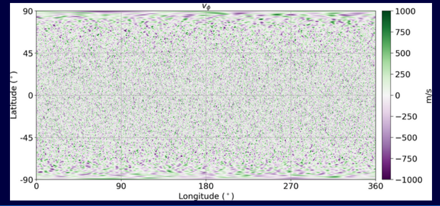
Acquire, process, map, and bin observational magnetic field data into data-assimilation database



ConFlow

Convective Flow generator

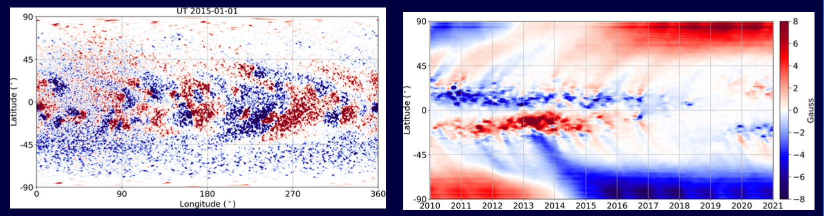
Generate sequence of supergranular flow fields



HipFT

High-performance Flux Transport

Integrate surface flux transport model with differential rotation, meridional flows, diffusion, and flux sources using high-order numerical schemes and CPU/GPU parallelism over multiple realizations

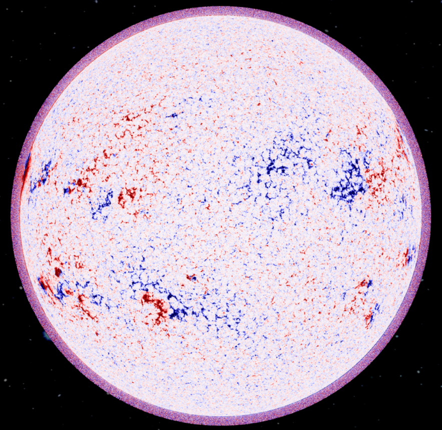


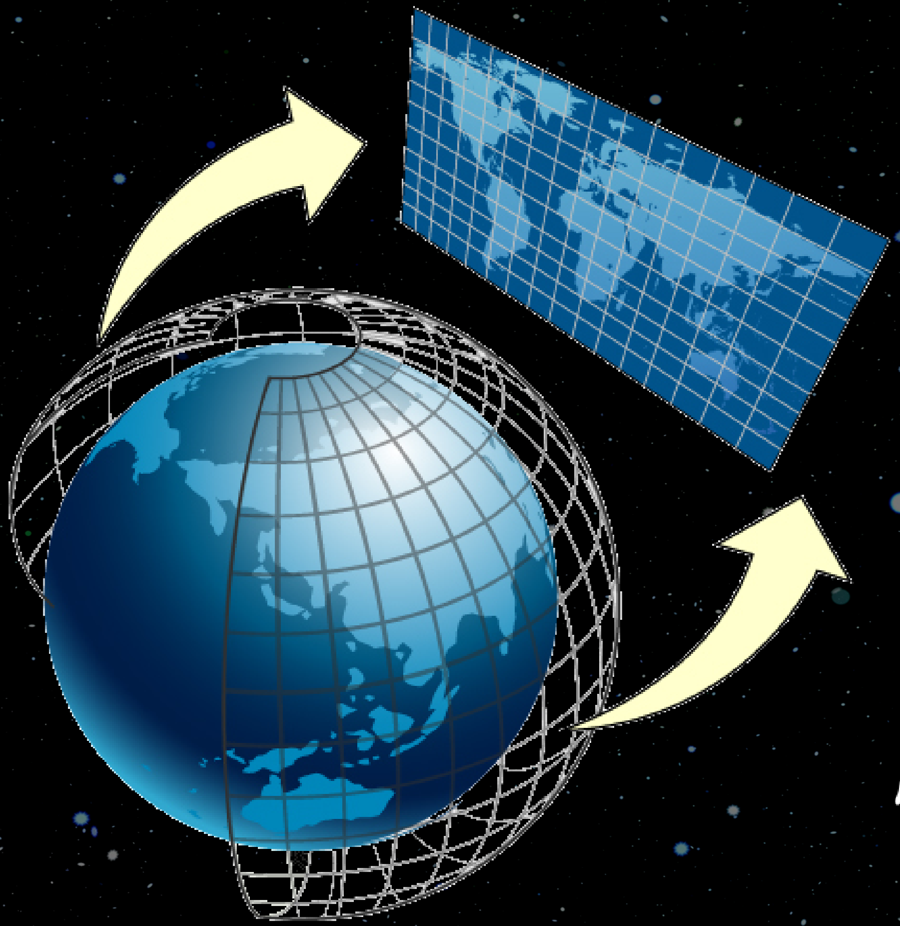
Magnetic Mapping and Processing (MagMAP)

MagMAP

Acquire, process, map, and bin observational magnetic field data into data-assimilation database

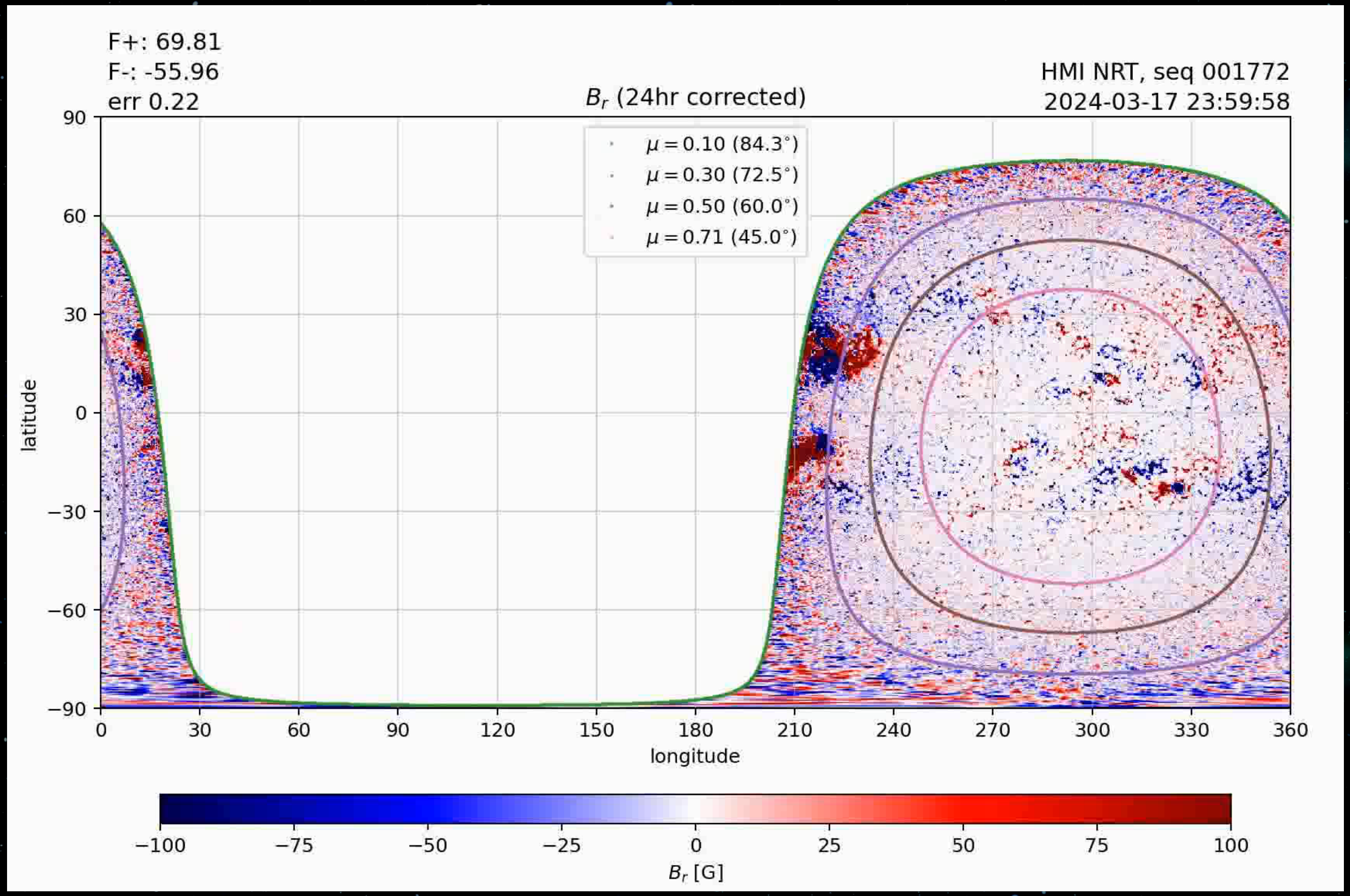
SDO HMI 720s LOS Magnetograms through JSOC `drms` py package





$$B_r = B_{LOS}/\mu$$

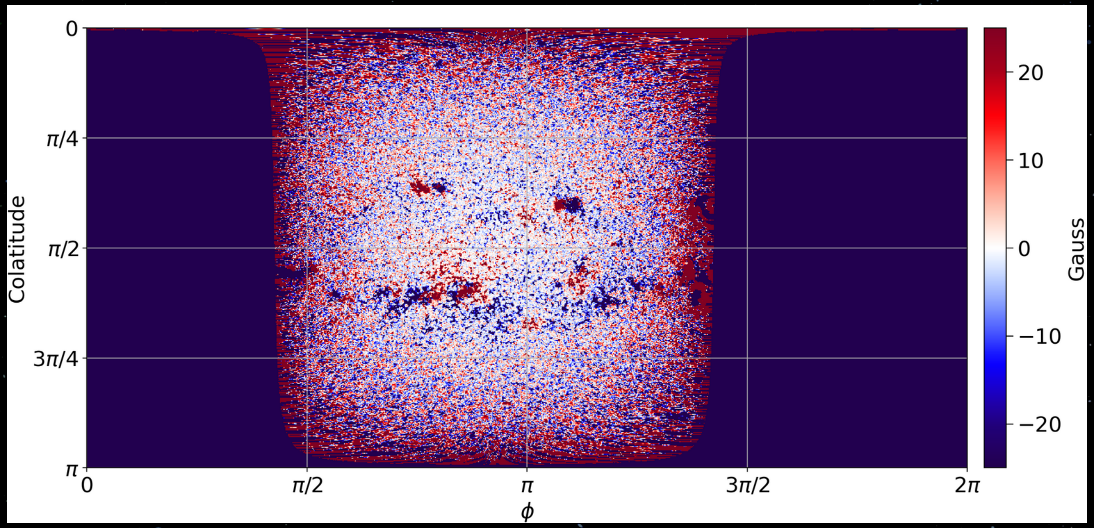
$$\mu = \cos\theta_d \in [0, 1]$$

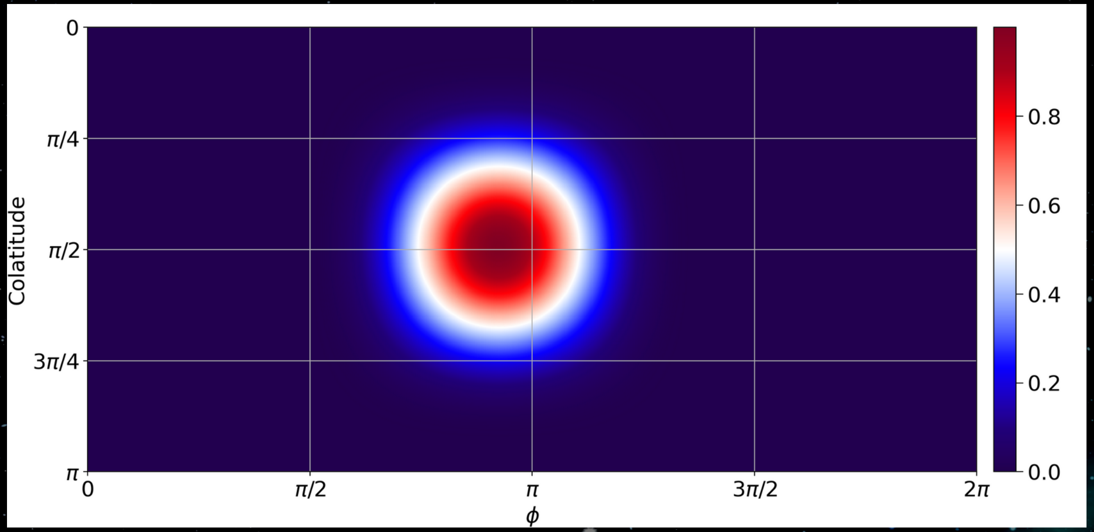


MagMAP

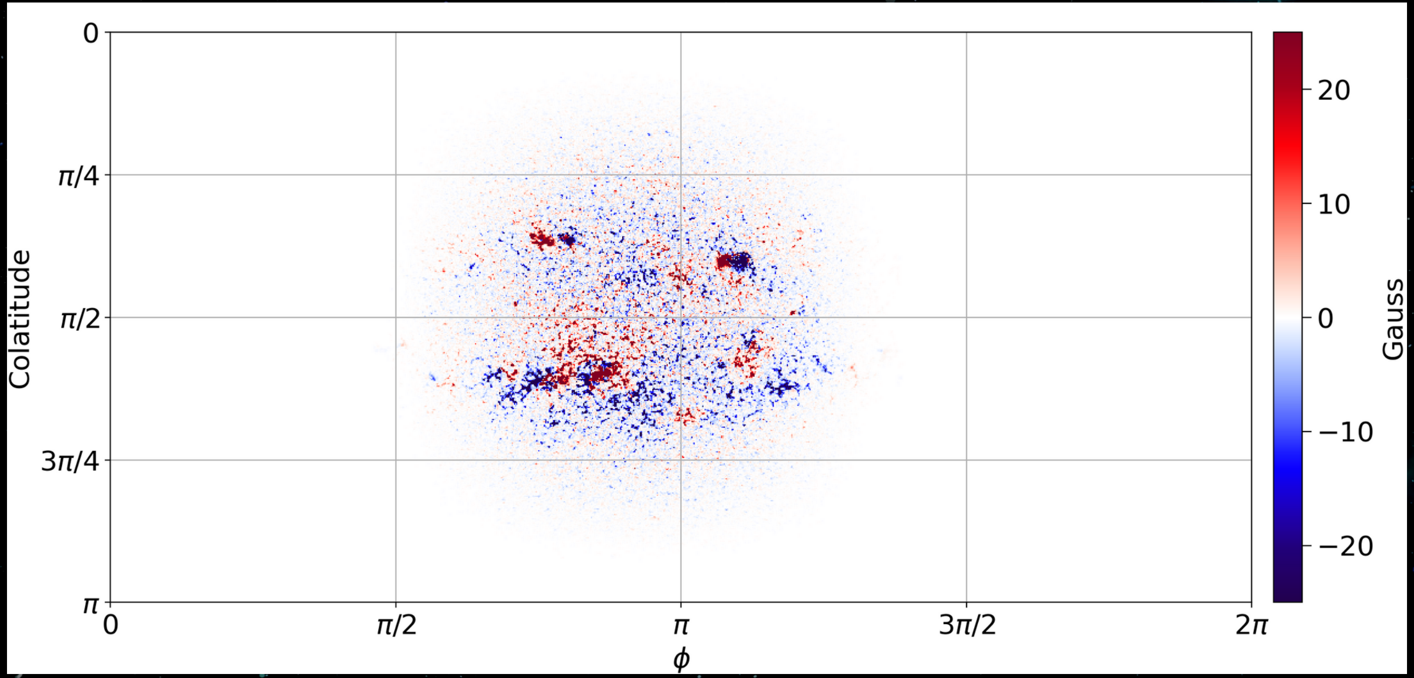
Makes three layers: Data, Default-weight-map, mu

$$\mu = \cos\theta_d \in [0,1]$$





Example of data as assimilated into HipFT:



github.com/predsci/magmap

COMING SOON

Convective Flow Generator (ConFlow)

- Diffusion in SFT models used as a proxy for the flux cancellation caused by granular and super-granular motions
- However, there are advantages to directly modeling these flows
- The default HipFT resolution of 1024x512 is high enough to resolve most of the super-granular scale sizes
- ConFlow generates a sequence of flow data encompassing random motions and super-granulation
- HipFT reads in the files and drives the FT with the flows
- **Some diffusion is still necessary to represent flux cancellation at smaller scales**

github.com/predsci/conflow

COMING SOON

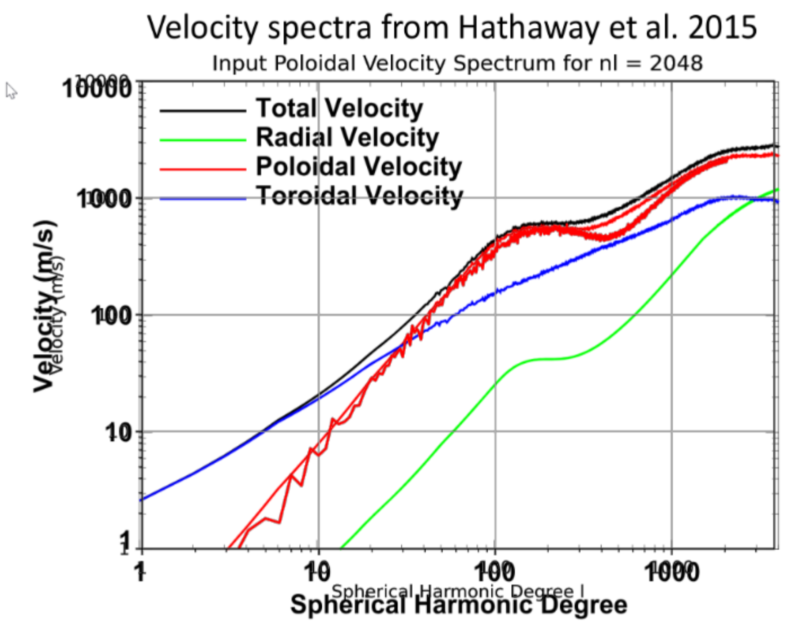
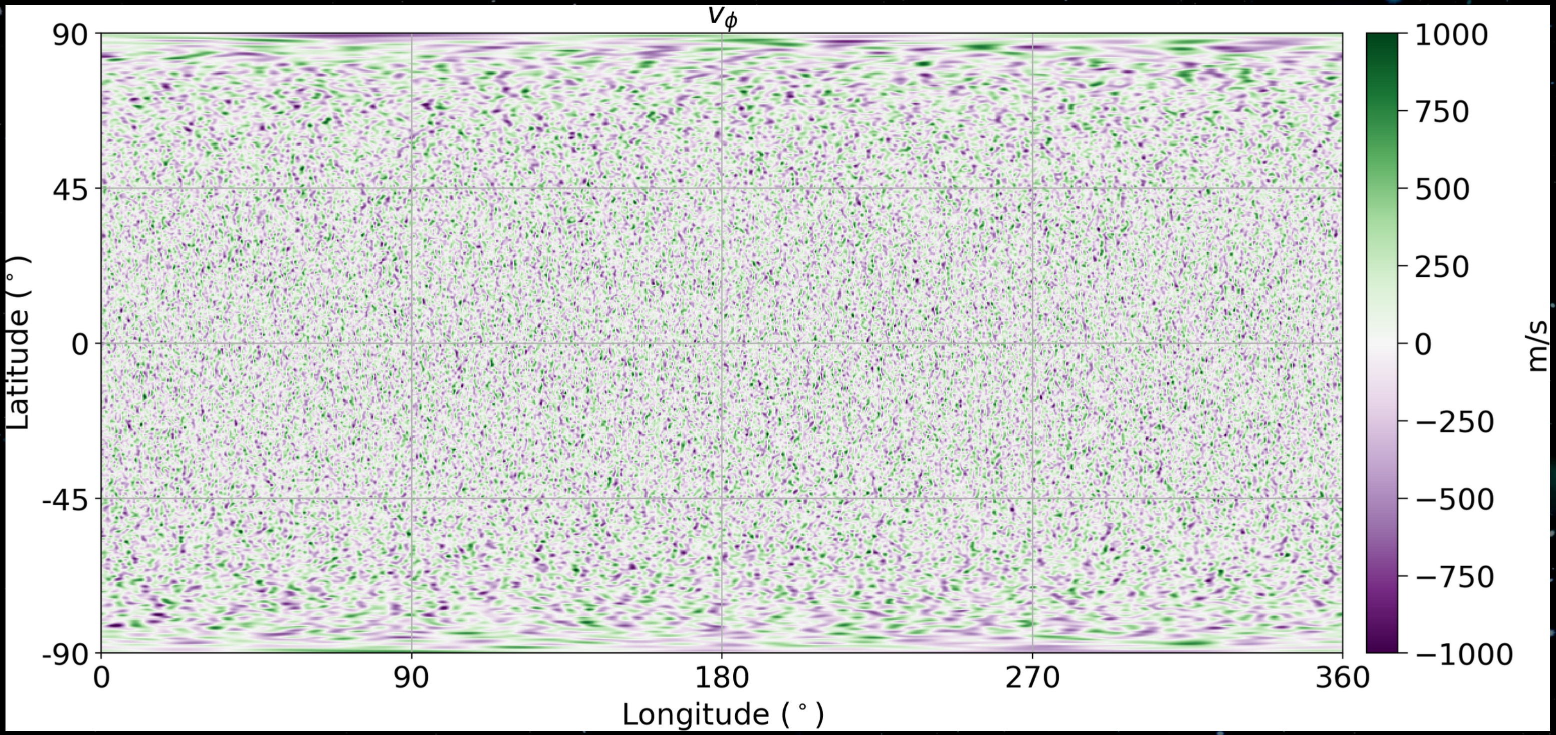


Figure 12. Total velocity spectrum (black line) from the data simulation. The contribution from the poloidal flow is shown in red. The contribution from the toroidal flow is shown in blue. The contribution from the radial flow is shown in green.

ConFlow

[Hathaway et. al. (2010,2015)]



High-performance Flux Transport (HipFT)

Implements advection, diffusion, data assimilation, and flux emergence over multiple realizations using high-accuracy numerical methods and CPU/GPU parallelism

$$\frac{\partial B_r}{\partial t} = -\nabla_s \cdot (B_r \mathbf{v}) + \nabla_s \cdot (\nu \nabla_s B_r) + S,$$

Advection

Diffusion

Data assimilation, flux emergence, etc.

github.com/predsci/hipft

HipFT Model: Flows

$$\nabla_s \cdot (B_r \mathbf{v}) = \frac{1}{R_\odot \sin\theta} \frac{\partial}{\partial \theta}(\sin\theta\, B_r\, v_\theta) + \frac{1}{R_\odot \sin\theta} \frac{\partial}{\partial \phi}(B_r\, v_\phi),$$

Differential rotation :

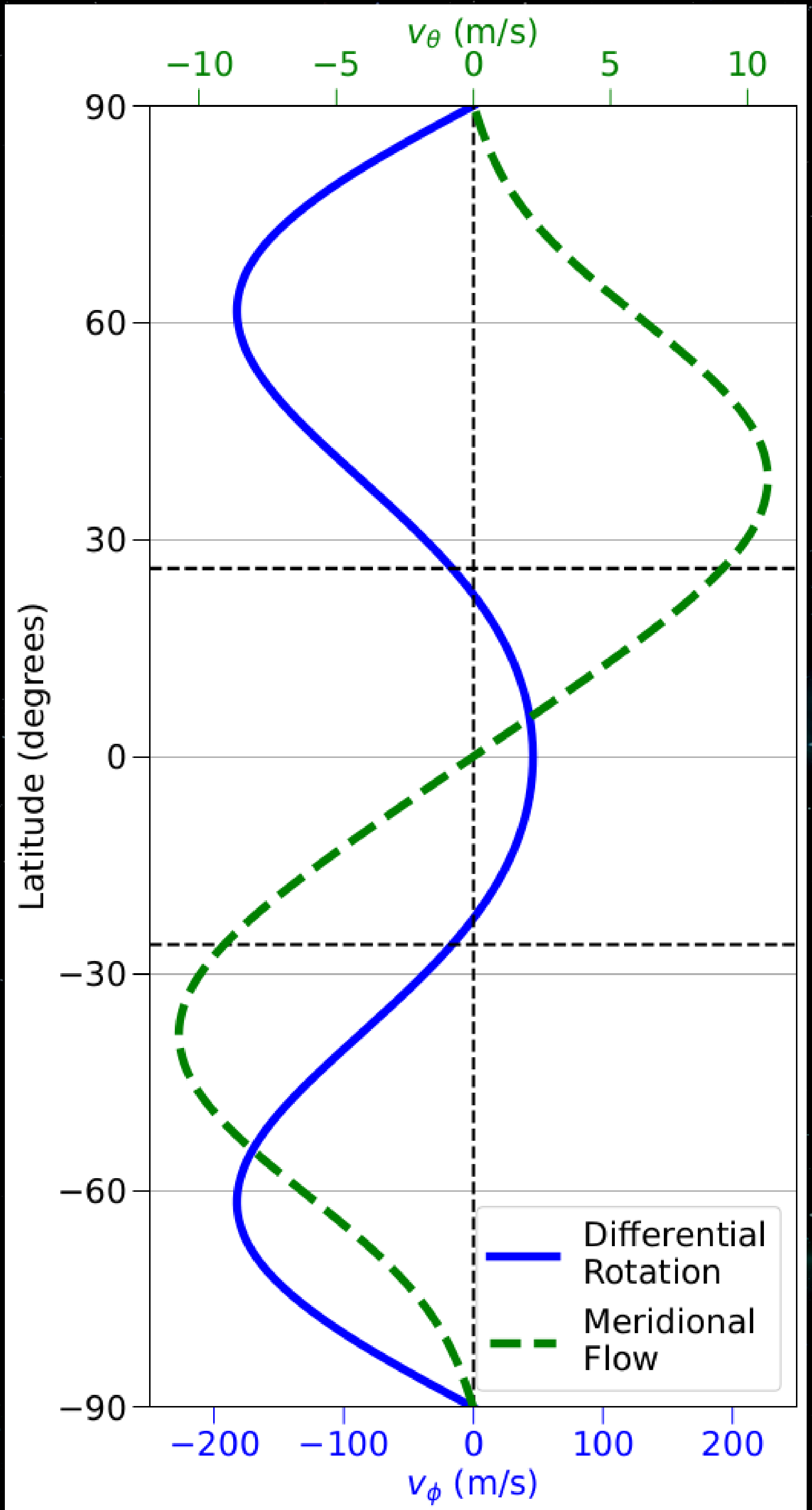
$$v_\phi(\theta) = \left[d_0 + d_2 \cos^2(\theta) + d_4 \cos^4(\theta)\right] \sin\theta,$$

Meridional Flows :

$$v_\theta(\theta) = -\left[m_1 \cos\theta + m_3 \cos^3\theta + m_5 \cos^5\theta\right] \sin\theta,$$

Flow Attenuation:

$$v_{\theta/\phi} \rightarrow v_{\theta/\phi} \left[1.0 - \tanh\left(\frac{|B_r|}{B_0}\right)\right]$$



HipFT Model: Diffusion

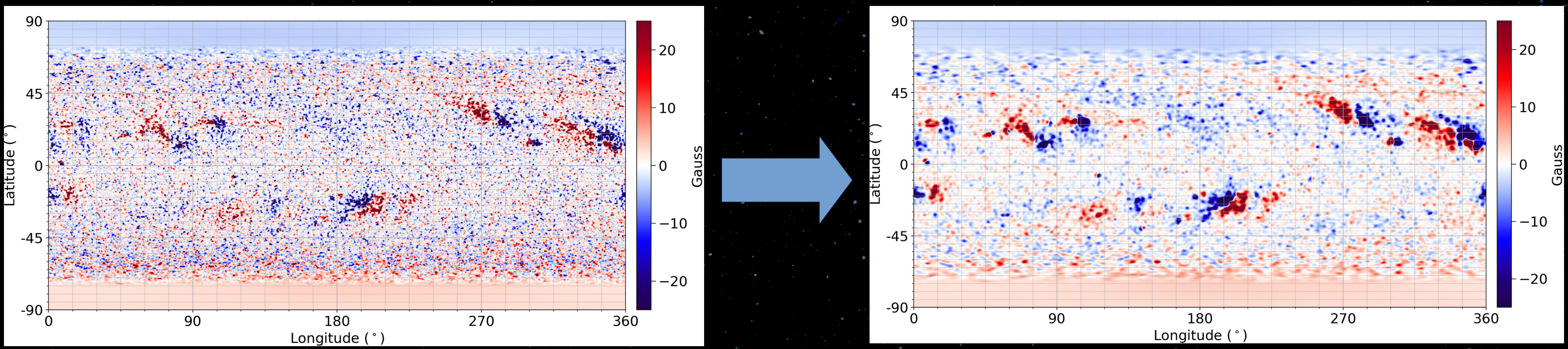
$$\nabla_s \cdot (\nu \nabla_s B_r) = \frac{1}{R_\odot^2 \sin\theta} \frac{\partial}{\partial \theta} \left(\nu(\theta, \phi, B_r) \sin\theta \frac{\partial B_r}{\partial \theta} \right) + \frac{1}{R_\odot^2 \sin^2\theta} \frac{\partial}{\partial \phi} \left(\nu(\theta, \phi, B_r) \frac{\partial B_r}{\partial \phi} \right)$$

- Diffusion coefficient can be constant, or a user-defined spatially varying file

$$\nu = 300\,\mathrm{km}^2/s$$

- HipFT can be used as a magnetogram smoother, in which case one can select a grid-based diffusion coefficient

$$\nu_{\mathrm{grid}} = (\Delta\theta)^2 + (\Delta\phi \sin\theta)^2$$



HipFT Model: Data Assimilation

- Data assimilation uses the output data from MagMAP

$$B_r^{new} = F\, B_{r;data} + (1 - F)\, B_r^{old}$$

- A default weighting function is included in the data cube

$$F = \begin{cases} \mu^4, & \mu \geq 0.1 \\ 0, & \mu < 0.1 \end{cases}$$

- The center-to-limb distance is also provided, which can be used to generate a user-defined custom weight profile:

$$B_r^{new} = B_r^{old} + \Delta B_r$$

$$\Delta B_r = F\,(B_{r;data} - B_r^{old})$$

$$F = \mu^{\alpha_\mu} \qquad \mu < \mu_{\mathrm{lim}} \;\&\; |\theta_1| < \theta_{1,\mathrm{lim}}, \qquad F = 0 \;\; \text{o.w.},$$

- Option to flux balance change in B to maintain the flux balance of the maps

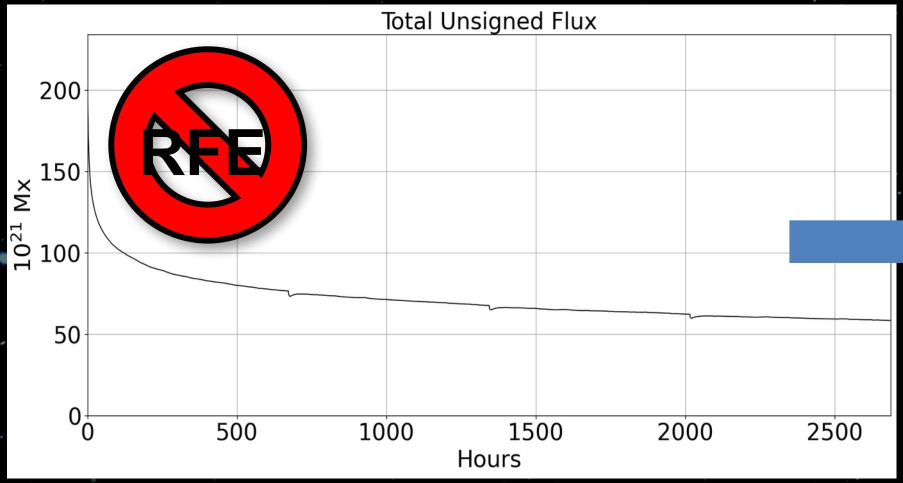
$$\Delta B_r = \begin{cases} \Delta B_r / \sqrt{|\Phi_+/\Phi_-|}, & \Delta B_r > 0 \\ \Delta B_r \sqrt{|\Phi_+/\Phi_-|}, & \Delta B_r \leq 0 \end{cases}$$

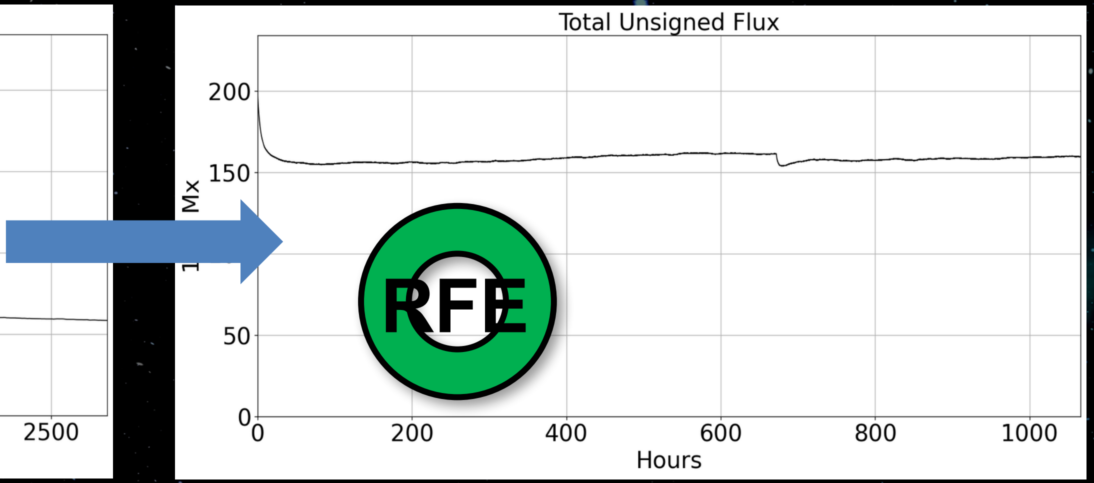
HipFT Model: Random Flux Emergence

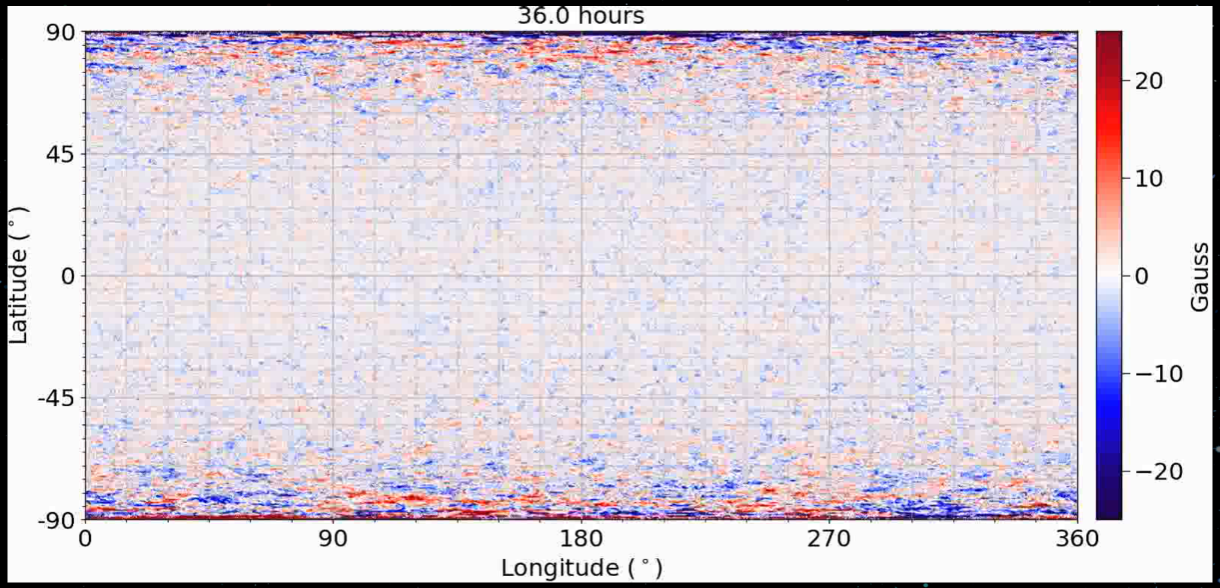
- The flux-canceling processes in SFT models reduces the unsigned flux (UF) compared to that of the assimilated data
- This leads to unrealistic localized low UF regions and a variable average UF away from the assimilation region
- This can adversely affect MHD models that use UF in the heating model
- HipFT can add random flux emergence as a source term
- The parameters can be tuned to yield a constant average UF in the quiet Sun regions calibrated to the time period of interest and resolution of the model

$$B_{r;\mathrm{RFE}} = \frac{|\Phi_{\mathrm{RFE}}|}{N\,A_{\mathrm{cell}}} \frac{1}{\sigma_{\mathrm{RFE}}\sqrt{2\,\pi}} \mathrm{Rnd}(\sigma_{\mathrm{RFE}})$$

$$B_{r;\mathrm{RFE};1} \longleftarrow \boxed{\tau_{\mathrm{RFE}}} \longrightarrow B_{r;\mathrm{RFE};2}$$

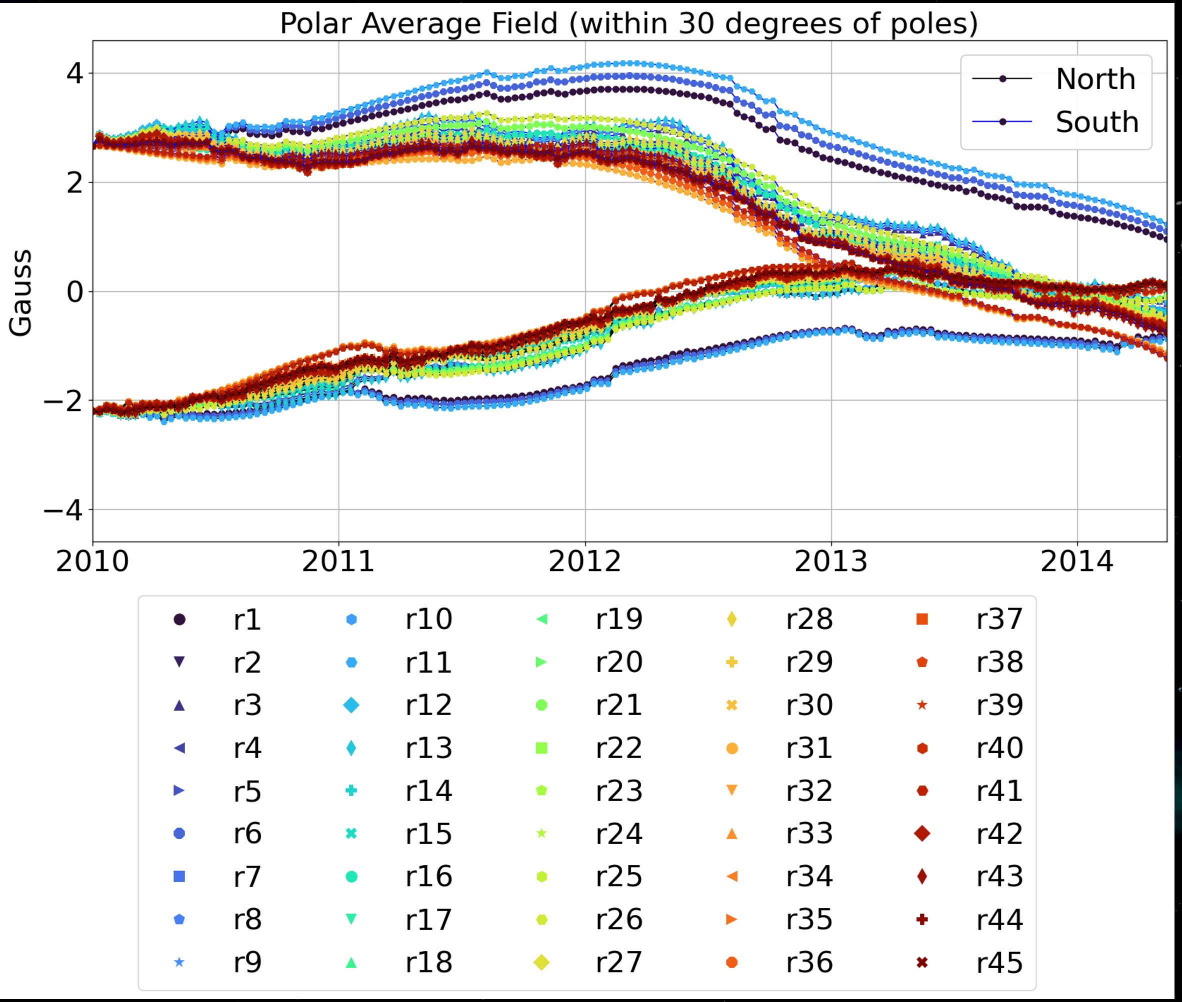


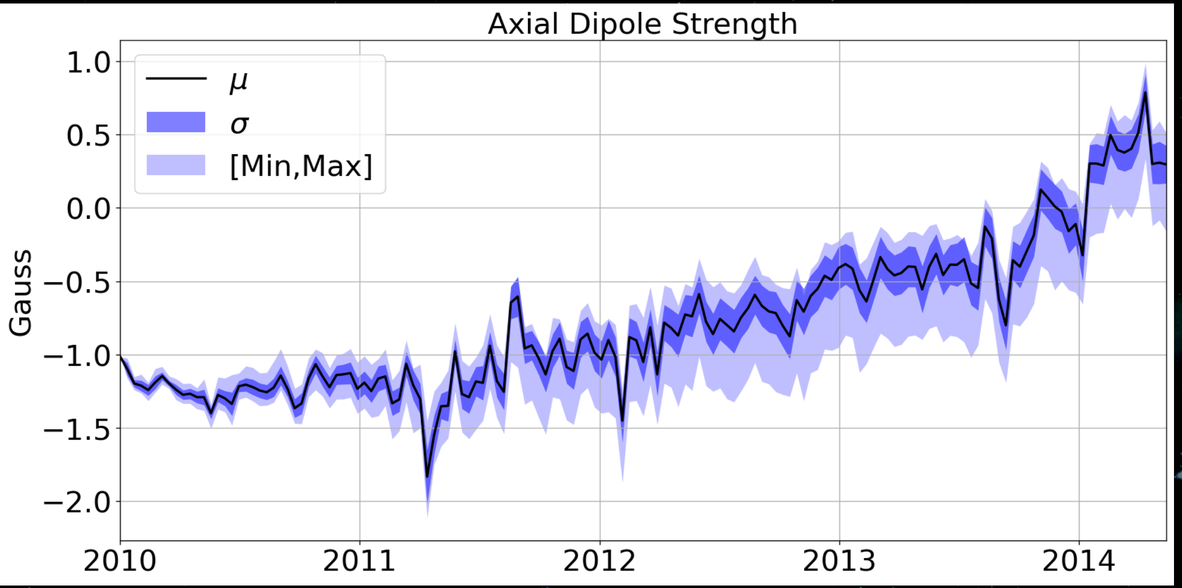




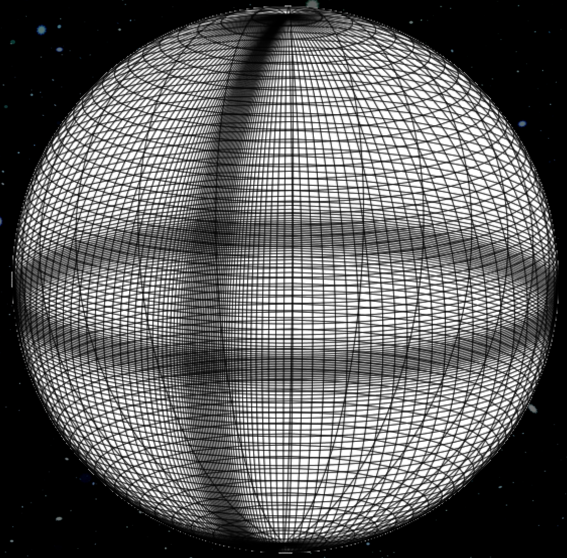
HipFT Model: Multiple Realizations

- Can run multiple realizations simultaneously across many model parameters
- Current cross-realization parameters include diffusion rate, flow profile coefficients, flow attenuation levels, and data assimilation options
- Post processing python scripts are included to analyze results

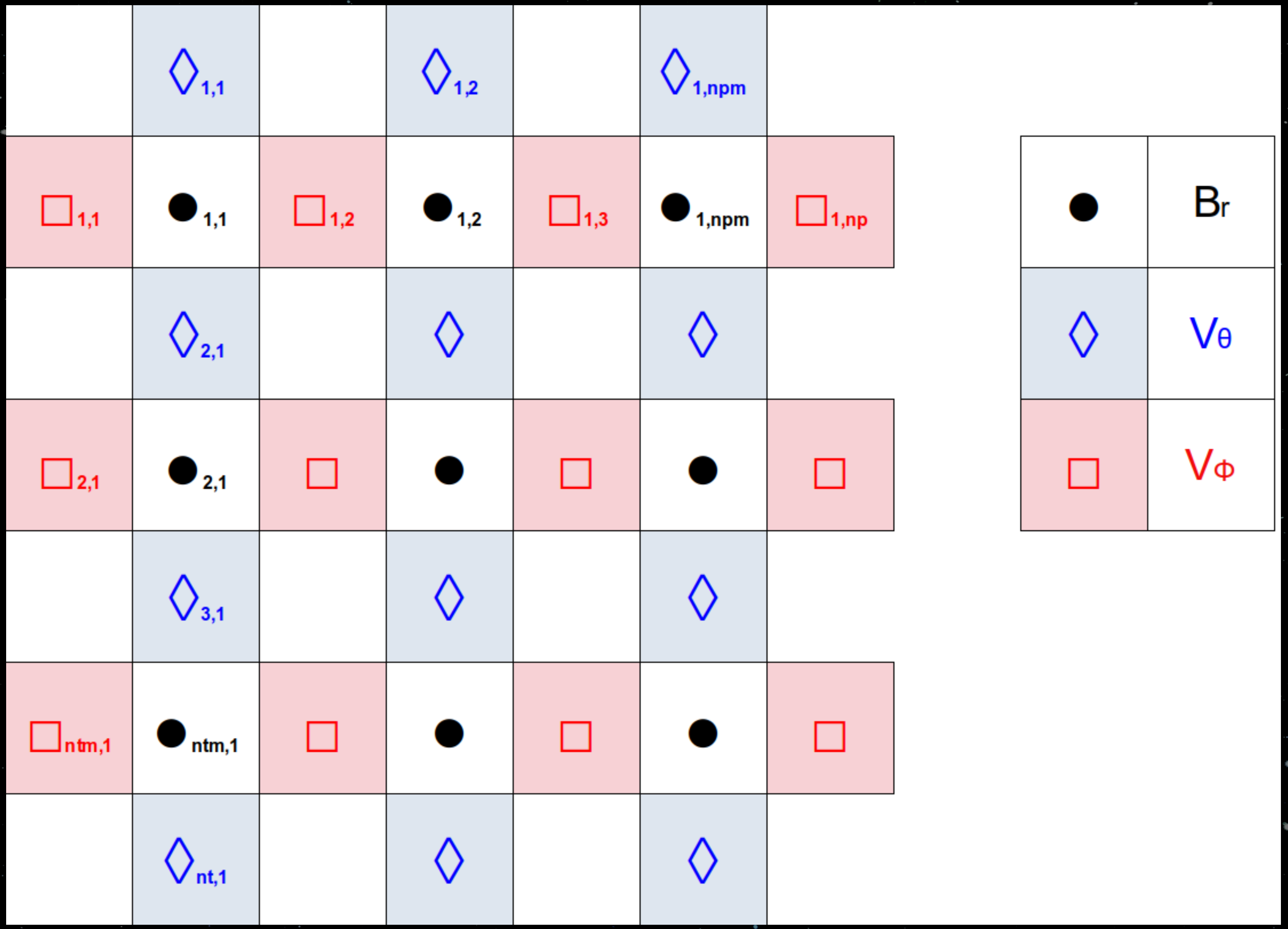


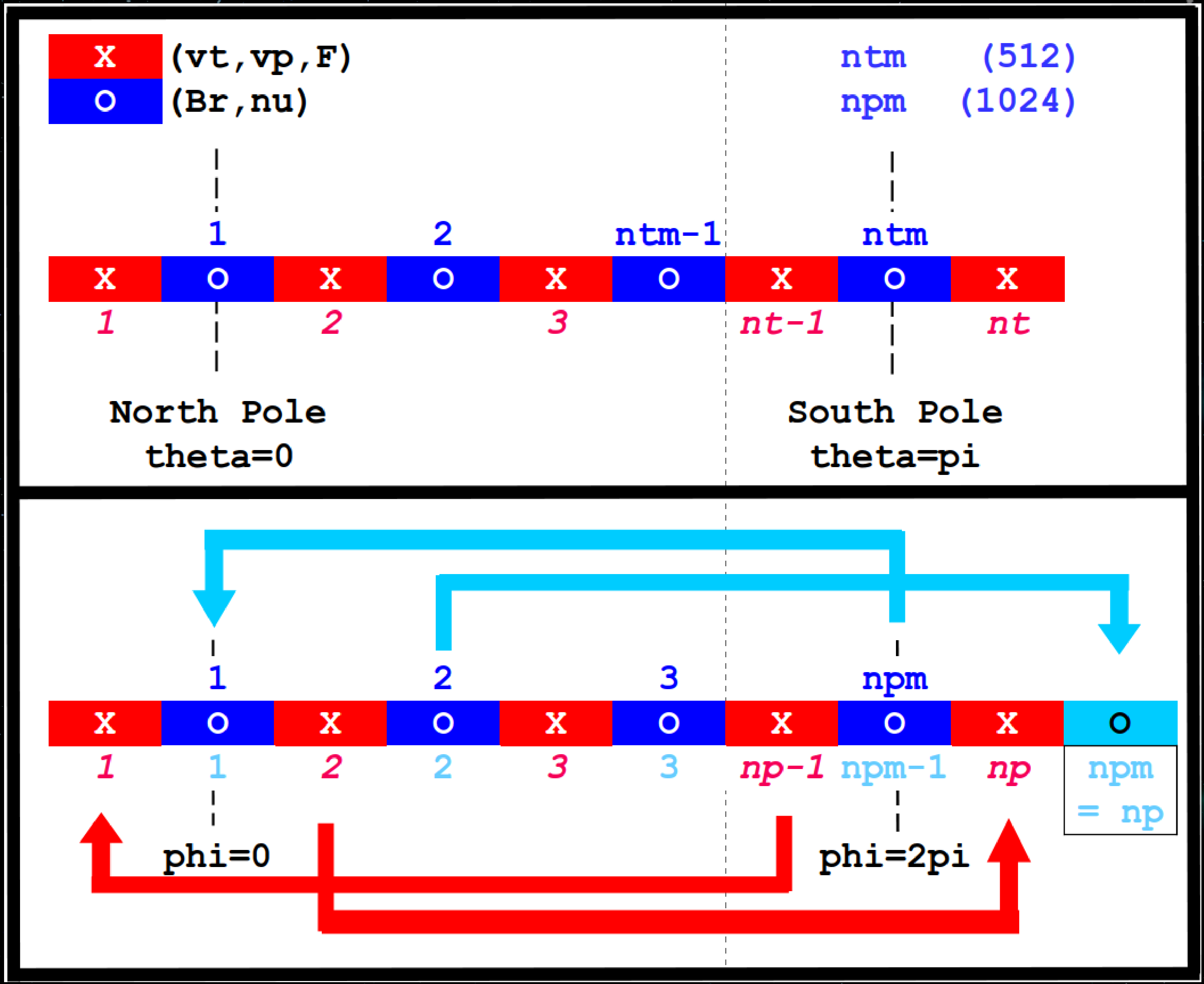


HipFT Numerical Methods: Grid

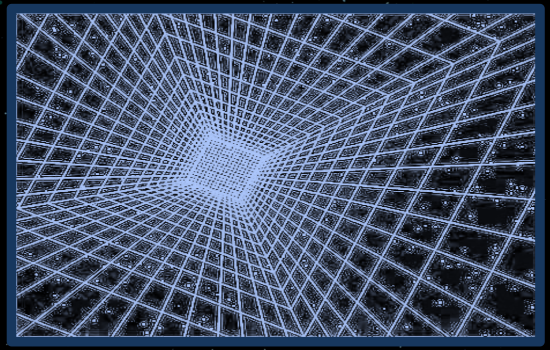


Non-uniform, logically-rectangular spherical surface staggered grid





HipFT Numerical Methods: Advection (Spatial)



$$\nabla_s \cdot (B_r \mathbf{v}) = \frac{1}{R_\odot \sin\theta} \frac{\partial}{\partial \theta}(\sin\theta \, B_r \, v_\theta) + \frac{1}{R_\odot \sin\theta} \frac{\partial}{\partial \phi}(B_r \, v_\phi),$$

$$[\nabla_s \cdot (B_r \mathbf{v}_s)]_{(j,k)} \approx \frac{\sin\theta_{j+\frac{1}{2}} F_{\theta:j+\frac{1}{2},k} - \sin\theta_{j-\frac{1}{2}} F_{\theta,j-\frac{1}{2},k}}{\sin\theta_j \, \Delta\theta_j} + \frac{F_{\phi:j,k+\frac{1}{2}} - F_{\phi:j,k-\frac{1}{2}}}{\sin\theta_j \, \Delta\phi_k}$$

3rd-order WENO3-CS

[Cravero and Semplice, Sci Comput (2016) 67:1219–1246]

$$F_{i-1/2} = u^+_{i-1/2} + u^-_{i-1/2},$$

where $u^+_{i-1/2}$ and $u^-_{i-1/2}$ are the the left and right moving numerical fluxes at the cell boundary respectively. $u^+_{i-1/2}$ and $u^-_{i-1/2}$ are defined as follows:

$$u^\pm_{i-1/2} = \frac{w^\pm_0}{w^\pm_0 + w^\pm_1} p^\pm_0 + \frac{w^\pm_1}{w^\pm_0 + w^\pm_1} p^\pm_1,$$

where the w fractions are the nonlinear weights and p variables are the reconstruction polynomials. The scheme utilizes nonlinear weights based on the smoothness of our function for deciding how to add the reconstruction polynomials together to get $u^+_{i-1/2}$ and $u^-_{i-1/2}$. The WENO3 scheme uses a first-order polynomial approximation, which gives us the following reconstruction polynomials:

$$p^-_0 = (1 + D^{c/cm}_{i-3/2})\,\mathrm{LM}_{i-1} - D^{c/cm}_{i-3/2}\mathrm{LM}_{i-2}, \qquad p^+_0 = (1 + D^{c/cp}_{i})\,\mathrm{LP}_{i} - D^{c/cp}_{i}\mathrm{LP}_{i+1},$$

$$p^-_1 = D^{c/cm}_{i}\,\mathrm{LM}_{i-1} + D^{c/cp}_{i-3/2}\mathrm{LM}_{i}, \qquad p^+_1 = D^{c/cp}_{i-3/2}\,\mathrm{LP}_{i} + D^{c/cm}_{i}\mathrm{LP}_{i-1},$$

where D are calculation constants and LP_i and LM_i are the left and right moving fluxes respectively. The components of the nonlinear weights are defined as:

$$w^-_0 = \frac{D^{p/t}_{i-3/2}}{(\epsilon_w + \beta^-_0)^2}, \qquad w^-_1 = \frac{D^{cm/t}_{i-3/2}}{(\epsilon_w + \beta^-_1)^2}, \qquad w^+_0 = \frac{D^{m/t}_{i-1/2}}{(\epsilon_w + \beta^+_0)^2}, \qquad w^+_1 = \frac{D^{cp/t}_{i-1/2}}{(\epsilon_w + \beta^+_1)^2}, \qquad \text{(B1)}$$

where the βs are called 'smoothness indicators' which describes the smoothness of the region and ϵ_w is a tiny value to avoid division by zero. The value of ϵ_w is often set to a value smaller than the typical solution values (Shu 2009). However, [RC] [REF] showed that using the cell spacing for ϵ_w can avoid a huge drop in accuracy near critical points, keeping the scheme 3rd-order:

$$\epsilon_w = \Delta x_i,$$

where Δx_i is the local cell spacing (we therefore refer to the overall scheme as 'WENO-CS' in referfence to [RC] [REF]). The βs are defined as:

$$\beta^-_0 = 4\,(D^{c/cm}_{i-3/2}\,(\mathrm{LM}_{i-1} - \mathrm{LM}_{i-2}))^2, \qquad \beta^+_0 = 4\,(D^{c/cp}_{i}\,(\mathrm{LP}_{i+1} - \mathrm{LP}_{i}))^2,$$

$$\beta^-_1 = 4\,(D^{c/cp}_{i-3/2}\,(\mathrm{LM}_{i} - \mathrm{LM}_{i-1}))^2, \qquad \beta^+_1 = 4\,(D^{c/cm}_{i}\,(\mathrm{LP}_{i} - \mathrm{LP}_{i-1}))^2,$$

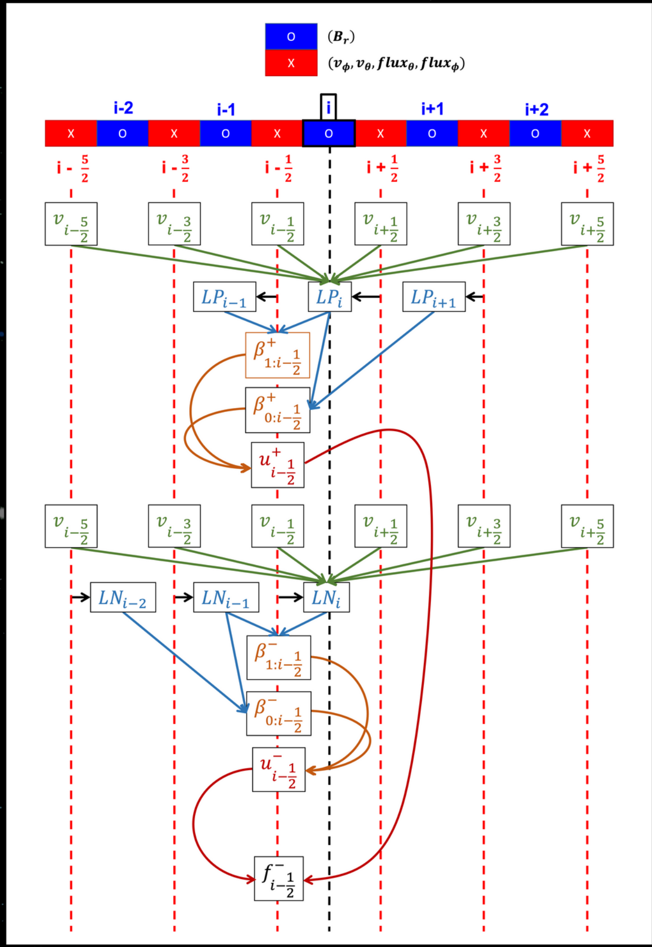
where because of the non-uniform grid the following equations are used for the values of D instead of the normal constants (Smit et al. 2005)

$$D^{c/cp}_{i-1/2} = \frac{\Delta x_i}{\Delta x_i + \Delta x_{i+1}}, \qquad D^{p/t}_{i-1/2} = \frac{\Delta x_{i+1}}{\Delta x_{i-1} + \Delta x_i + \Delta x_{i+1}}, \qquad D^{cp/t}_{i-1/2} = \frac{\Delta x_i + \Delta x_{i+1}}{\Delta x_{i-1} + \Delta x_i + \Delta x_{i+1}},$$

$$D^{c/cm}_{i-1/2} = \frac{\Delta x_i}{\Delta x_i + \Delta x_{i-1}}, \qquad D^{m/t}_{i-1/2} = \frac{\Delta x_{i-1}}{\Delta x_{i-1} + \Delta x_i + \Delta x_{i+1}}, \qquad D^{cm/t}_{i-1/2} = \frac{\Delta x_i + \Delta x_{i-1}}{\Delta x_{i-1} + \Delta x_i + \Delta x_{i+1}},$$

Fig. 3 shows the D_i constants on the grid where the subsection of the line marked with an x denotes the numerator. To obtain the values of LP_i and LM_i, Local Lax-Friedrichs flux splitting [Shu (2009)] is used:

$$\mathrm{LP}_i = \frac{1}{2} B_{r:i}\,(v_{i+1/2} - \alpha_i), \qquad \mathrm{LM}_i = \frac{1}{2} B_{r:i}\,(v_{i-1/2} + \alpha_i), \qquad \text{(B2)}$$



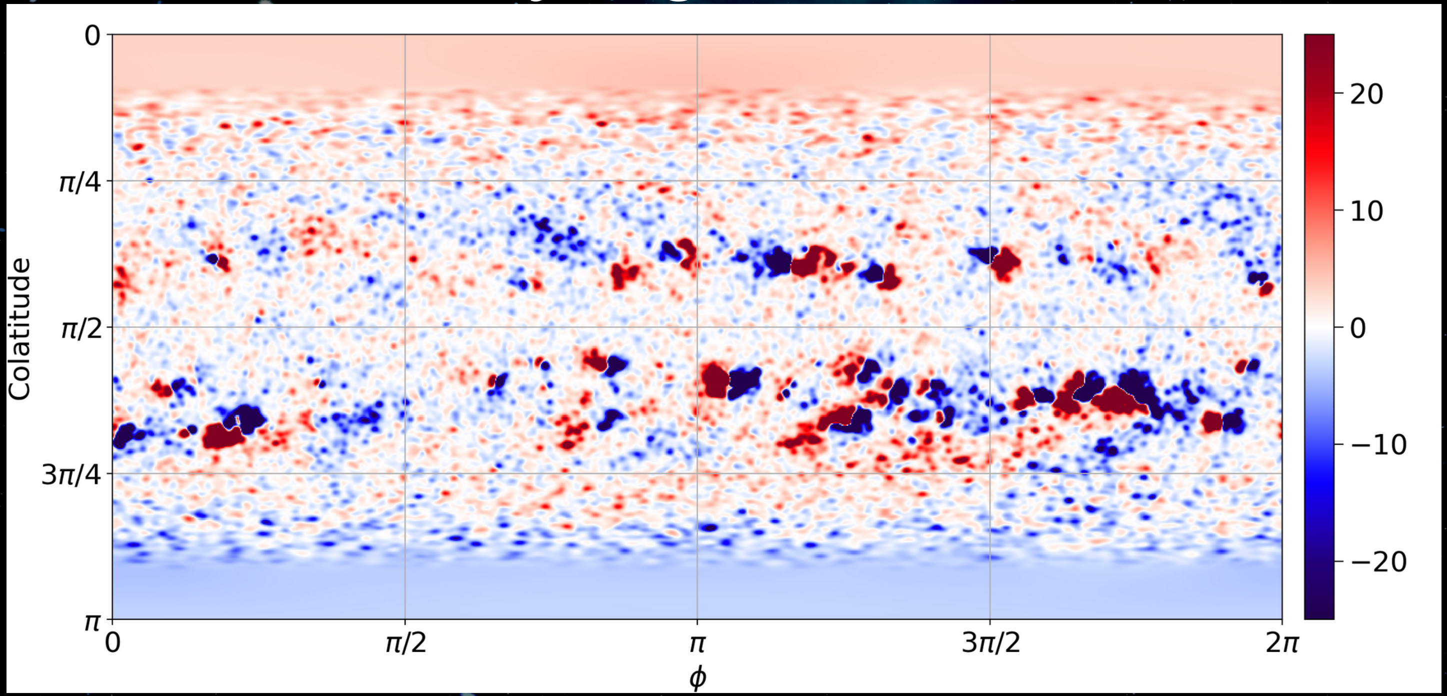
1st-order Upwind

$$F_{i-1/2} = v_{i-\frac{1}{2}} \frac{1}{2} \left[(1 - \mathrm{uw})\, B_{r:i} + (1 + \mathrm{uw})\, B_{r:i-1} \right]$$

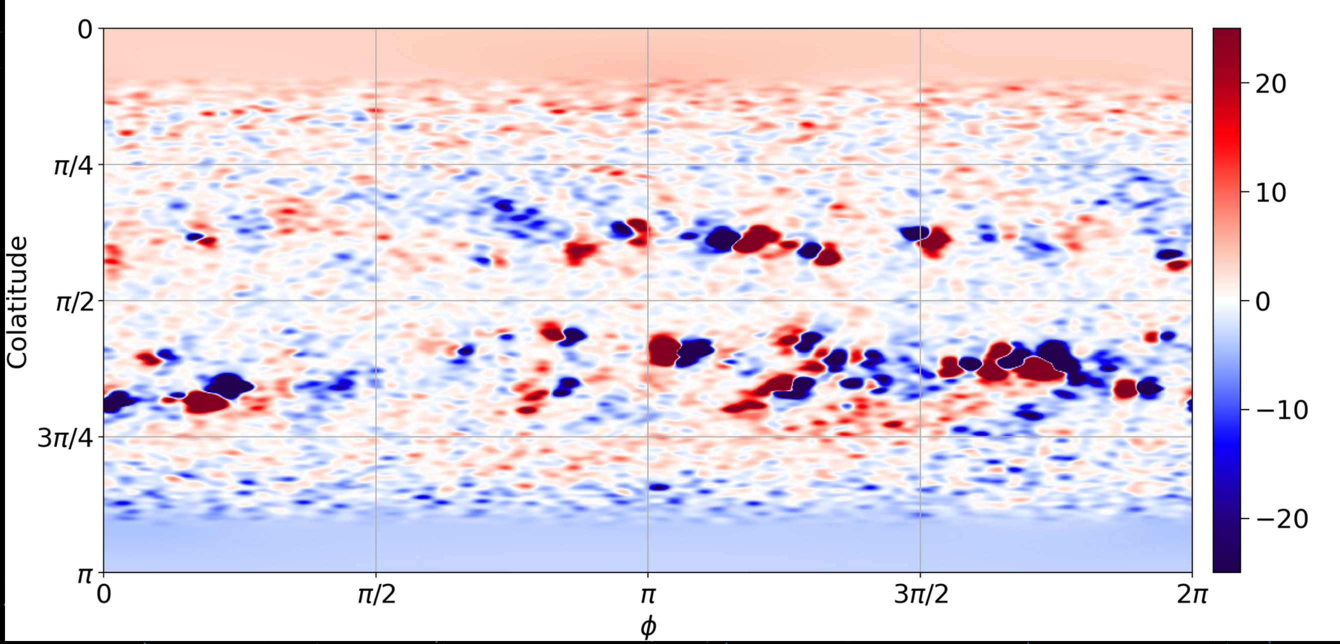
$$\mathrm{uw} = \alpha_{\mathrm{uw}} \, \mathrm{sign}(v_{i-\frac{1}{2}}),$$

HipFT Numerical Methods: Advection (Spatial)

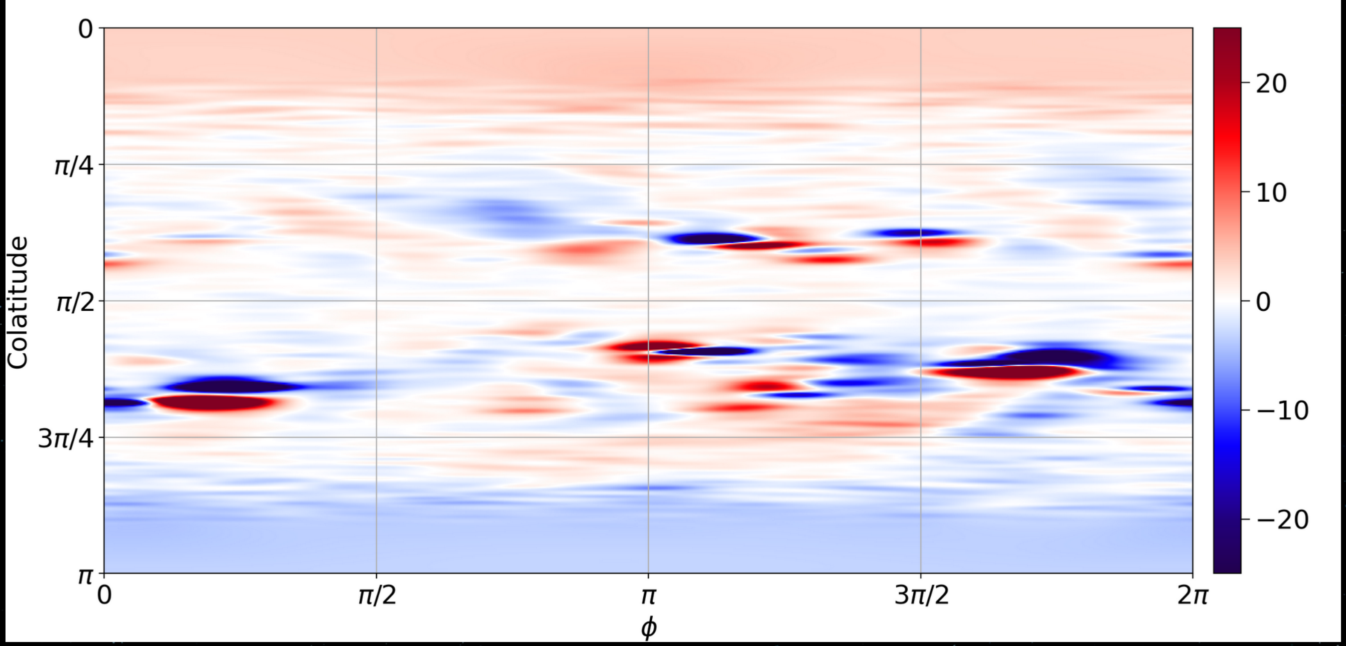
Advection-only, rigid rotation for 1 CR



3rd-order WENO3-CS



1st-order Upwind



HipFT Numerical Methods: Advection (Time)

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{F}(\mathbf{u})$$

3rd-order Strong Stability Preserving Runge-Kutta (4-stage)

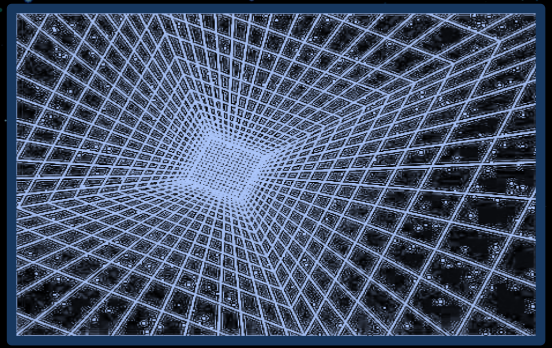
Program 6.3 (Low-storage SSPRK(4,3)).

```
q1 = u;
q2 = q1 + dt/2 * F(q1);
q2 = q2 + dt/2 * F(q2)
q2 = 2/3 * q1 + 1/3 * (q2 + dt/2 * F(q2))
q2 = q2 + dt/2 * F(q2)
u = q2
```

4-stage version allows a stable time step twice as large as the 3-stage version!

$$\Delta t < \frac{1}{2}\left[\frac{|v_\theta|}{\Delta\theta} + \frac{|v_\phi|}{\sin\theta\,\Delta\phi}\right]^{-1} \times 2$$

HipFT Numerical Methods: Diffusion (Spatial)



$$\nabla_s \cdot (\nu \nabla_s B_r) = \frac{1}{R_\odot^2 \sin\theta} \frac{\partial}{\partial \theta} \left(\nu(\theta, \phi, B_r) \sin\theta \frac{\partial B_r}{\partial \theta} \right) + \frac{1}{R_\odot^2 \sin^2\theta} \frac{\partial}{\partial \phi} \left(\nu(\theta, \phi, B_r) \frac{\partial B_r}{\partial \phi} \right)$$

$$\nabla_s \cdot (\nu(\theta,\phi) \nabla_s B_r) \approx \frac{1}{\sin\theta_j \Delta\theta_j} \left[\nu_{j+\frac{1}{2},k} \sin\theta_{j+\frac{1}{2}} \frac{B_{r:j+1,k} - B_{r:j,k}}{\Delta\theta_{j+\frac{1}{2}}} - \nu_{j-\frac{1}{2},k} \sin\theta_{j-\frac{1}{2}} \frac{B_{r:j,k} - B_{r:j-1,k}}{\Delta\theta_{j-\frac{1}{2}}} \right] + \frac{1}{\sin^2\theta_j \Delta\phi_k} \left[\nu_{j,k+\frac{1}{2}} \frac{B_{r:j,k+1} - B_{r:j,k}}{\Delta\phi_{k+\frac{1}{2}}} - \nu_{j,k-\frac{1}{2}} \frac{B_{r:j,k} - B_{r:j,k-1}}{\Delta\phi_{k-\frac{1}{2}}} \right].$$

2nd-order Central Difference

HipFT Numerical Methods: Diffusion (Time)

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{F}(\mathbf{u})$$

2nd-order Runge-Kutta-Gegenbauer Super Time-Stepping

[Skaras et. al (2021) J. of Comp Phys 425 109879]

Explicit and unconditionally stable!

$$s = \left\lceil \frac{1}{2}\sqrt{25 + 24\,\frac{\Delta t}{\Delta t_{\text{Euler}}}} - \frac{3}{2} \right\rceil$$

$$w = \frac{6}{(s+4)\,(s-1)}, \quad b_0 = 1,\ b_1 = \tfrac{1}{3},\ b_2 = \tfrac{1}{15}, \quad \mu_1 = 1,$$
$$\tilde{\mu}_1 = w, \quad \mu_2 = \frac{1}{2}, \quad \nu_2 = -\frac{1}{10}, \quad \tilde{\mu}_2 = \mu_2\, w, \quad \gamma_2 = 0.$$

$$u_0 = u^n$$
$$y_0 = F(u_0)$$
$$u_1 = u_0 + \tilde{\mu}_1\,\Delta t\, y_0$$

do k = 2 : s

$$u_k = \mu_k\, u_{k-1} + \nu_k\, u_{k-2} + (1 - \mu_k - \nu_k)\, u_0 + \tilde{\mu}_k\,\Delta t\, F(u_{k-1}) + \gamma_k\,\Delta t\, y_0$$

enddo

$$u^{n+1} = u_s,$$

do k = 3 : s

$$b_k = \frac{4\,(k-1)\,(k+4)}{3\,k\,(k+1)\,(k+2)\,(k+3)}$$
$$\mu_k = \left(2 + \frac{1}{k}\right)\frac{b_k}{b_{k-1}}$$
$$\nu_k = -\left(\frac{1}{k} + 1\right)\frac{b_k}{b_{k-2}}$$
$$\tilde{\mu}_k = \mu_k\, w$$
$$\gamma_k = \left(\frac{k\,(k+1)}{2}\, b_{k-1} - 1\right)\tilde{\mu}_k,$$

enddo

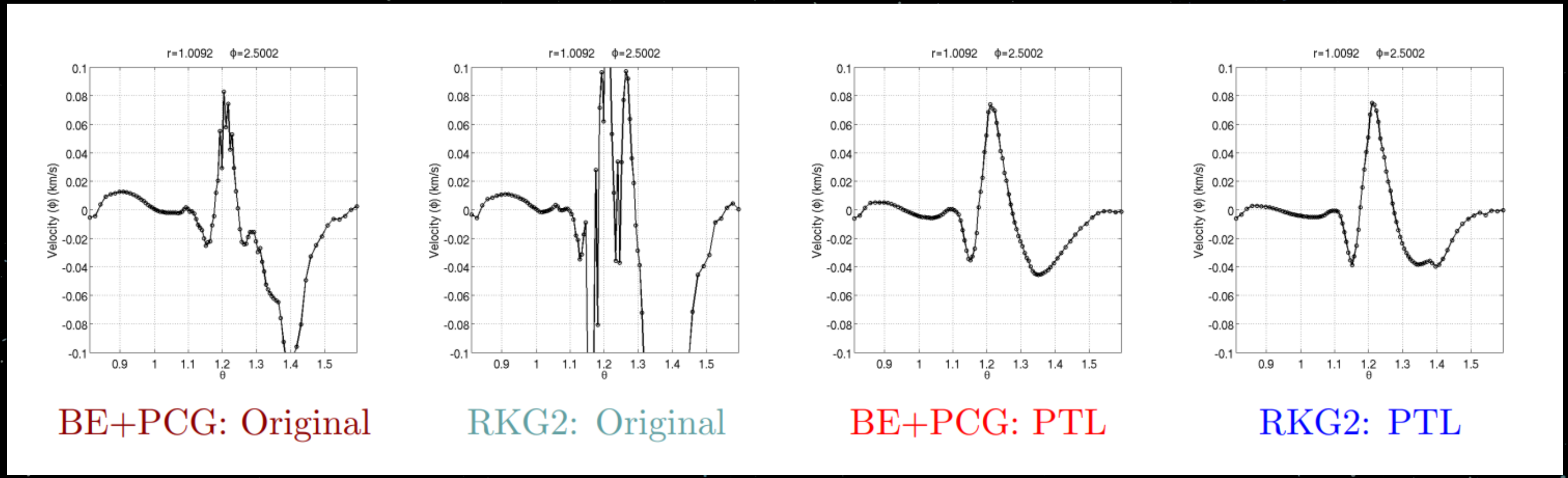
HipFT Numerical Methods: Diffusion (Time)

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{F}(\mathbf{u})$$

Practical Time Step Limit (PTL)

- When using unconditionally stable schemes for diffusion, taking too large a time step can causes issues in the solution (such as not damping oscillations, or causing new ones)

- To avoid this, we use a PTL time step limit defined as:



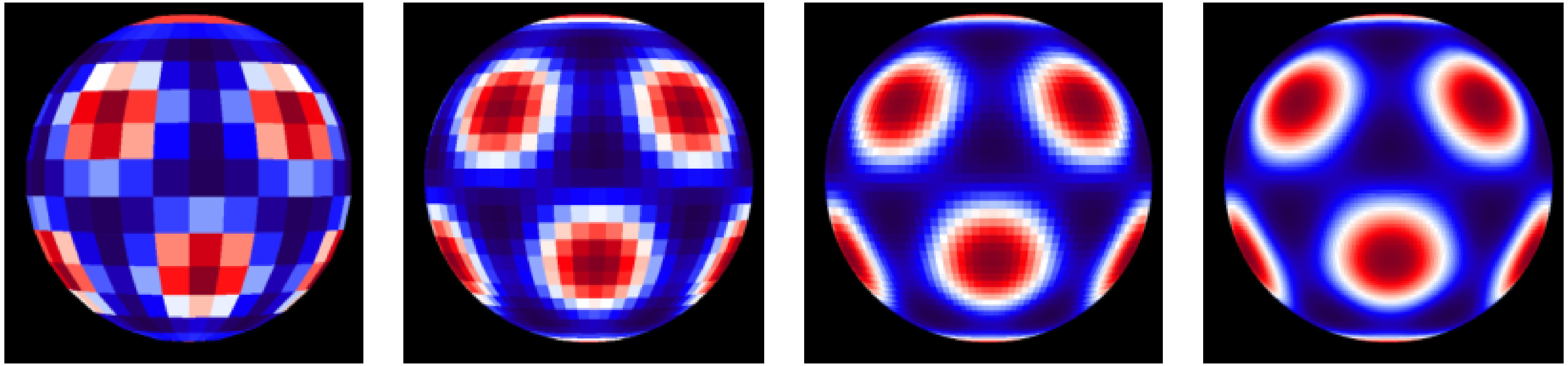
$$\Delta t_{\mathrm{PTL}} \leq \min\left[-\frac{u^n_{\vec{k}} - u^n_{\vec{k}+\vec{i}}}{F_{\vec{k}}(u^n) - F_{\vec{k}+\vec{i}}(u^n)}\right]$$

[Caplan et al (2024) J. Phys.: Conf. Ser. 2742 012020]

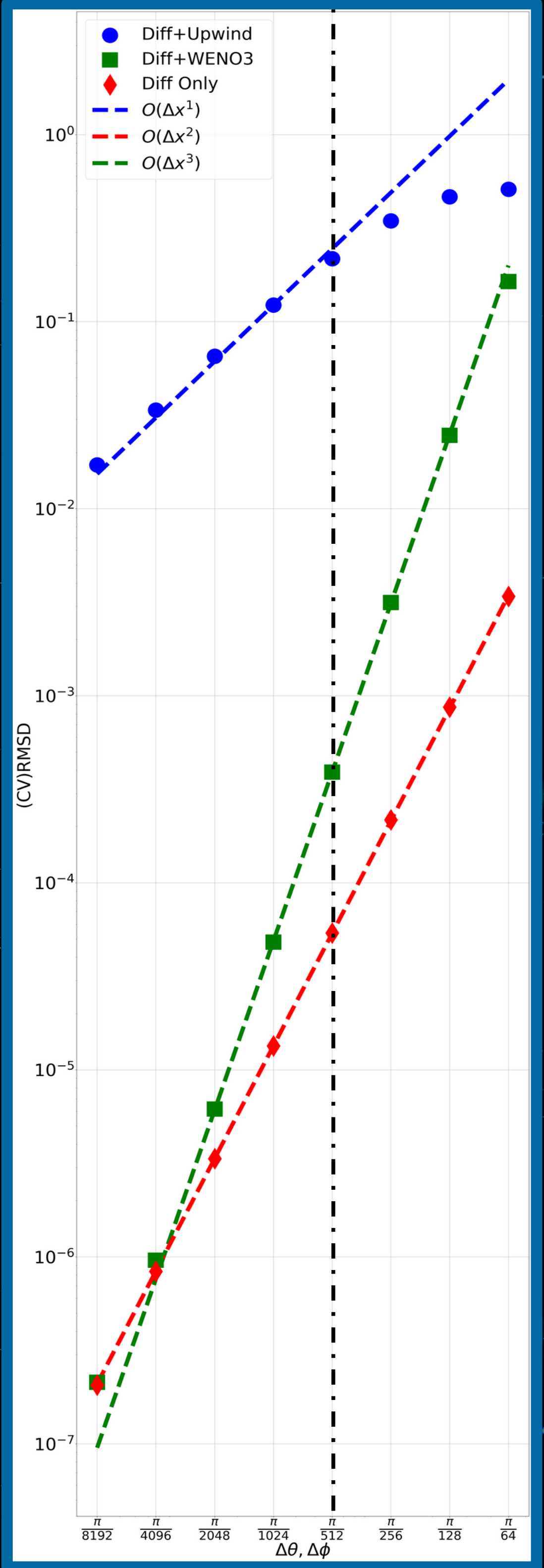
HipFT Numerical Methods: Validation

We use an analytic time-dependent diffusion solution combined with a rigid rotational velocity for 1 rotation:

$$u(\theta,\phi,t) = 1000\,e^{-42\,\nu\,t}\left(Y_6^0(\theta,\phi)+\sqrt{\frac{14}{11}}\,Y_6^5(\theta,\phi)\right) \qquad v_\phi = \Omega\,\sin\theta$$



$\Delta\theta,\Delta\phi=\pi/16$ | $\Delta\theta,\Delta\phi=\pi/32$ | $\Delta\theta,\Delta\phi=\pi/64$ | $\Delta\theta,\Delta\phi=\pi/128$



HipFT Code Implementation: Parallelism

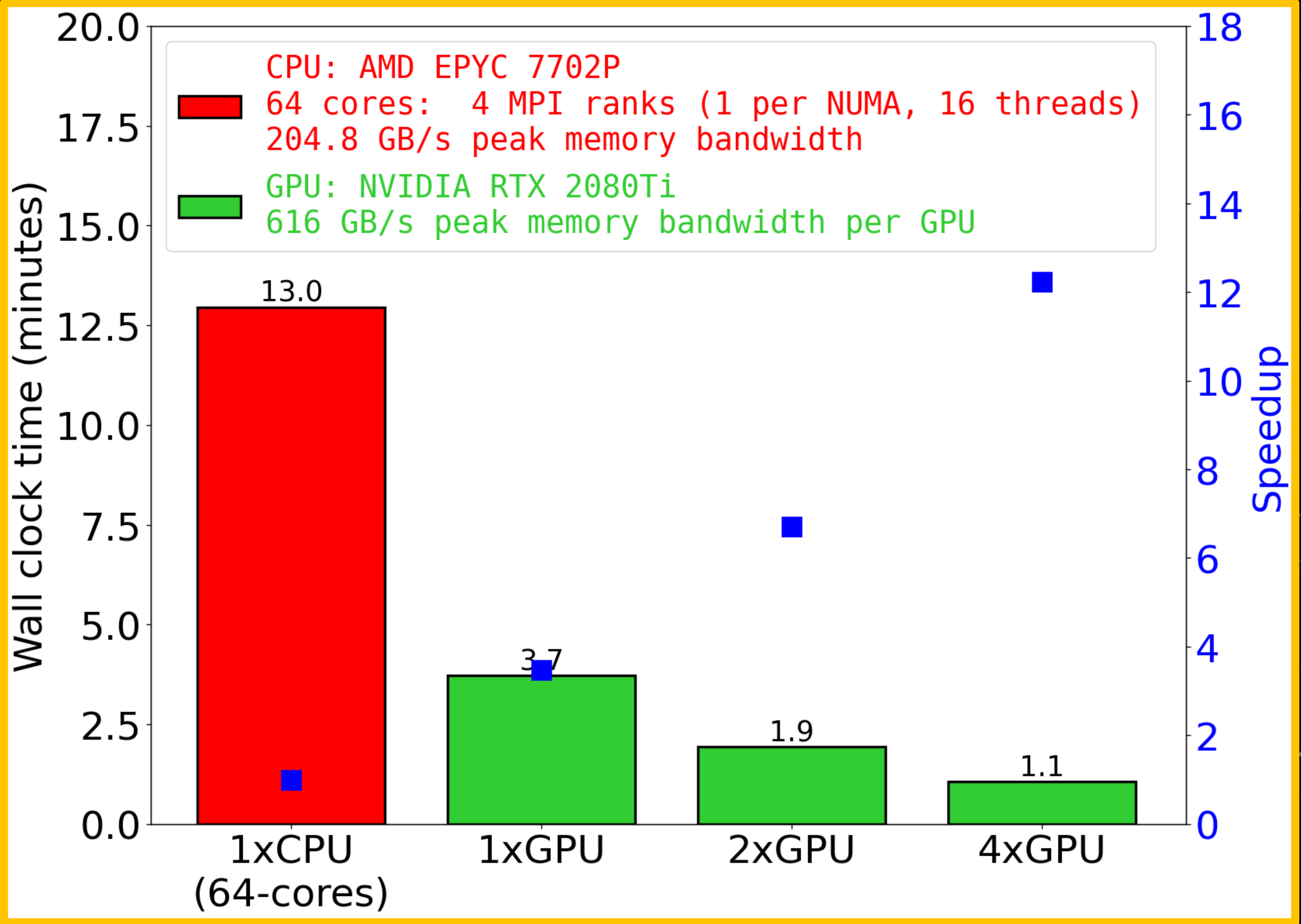
- Written in Fortran 2023
- Uses standard parallelism (do concurrent) for GPU offload or multi-threaded CPU parallelism
- Uses OpenMP Target directives for CPU-GPU data movement
- Uses MPI to parallelize across realizations (multi-GPU, multi-node)

```
do concurrent (i=1:N,j=1:M)
  Computation
enddo
```

```
!$omp target enter data map(to:a)
!$omp target exit data map(from:a)
```

HipFT Code Implementation: Parallelism

In-house workstation:
EPYC 7702P 64-core CPU
Four RTX 2080Ti GPUs

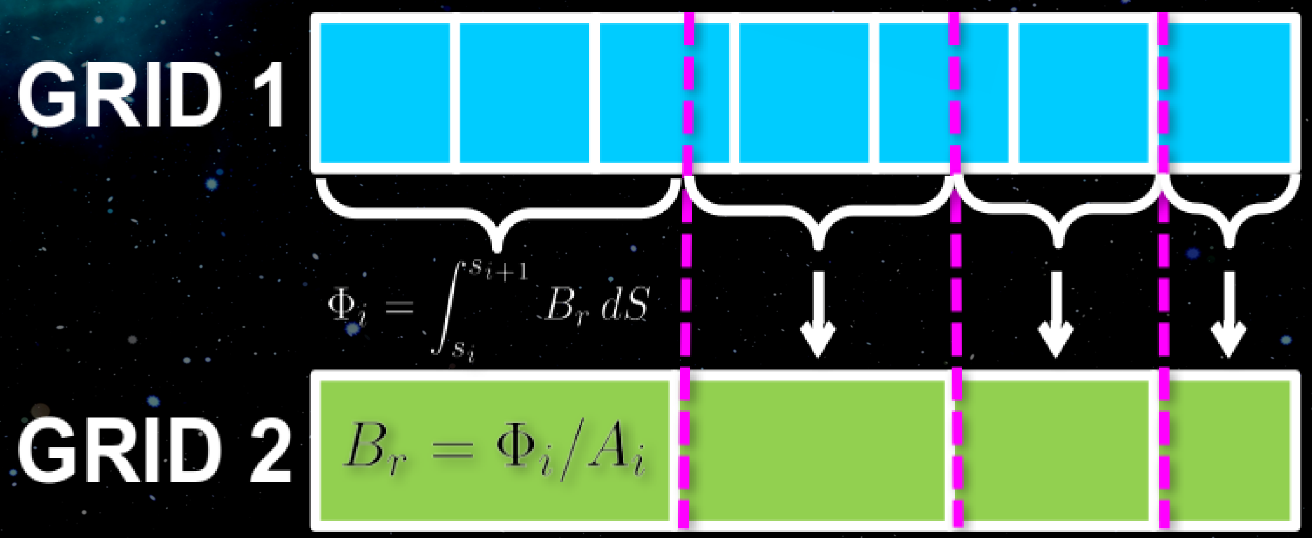


Test: 28-day run at 1024x512 with analytic flow models and diffusion. Eight realizations spanning various levels of diffusion and flow attenuation

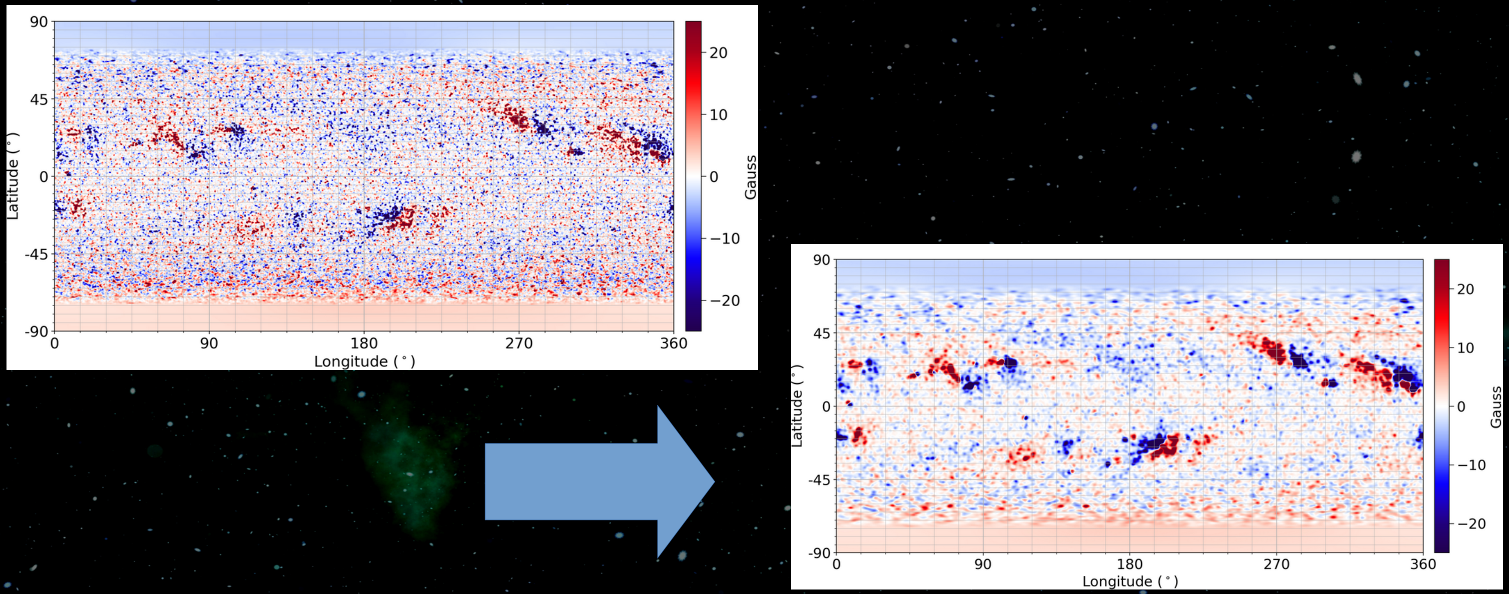
OFT: Map Processing

IMPORTANT

- The OFT maps are 1024x512 by default, which is higher than we want to model
- A flux-preserving re-mapping is done to get to the desired resolution
- In case the map was not flux balanced in HipFT, apply flux balancing
- Finally, to make the structures in the map resolvable, we smooth the map by applying diffusion
- The last two steps are performed using a tiny run of HipFT



$$B_r^* = \begin{cases} B_r / \sqrt{|\Phi_+ / \Phi_-|} & \text{if } B_r > 0 \\ B_r \sqrt{|\Phi_+ / \Phi_-|} & \text{if } B_r \leq 0 \end{cases}$$




BREAK TIME!!!


Running OFT: Zenodo Package for MagMAP and ConFlow Output

- Since MagMAP and ConFlow are not yet released on github, we provide output data sets for use with HipFT on Zenodo
- A full year of MagMAP processed HMI magnetograms for 2022
- A 28-day long sequence of super granular convective flow maps
- The full dataset is ~50GB
- For this workshop, I have prepared a smaller (~12GB) dataset that has only 1 month of MagMAP data
- I have this dataset on a USB thumb drive to avoid needing to download it
- It is also on the UAH server (bladerunner)

zenodo.org/records/11205509

Running HipFT: Input file / Namelist

- Sample full namelist input file with descriptions of all inputs and their default values: **`hipft/doc/hipft.in.documentation`**

```
&hipft_input_parameters
!
  verbose = 0
!
!       -----> Set this to output a info for use with debugging.
!       -----> The higher the integer, the more info (2 is max as of now).
!
  res_nt = 0
  res_np = 0
!
!       -----> Resolution in theta and phi on uniform grid.
!       -----> These are automatically set when reading in an initial map with "initial_map_filename"
!       -----> They are currently only used for validation runs.
!
  n_realizations = 1
!
!       -----> Set number of realizations.
!
  initial_map_filename = ''
!
!       -----> Initial map
!       -----> The resolution and grid of the run is determined by this input map.
!       -----> This means all data assimilation maps must match the grid of the input map,
!       -----> and input flows must match the correct staggered velocity meshes that correspond
!       -----> to the input map grid.
!
  initial_map_flux_balance = .false.
!
!       -----> Toggle to flux balance (multiplicitive) the initial map.
!
```

Running HipFT: Command line

Need this with OpenMPI+gfortran to make sure all threads from **`-ftree-parallelize-loops`** are spread across all cores

```
nohup mpirun -bind-to none -np 1 hipft 1>hipft.log 2>hipft.err &
```

```
nohup mpirun -np 1 hipft 1>hipft.log 2>hipft.err &
```

Running HipFT: Outputs

- Multiple output files:
  - **hipft_history_num_r000001.out**
    - Time histories of numerical method quantities
  - **hipft_history_sol_r000001.out**
    - Time histories of physical quantities
  - **hipft_output_map_list.out**
    - List of output maps and their output times
  - **hipft_run_parameters_used.out**
    - Namelist dump of ALL input parameters
  - **hipft_timing.out**
    - Timing information about the run

Running HipFT: Post Processing Scripts

hipft_add_dates_to_map_output_list.py
- Associate dates with map output times

hipft_make_plots_and_movies.py
- Plot all output maps and make a movie of them

hipft_make_butterfly_diagram.py
- Make a butterfly diagram data set from a run

hipft_plot_butterfly_diagram.py
- Plot the butterfly diagram data

hipft_combine_run_histories.py
- Combine several HipFT run histories into one file

hipft_plot_histories.py
- Plot one (or compare multiple) run histories

hipft_compare_run_diags.py
- Compare HipFT runs to each other

hipft_print_history_summary.py
- Summarize history quantities

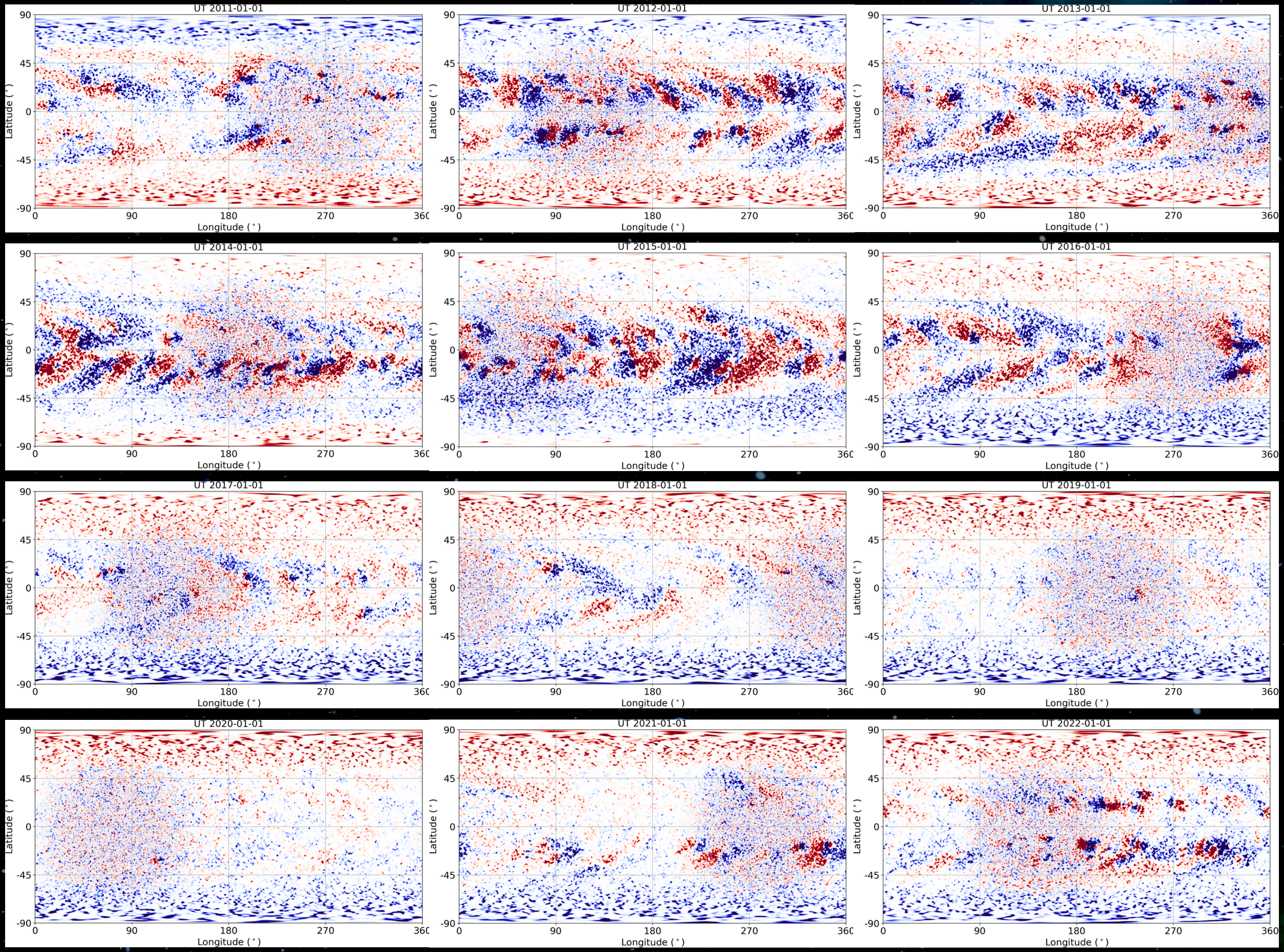
hipft_extract_realization.py
- Extract a single realization from a multi-realization HipFT output map

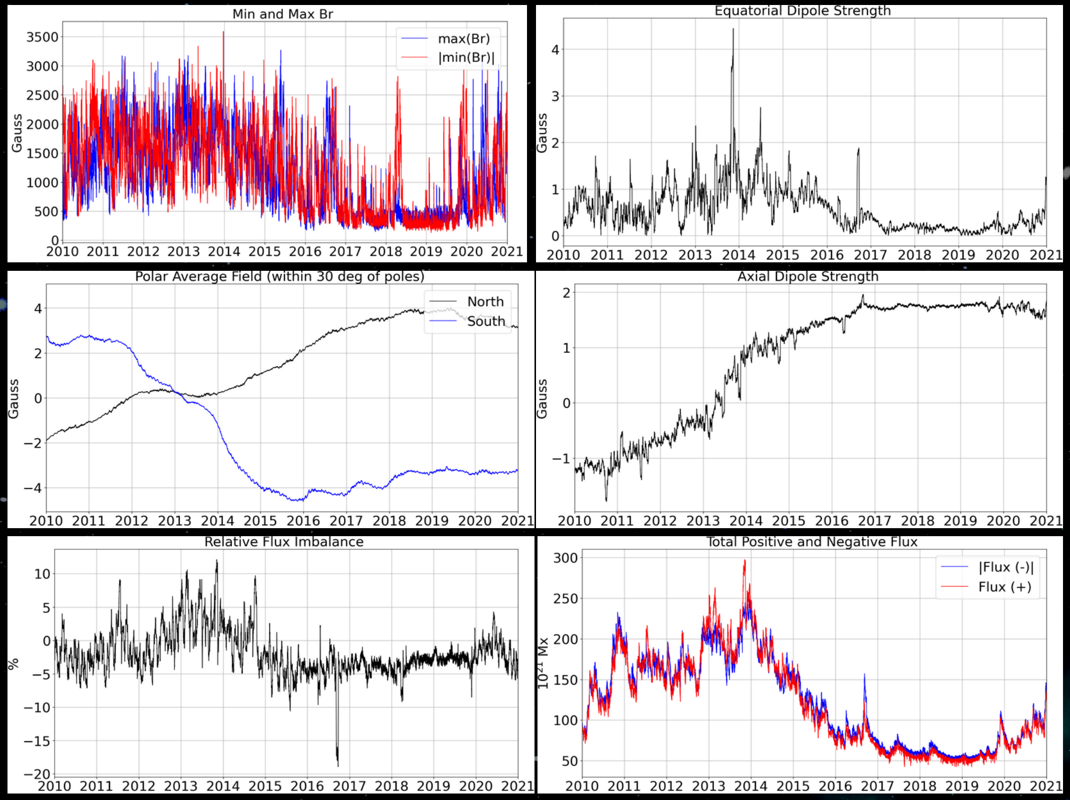
plot2d
- Plot a HipFT map

hipft_get_histories_from_files.py
- Generate HipFT history diagnostics from a sequence of output maps

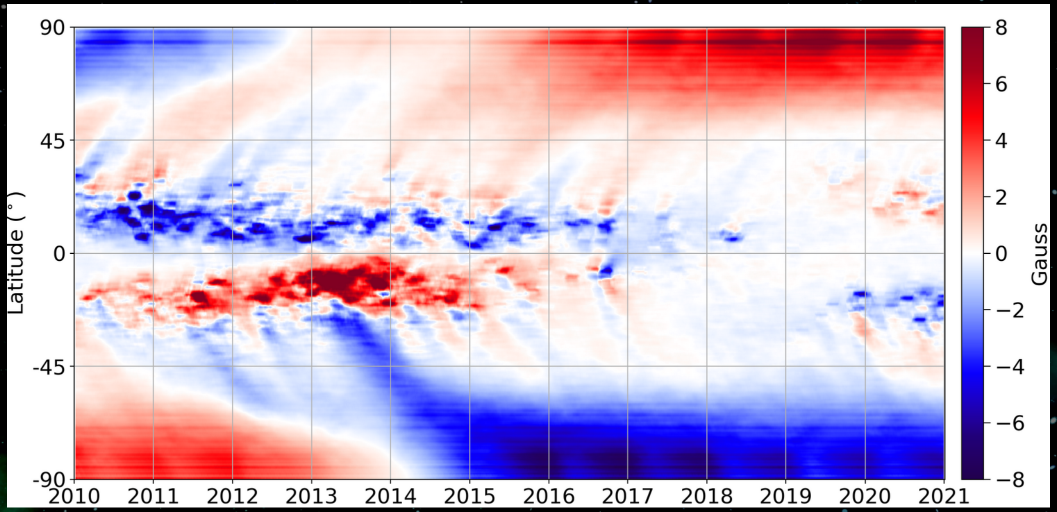
hipft_clear_run.sh
- Clear out a HipFT run

Running HipFT: Example 11yr Run

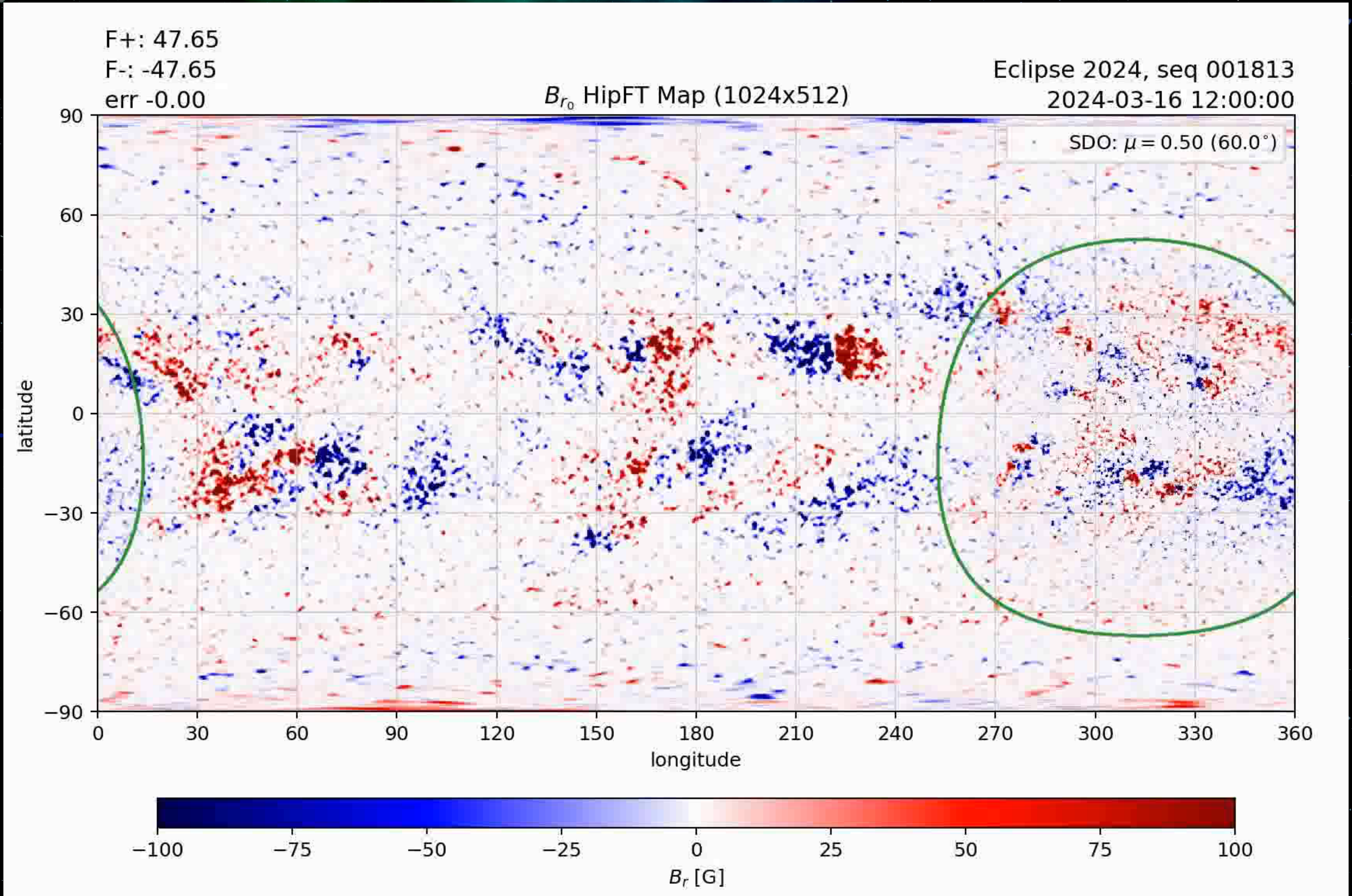




Butterfly Diagram (1CR average)



Running HipFT: Example: 2024 Eclipse



Running HipFT: Examples and Testsuite

In `hipft/examples`:

```
flux_transport_1rot_flowAa_diff_r8/
flux_transport_1yr_flowCAa_diff300_assimdata_rfe/
smooth_magnetogram/
```

In `hipft/testsuite`:

```
advect_gaussians_phi/
advect_gaussians_theta/
diffuse_advect_soccer/
diffuse_soccer/
advect_gaussians_phi_theta/
diffuse_advect_atten_map_1cr/
diffuse_dipole/
run_test_suite.sh
```

OFT: Map Processing

oft/bin/

psi_map_prep.py

- Process map with remapping, flux balancing, and smoothing (uses HipFT)

psi_remap_mm.py

- Remap a map from one resolution to a lower one

prep_multiple_maps.py

- Run psi_map_prep.py on a folder of maps

OFT: Installation

- Installation guides for:
  - Linux
  - Mac
  - Windows with WSL

OFT Assignment

- Run the 72-hour HipFT run with provided MagMAP and ConFlow data, and process the output maps

predsci.com/~caplanr/swqu_workshop


BREAK TIME!!!