Draft version January 14, 2025
Typeset using LaTeX default style in AASTeX63

Open-source Flux Transport (OFT). I. HipFT – High-performance Flux Transport

Ronald M. Caplan,[1] Miko M. Stulajter,[1] Jon A. Linker,[1] Cooper Downs,[1] Lisa A. Upton,[2] Raphael Attie,[3] Charles N. Arge,[3] and Carl J. Henney[4]

[1]Predictive Science Inc., 9990 Mesa Rim Road, Suite 170, San Diego, CA 92121, USA
[2]Southwest Research Institute, 6220 Culebra Road, San Antonio, TX 78238, USA
[3]NASA Goddard Space Flight Center, 8800 Greenbelt Road, Greenbelt, MD 20771, USA
[4]Air Force Research Laboratory, Space Vehicles Directorate, Kirtland AFB, NM 87117, USA

(Dated: January 14, 2025)

ABSTRACT

Global solar photospheric magnetic maps play a critical role in solar and heliospheric physics research. Routine magnetograph measurements of the field occur only along the Sun-Earth line, leaving the far-side of the Sun unobserved. Surface Flux Transport (SFT) models attempt to mitigate this by modeling the surface evolution of the field. While such models have long been established in the community (with several releasing public full-Sun maps), none are open source. The Open Source Flux Transport (OFT) model seeks to fill this gap by providing an open and user-extensible SFT model that also builds on the knowledge of previous models with updated numerical and data acquisition/assimilation methods along with additional user-defined features. In this first of a series of papers on OFT, we introduce its computational core: the High-performance Flux Transport (HipFT) code (github.com/predsci/hipft). HipFT implements advection, diffusion, and data assimilation in a modular design that supports a variety of flow models and options. It can compute multiple realizations in a single run across model parameters to create ensembles of maps for uncertainty quantification and is high-performance through the use of multi-CPU and multi-GPU parallelism. HipFT is designed to enable users to easily write extensions, enhancing its flexibility and adaptability. We describe HipFT's model features, validations of its numerical methods, performance of its parallel and GPU-accelerated code implementation, analysis/post-processing options, and example use cases.

Keywords: Solar surface (1527); Solar photosphere (1518); Solar magnetic flux emergence (2000); Solar magnetic fields (1503); Solar differential rotation (1996); Solar meridional circulation (1874); Astronomy software (1855); Astronomy data analysis (1858); Computational methods (1965); Computational astronomy (293); GPU computing (1969); Open source software (1866); Publicly available software (1864)

1. INTRODUCTION

The magnetic field of the Sun plays a key role in solar and heliospheric physics, as it is a major driver of the structure and dynamics of the solar corona, and is the energy source for solar activity. The field is measured most easily in the photosphere, and global observations are provided in the form of full-disk magnetograms by instruments such as the Helioseismic and Magnetic Imager (HMI) onboard the Solar Dynamics Observatory (SDO) (Scherrer et al. 2012), the NSO Global Oscillation Network Group (GONG) (Harvey et al. 1996), and the Vector Spectromagnetograph

Corresponding author: Ronald M. Caplan
caplanr@predsci.com

(VSM) (Keller et al. 2003b), part of National Solar Observatory/Synoptic Optical Long-term Investigations of the Sun (NSO/SOLIS) (Keller et al. 2003a). Most regular observations of the surface field are along the Earth-Sun line (although the PHI imager on Solar Orbiter (Solanki et al. 2020) now provides intermittent measurements from other vantage points). This creates large data gaps in the full surface field. Observatories create so-called "synoptic" maps by combining portions of magnetograms over the course of a solar rotation to produce full-sun maps of the magnetic field, often in Carrington coordinates. These Carrington Rotation (CR) maps are really diachronic in nature (e.g., Linker et al. 2017), as they are built up over time and do not attempt to represent the Sun's magnetic field at any given instant. By their nature, synoptic CR maps necessarily contain older data, and do not reflect the surface flux evolution and emergence that occurred after data ingestion. From Earth's vantage point, one or both of the Sun's poles are either poorly observed or obscured throughout the year, and so filling the polar regions of the maps with values requires extrapolation or other methods.

Full-Sun magnetic maps are usually created from measurements of the line-of-sight (LOS) field, as this component is most reliably measured, especially in weaker field regions. They can be used to to predict solar irradiance and activity indices (Chapman & Boyden 1986; Henney et al. 2015; Warren et al. 2021), and they have a long history of use as boundary conditions in models of the solar corona and heliosphere, such as in potential field (e.g., Wiegelmann & Sakurai 2021) and magnetohydrodynamic (MHD) (e.g., Riley et al. 2011; Mackay & Yeates 2012; Gombosi et al. 2018; Feng 2019) models. The first potential field source-surface (PFSS) models to use photospheric measurements (e.g., Altschuler & Newkirk 1969) used the LOS field (B_{LOS}) directly as a boundary condition, and the first such global MHD model (Usmanov 1993) used the radial field (B_r) derived from a PFSS model using this specification. Wang & Sheeley Jr (1992) argued that the the LOS field is predominantly radial where it is measured in the photosphere, and specifying B_r by employing this assumption is the more appropriate boundary condition for coronal models. Mikić & Linker (1996) used this approach to directly specify B_r in a global MHD model. This approach has now become standard for most models (potential field or MHD), to the point that observatories typically provide CR maps of B_r derived from B_{LOS}.

While diachronic maps provide a useful average description of the Sun's field over the course of a solar rotation, for many applications, a representation of the global field at a particular time is desired. Instantaneous global observations of the field are not presently available, but the processes by which the magnetic flux on the Sun evolves (primarily differential rotation, meridional flow, supergranular diffusion, and random flux emergence) have been studied for many years. Following the Babcock-Leighton description of the solar dynamo (Babcock 1961; Leighton 1964) Surface Flux Transport (SFT) models have been developed to describe these processes (see the reviews by Sheeley (2005), Jiang et al. (2014) and Yeates et al. (2023) for a comprehensive history). These models treat the photospheric B_r as a passive scalar quantity that is evolved on the solar surface. SFT models first appeared in the 1980s (e.g., DeVore et al. 1984; Wang et al. 1989) to investigate the evolution of the surface field over the course of the solar cycle. Assimilative SFT models (e.g., Worden & Harvey 2000; Schrijver & DeRosa 2003) ingest data from magnetograms (typically B_r derived from B_{LOS}) on the observed portion of the Sun's surface, and evolve the field on the unobserved portions (including the poles). They produce a continuous approximation of the state of the photospheric magnetic field as a sequence of "synchronic" maps - maps that attempt to represent the state of the Sun's magnetic field at a given instant in time. While the emergence of new flux on the far-side of the Sun (such as active regions (ARs)) will be missed in this approach, the evolved existing field is expected to be much closer to the true state.

Assimilative SFTs have used a variety of assumptions and calculation methods, that can lead to quantitative differences between the maps. Three assimilative SFTs that have been used frequently to generate maps for coronal modeling and space weather applications are the Lockheed Martin Solar and Astrophysics Laboratory (LMSAL) Evolving Surface-Flux Assimilation Model (ESFAM) (Schrijver & DeRosa 2003), the Air Force Data Assimilative Photospheric flux Transport (ADAPT) model, (Arge et al. 2010, 2013; Hickmann et al. 2015), and the Advective Flux Transport (AFT) model Upton & Hathaway (2013). While ESFAM and ADAPT release pre-computed full-Sun maps for public use[1], none of these models are open source. As a result, users of maps must rely on the assumptions and parameter choices of the model developers, which may have been tailored for specific purposes. The ADAPT model has pioneered the use of multiple realizations to characterize possible variability, but Barnes et al. (2023) found that greater differences were present between maps generated by different SFTs than amongst the ADAPT ensembles. The

[1] ADAPT: https://nso.edu/data/nisp-data/adapt-maps, LMSAL-ESFAM: https://www.lmsal.com/forecast

consequences of different SFT map properties for coronal/heliospheric models have not been investigated extensively, although Knizhnik et al. (2024) demonstrated similar performance of solar wind models using ADAPT and AFT for a single Carrington rotation.

To foster the open use and development of SFT models, and allow community investigation of the effects of different model assumptions and parameters, we have created the Open-source Flux Transport (OFT) model. OFT is patterned after AFT in that it solves the transport equation on an Eulerian mesh, allows higher resolution than other models, and can utilize a super granular flow model (Hathaway et al. 2010, 2015) to better represent the quiet sun network and the decay of ARs. Like ADAPT, OFT allows for the generation of an ensemble of maps, and like ESFAM and ADAPT, can incorporate the addition of random flux to maintain the quiet sun background magnetic flux away from the window where assimilation occurs.

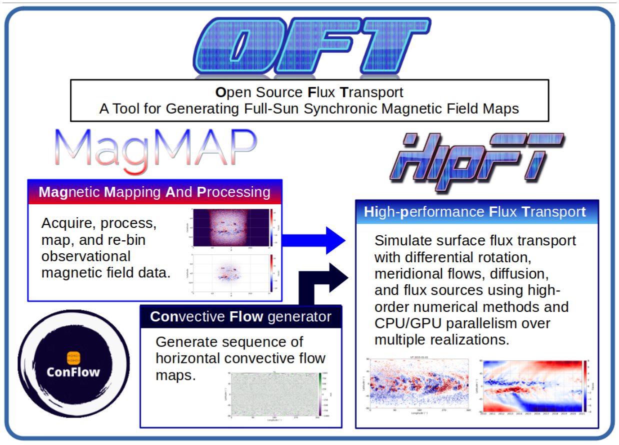

OFT
Open Source Flux Transport
A Tool for Generating Full-Sun Synchronic Magnetic Field Maps
MagMAP
Magnetic Mapping And Processing
Acquire, process, map, and re-bin observational magnetic field data.
HipFT
High-performance Flux Transport
Simulate surface flux transport with differential rotation, meridional flows, diffusion, and flux sources using high-order numerical methods and CPU/GPU parallelism over multiple realizations.
ConFlow
Convective Flow generator
Generate sequence of horizontal convective flow maps.

Figure 1. Overview of the components of the Open-source Flux Transport model.

OFT is publicly hosted on github[2], allowing for community use, contributions, and development. It blends several key state-of-the-art features from other SFT models into a modular, modern, and computationally efficient code base. OFT is broken up into three main components (shown in Fig. 1): Magnetic Mapping and Processing (MagMAP[3]), Convective Flow Generator (ConFlow[4]), and High-performance Flux Transport (HipFT[5])). Each component has its own independent public repository, which are all added to the OFT repository as git submodules[6]. MagMAP is used to obtain and accurately remap line-of-sight and vector magnetograph observations into heliographic coordinates, as well as down-sample the resulting map to a desired resolution in an integral flux-preserving manner. The resulting sequence of map files and meta data is then ready to be used by HipFT's data assimilation module. Details on the data acquisition and mapping techniques used in MagMAP will be described in a forthcoming OFT paper (Upton 2025). ConFlow is used to generate a sequence of randomly-seeded super granular tangential surface flows which convect according to specified differential rotation and meridional flow models. The resulting sequence of flows and metadata are then ready to be used in HipFT. Details on the supergranular flow model used in ConFlow will be described in a forthcoming OFT paper (Attie 2025). HipFT (the focus of this paper) is the computational core of OFT, charged with integrating the surface flux transport model, and includes advection, diffusion, source terms, and data assimilation. While loosely based on the AFT model, HipFT is a new code built from the ground up including multiple extensions and enhancements.

The paper is organized as follows: In Sec. 2 we describe the components of HipFT's flux transport model. The numerical methods used to integrate the model and their validations are described in Sec. 3, while the code implementation and performance are shown in Sec. 4. The use of the HipFT is described in Sec. 5 with selected example cases described in Sec. 6. We discuss the availability of the code in Sec. 7, and summarize in Sec. 8.

[2] https://github.com/predsci/oft
[3] https://github.com/predsci/magmap
[4] https://github.com/predsci/conflow
[5] https://github.com/predsci/hipft - The version of HipFT described and used in this paper is `v1.14.1`.
[6] https://git-scm.com/book/en/v2/Git-Tools-Submodules

2. SURFACE FLUX TRANSPORT MODEL

As mentioned in Sec. 1, HipFT's surface flux transport model is based on previous SFT models, especially the AFT model (Upton & Hathaway 2013). SFTs generally solve some form of the advective-diffusion equation for B_r with source terms (recently, an approach to SFT modeling using Physics-Informed Neural Networks has also been described (Athalathil et al. 2024)). HipFT solves the following form of this equation:

$$\frac{\partial B_r}{\partial t} = -\nabla_s \cdot (B_r \,\mathbf{v}) + \nabla_s \cdot (\nu \,\nabla_s \,B_r) + S + D, \tag{1}$$

where $B_r(\mathbf{x}, t)$ is the surface radial magnetic field, $\mathbf{v}(\mathbf{x}, t, B_r) = (v_\theta, v_\phi)$ is the non-homogeneous surface flow velocity vector, $\nu(\mathbf{x})$ is the diffusivity coefficient, $S(\mathbf{x}, t, B_r)$ is a source term, ∇_s and $\nabla_s\cdot$ are the two-dimensional spherical surface gradient and divergence operators respectively, $D(\mathbf{x}, t, B_r, B_{r;d}(t))$ is the application of data-assimilation where $B_{r;d}(t)$ is the data being assimilated, and $\mathbf{x} = (\theta, \phi)$, where θ and ϕ are the co-latitudinal and longitudinal directions respectively.

2.1. *Surface Flows*

The surface flows are implemented using the advection term

$$\nabla_s \cdot (B_r \,\mathbf{v}) = \frac{1}{R_\odot \,\sin\theta} \frac{\partial}{\partial\,\theta}(\sin\theta \,B_r \,v_\theta) + \frac{1}{R_\odot \,\sin\theta} \frac{\partial}{\partial\phi}(B_r \,v_\phi), \tag{2}$$

where $v_\theta(\mathbf{x}, \mathbf{B_r})$ and $v_\phi(\mathbf{x}, \mathbf{B_r})$ are the flow velocities in the co-latitude and longitudinal directions respectively, and we take the solar radius to be $R_\odot = 6.69 \times 10^8$ m/s. HipFT allows flexibility in specifying these flows, including allowing fully custom flows to be read from file(s). It also includes built-in common analytic descriptions of differential rotation and meridional flows. Here we describe these flow options, as well as a magnetic field flow attenuation.

2.1.1. *Analytic models for differential rotation and meridional flows*

A commonly used model for differential rotation (DR) in the Carrington frame can be expressed as

$$v_\phi(\theta) = \left[d_0 + d_2 \,\cos^2(\theta) + d_4 \,\cos^4(\theta)\right] \,\sin\,\theta, \tag{3}$$

where the parameters d_0, d_2, and d_4 are inputs in units of m/s and chosen based on analysis of solar observations. Several studies have produced various values for these parameters (Howard & Harvey 1970; Snodgrass 1983; Snodgrass & Ulrich 1990; Upton & Hathaway 2014). HipFT allows the user to specify these values and vary them over multiple realizations. The default values are set to $d_0 = 46$ m/s, $d_2 = -262$ m/s, and $d_4 = -379$ m/s, which are updated parameters used in the AFT model based on the data and procedures described in Hathaway et al. (2022). In Fig. 2, we show the profile of the default HipFT DR model.

For meridional flows (MF), several models have been used (Wang et al. 1989; Mackay & Van Ballegooijen 2006; Yang et al. 2024). For HipFT, we implement the form used in AFT (Hathaway et al. 2022), given by

$$v_\theta(\theta) = -\left[m_1 \,\cos\,\theta + m_3 \,\cos^3\theta + m_5 \,\cos^5\theta\right] \,\sin\,\theta, \tag{4}$$

where, as in the DR model, the parameters m_1, m_3, and m_5 are input in units of m/s and chosen based on analysis of solar observations. In HipFT, the default values are $m_1 = 22$ m/s, $m_3 = 11$ m/s, and $m_5 = -28$ m/s, which are derived in the same manner as those for DR described above. In Fig. 2, we show the profile of the default HipFT MF model.

2.1.2. *Custom flow profiles and supergranular flows*

HipFT allows the user to specify a sequence of velocity map files that specify flows over time to be added to any already selected analytical flow profile such as those in Sec. 2.1.1. A motivating use case for this feature is adding super granular convective flow models (Rincon & Rieutord 2018). In Sec. 6.1, we show an example of a production HipFT run utilizing such flows generated by ConFlow (Attie 2025).

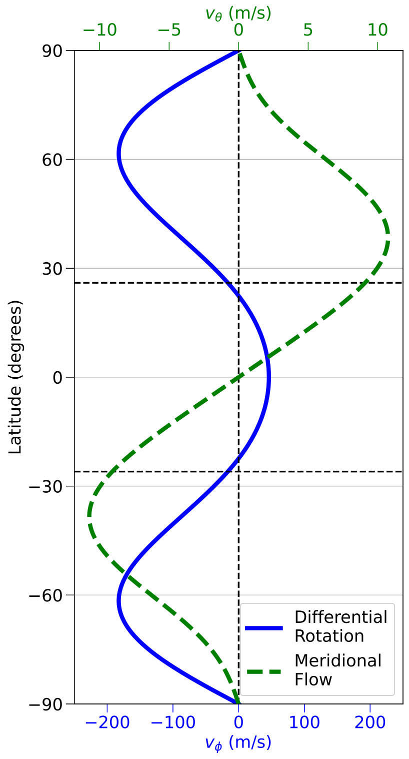


Figure 2. Analytic flow models for differential rotation (blue solid line) and meridional flows (green dashed line) used in HipFT. The profiles are shown using HipFT's default parameter values.

2.1.3. *Flow attenuation*

Using flows that are independent of the value of the surface magnetic field can miss important dynamics. For example, active regions (AR) are observed to have different flows profiles than DR and MF measured in the quiet Sun (Stenflo 1974, 1977). To account for this, we add the option to attenuate the flow based on the magnitude of the surface field. This takes the form

$$v_{\theta/\phi} \rightarrow v_{\theta/\phi}\left[1.0 - \tanh\left(\frac{|B_r|}{B_0}\right)\right], \tag{5}$$

where B_0 is an input saturation value (default $B_0 = 500$ Gauss).

2.2. *Diffusion*

The decay of ARs and flux cancellation caused by supergranular and granular flows require extremely high-resolution flow profiles to capture directly, and are subject to other processes not able to be directly modeled in SFT models. Instead, it is common to add a diffusion term to the SFT model to try to capture the mean effect. The diffusion term in HipFT appears as

$$\nabla_s \cdot (\nu\, \nabla_s\, B_r) = \frac{1}{R_\odot^2\, \sin\theta}\frac{\partial}{\partial\,\theta}\left(\nu(\theta,\phi,B_r)\,\sin\theta\,\frac{\partial\, B_r}{\partial\,\theta}\right) + \frac{1}{R_\odot^2\,\sin^2\theta}\frac{\partial}{\partial\phi}\left(\nu(\theta,\phi,B_r)\frac{\partial B_r}{\partial\phi}\right), \tag{6}$$

where $\nu(\theta,\phi,B_r)$ is the chosen diffusivity.

The value for ν required to obtain the desired amount of flux cancellation and field decay varies widely in the literature, ranging from $100\,\mathrm{km}^2/\mathrm{s}$ to over $600\,\mathrm{km}^2/\mathrm{s}$ (Jiang et al. 2014; Yeates et al. 2023). Selecting a scalar value for ν is difficult, as the required diffusivity to match observations can depend on spatial scale and the chosen flow velocities (Jiang et al. 2014). A further consideration is the (sometimes overlooked) contributions to diffusivity caused by the model's numerical method used to integrate advection. For example, when using finite-difference or finite-volume Upwinding (see Sec. 3.6.2), a significant amount of diffusivity is added to the solution. As a result, the model requires a smaller value of ν to achieve the desired amount of diffusion; making the required value resolution dependent. Another example is when using central differencing for advection, where the scheme is unconditionally unstable unless enough diffusion is added for stabilization.

Diffusion is also useful when running with super granular convective flows. Since small regions of convergent flows can appear, low-diffusive advection schemes such as the default scheme used in HipFT (see Sec. 3.6) can result in high-valued 'spike' pixels (as there are no radial inflow or outflow in the model). Although such convergent flows are often temporary (due to the rapid changes in supergranulur flows) large magnitude spikes can occasionally still occur. Introducing some amount of diffusion helps to reduce/eliminate them, with the optimal amount of diffusion to use being scheme and use-case specific.

The implementation of the diffusion equation of Eq. 6 allows HipFT to be re-purposed as an efficient flux-preserving magnetogram smoother (see Sec. 6) in which case the diffusivity can be set based on the grid cell size as:

$$\nu_g = \alpha_\nu \left[(\Delta\theta)^2 + (\Delta\phi \sin\theta)^2 \right], \tag{7}$$

where α_ν is a user specified factor and $\Delta\theta$ and $\Delta\phi$ are the local grid cell sizes.

HipFT also allows a user to easily write their own diffusivity as any function of space and B_r. Additionally, it includes the ability to read an external file for supplying a custom (optionally spatially-dependent) diffusivity.

2.3. *Data assimilation*

Data assimilation refers to ingesting available observational data into the model over time and is a key component for SFT models being used to generate synchronic full-Sun magnetic maps. Sophisticated methods of accounting for data uncertainty in the assimilation (such as Kalman filtering) have been utilized in SFT models (Hickmann et al. 2015; Dash et al. 2024). The data assimilation in the present version of HipFT uses a custom weighting between the observed and model data. It is designed to ingest the output data of the MagMAP software package (Upton 2025), where at each assimilation time t, the field is updated as

$$B_r(\mathbf{x},t) = F(\mathbf{x},t)\, B_{r;d}(t) + (1 - F(\mathbf{x},t))\, B_r(\mathbf{x},t), \tag{8}$$

where $F \in [0,1]$ is an assimilation weight function. While the weight layer in the MagMAP assimilation maps (see Upton (2025)) can be customized by the user based on more sophisticated uncertainty methods (such as those mentioned above), the default weighting is a simple function of center-to-limb angle and latitude, described as

$$F = \begin{cases} \mu^{\alpha_\mu}, & (\mu > \mu_{\text{lim}}) \wedge (|\theta_{\text{l}}| < \theta_{\text{l,lim}}) \\ 0, & \text{o.w.} \end{cases} \tag{9}$$

where $\mu = \cos\theta_d \in [0,1]$, θ_d is the center to limb angle of the observed solar disk data, θ_l is the map latitude, and α_μ, μ_{lim}, and $\theta_{\text{l,lim}}$ are chosen parameters.

The data assimilation functionality is implemented by reading in 3D MagMAP files. In each file, the first layer contains the new data, the second layer is taken to be the default full weight function to use directly (F), and the third layer contains the μ values extracted from the data, which can be used with the user-specified assimilation function of Eq. 9. Weights can also be fully customized by modifying the weight layer of the data files from MagMAP, or by implementing a new weight subroutine into HipFT. Details on the weight models and their effects are described in a forthcoming OFT paper (Upton 2025). MagMAP is designed to ingest data from magnetograms, but in principle, other proxy sources of magnetic field, such as EUV imaging (Hess Webber et al. 2020) and/or helioseismology (Liewer et al. 2014), could be used to insert ARs observed on the far side of the Sun (Arge et al. 2013; Chen et al. 2022; Yang et al. 2024).

HipFT can optionally enforce flux balance during the assimilation step by rewriting Eq. 8 as

$$B_r^{\text{new}} = B_r^{\text{curr}} + \Delta B_r, \tag{10}$$

where $\Delta B_r = F(\mathbf{x})\,[B_{r;d} - B_r^{\text{curr}}]$. Before the data is assimilated, ΔB_r is multiplicativly flux balanced as

$$\begin{aligned} \Delta B_r &= \Delta B_r / \sqrt{|\Phi_+/\Phi_-|}, \qquad \Delta B_r > 0, \\ \Delta B_r &= \Delta B_r * \sqrt{|\Phi_+/\Phi_-|}, \qquad \Delta B_r \leq 0, \end{aligned} \tag{11}$$

where Φ_+ and Φ_- are total fluxes of the initial difference, computed as surface integrals of ΔB_r for positive and negative values separately. By using this method, neutral lines are preserved, and the balance of the flux is distributed proportionally to the field.

2.4. *Flux emergence*

The quiet sun network is made up of an ocean of small-scale magnetic flux that is observed to be completely replaced by emergence and cancellation over a couple of days (Schrijver et al. 1997). To account for this source of flux, earlier SFTs (e.g. ADAPT and ESFAM) have incorporated random flux emergence into the maps. HipFT uses the source term, S, in Eq. 1 to facilitate flux emergence. A static source term can be read from a file and then applied each time step. Additionally, two models of adding small-scale (grid-level) random flux emergence (RFE) are implemented. For each model, the user can specify the total unsigned flux per hour, Φ/hr, of the source terms, as well as the desired time between source terms (representing the lifetime of the random flux elements).

Each model uses values sampled from a normal distribution with mean 0 and standard deviation σ

$$X \sim \mathcal{N}(0, \sigma^2) = \frac{1}{\sigma\sqrt{2\,\pi}} \exp\left[-\frac{x^2}{2\,\sigma^2}\right].$$

For the first model, we set the average source term to represent a uniform unsigned flux per hour per cell. We set the flux per hour of each cell such that the mean unsigned flux per hour ($\langle|\varphi|\rangle$) of each cell is equal to the chosen total unsigned flux per hour divided evenly across the total number of grid cells N_{cells}:

$$\langle|\varphi|\rangle = \frac{\Phi/\text{hr}}{N_{\text{cells}}}.$$

Since for a set of random variables $X \sim \mathcal{N}(0, \sigma^2)$,

$$\langle|X|\rangle = \frac{\int_{-\infty}^{\infty} |x|\, \mathcal{N}(0,\sigma^2)\, dx}{\int_{-\infty}^{\infty} \mathcal{N}(0,\sigma^2)\, dx} = \sqrt{\frac{2}{\pi}}\,\sigma, \tag{12}$$

we find

$$\sigma = \frac{\Phi/\text{hr}}{N_{\text{cells}}}\sqrt{\frac{\pi}{2}}.$$

Since if $X \sim \mathcal{N}(0, \sigma^2)$ and $Y \sim \mathcal{N}(0, 1)$, then $X = \sigma\, Y$, we can use a standard normal distribution to generate the flux per hour of each cell i as

$$\varphi_i = \frac{\Phi/\text{hr}}{N_{\text{cells}}}\sqrt{\frac{\pi}{2}}\, Y_i.$$

Dividing by the local cell area ($\Delta A_{j,k} = R_\odot^2\, \sin\theta_j\, \Delta\theta_j\, \Delta\phi_k$) yields the source B_r/hr value for each cell:

$$S_{j,k} = \frac{\Phi/\text{hr}}{\Delta A_{j,k}\, N_{\text{cells}}}\sqrt{\frac{\pi}{2}}\, Y_i. \tag{13}$$

This method creates larger source values in regions with finer grid resolution. The motivation to mimic the behavior of observations where a finer image resolution will often reveal larger values of unsigned flux that would otherwise be averaged out over larger resolution elements. In practice for uniform grids in θ and ϕ, this will lead to larger absolute values of B_r in grid-cells near the poles due to the $\sin\theta$ weighting in the cell area. However, the contribution from each individual grid cell can be rapidly spread throughout the grid during the calculation via a combination of diffusion and flows, especially at the poles due to the $1/\sin\theta$ geometric factor in the longitudinal direction.

For the second model, we set the average source term to represent a uniform unsigned flux per hour per unit area:

$$\langle |S| \rangle = \frac{\Phi/\mathrm{hr}}{4\,\pi\,R_\odot^2}.$$

Using Eq. 12, we now get

$$\sigma = \frac{\Phi/\mathrm{hr}}{4\,\pi\,R_\odot^2}\sqrt{\frac{\pi}{2}},$$

and therefore the source B_r/hr value for each cell is computed as

$$S_i = \frac{\Phi/\mathrm{hr}}{4\,\pi\,R_\odot^2}\sqrt{\frac{\pi}{2}}\,Y_i. \tag{14}$$

This method results in a uniform distribution of B_r values over the grid cells regardless of cell area or grid spacing. This may be desirable when the surface flux distribution is thought to be fully resolved and the amount of random flux emergence per unit area should be constant. In practice, for a run using a uniform grid in θ and ϕ, the combination of diffusion and flows can lead to weaker, more diffuse contributions at the poles relative to the equatorial latitudes.

Due to their practical differences, the choice between the two above methods may depend on the particular application. For each method, two source maps of random flux ($S_{\mathrm{RFE};1}$ and $S_{\mathrm{RFE};2}$) are generated and flux balanced in the same manner as Eq. 11. At every step, a linear interpolation between the two current source maps is calculated for the current time t and the resulting source term is integrated in Eq. 1:

$$S_{\mathrm{RFE}}(t) = (1-\alpha(t))\,S_{\mathrm{RFE};1} + \alpha(t)\,S_{\mathrm{RFE};2}, \tag{15}$$

where $\alpha(t) = (t - t_{\mathrm{RFE},1})/\tau_{\mathrm{RFE}}$. When the time of the second source map is reached, a new source map is generated and the process repeats. This ensures a smooth evolution of the random flux elements over time.

To validate the implementation of RFE in HipFT, in Fig. 3 we show the result of running with only the RFE source term activated (i.e. no flows or diffusion) starting with a zero map for each method shown above with both an infinite and finite (0.3 hour) RFE lifetime. We see that the specified total unsigned flux per hour (here, 100×10^{21} Mx/hr) is realized in the case of infinite lifetime, while with a finite lifetime, it grows much more slowly due to flux cancellation between the pairs of RFE samples. In both cases, the flux remains balanced. The latitudinal dependence of the field when using method 1 can also be seen. See Sec. 6.1 and Upton (2025) for more details about the effects of the RFE model choices on production runs.

2.5. *Multiple Realizations*

When using phenomenological models like SFT that have a multitude of uncertainties including data quality, data-derived model parameters, model choices, multiple resolutions, etc., it is very important to try to identify and quantify the uncertainties (QU). A key tool in QU is generating multiple realizations of the model solutions by varying the model parameters, leading to an ensemble of solutions. Ideally, these ensembles should properly sample the span of the farthest reasonable boundaries in parameter space of the current model while keeping the number of realizations as low as possible.

HipFT has been designed to compute many realizations of maps within a single run efficiently. This makes it straight forward to set a range of model parameters to generate many realizations. The python post processing scripts included with HipFT are also designed to process the multiple realizations together. Currently, HipFT allows the user to specify an array of parameters for diffusion coefficient values, flow attenuation values, the model coefficients for DR and MF flow profiles, data assimilation latitude limits, μ limits, and weight exponents. The mechanism to provide model parameters across realizations is modular, making it straight forward for users to add new cross-realization parameters. In Sec. 4 we show how the realizations are distributed across the computational units of a HipFT run, while in Sec. 6 we show an example run with multiple realizations.

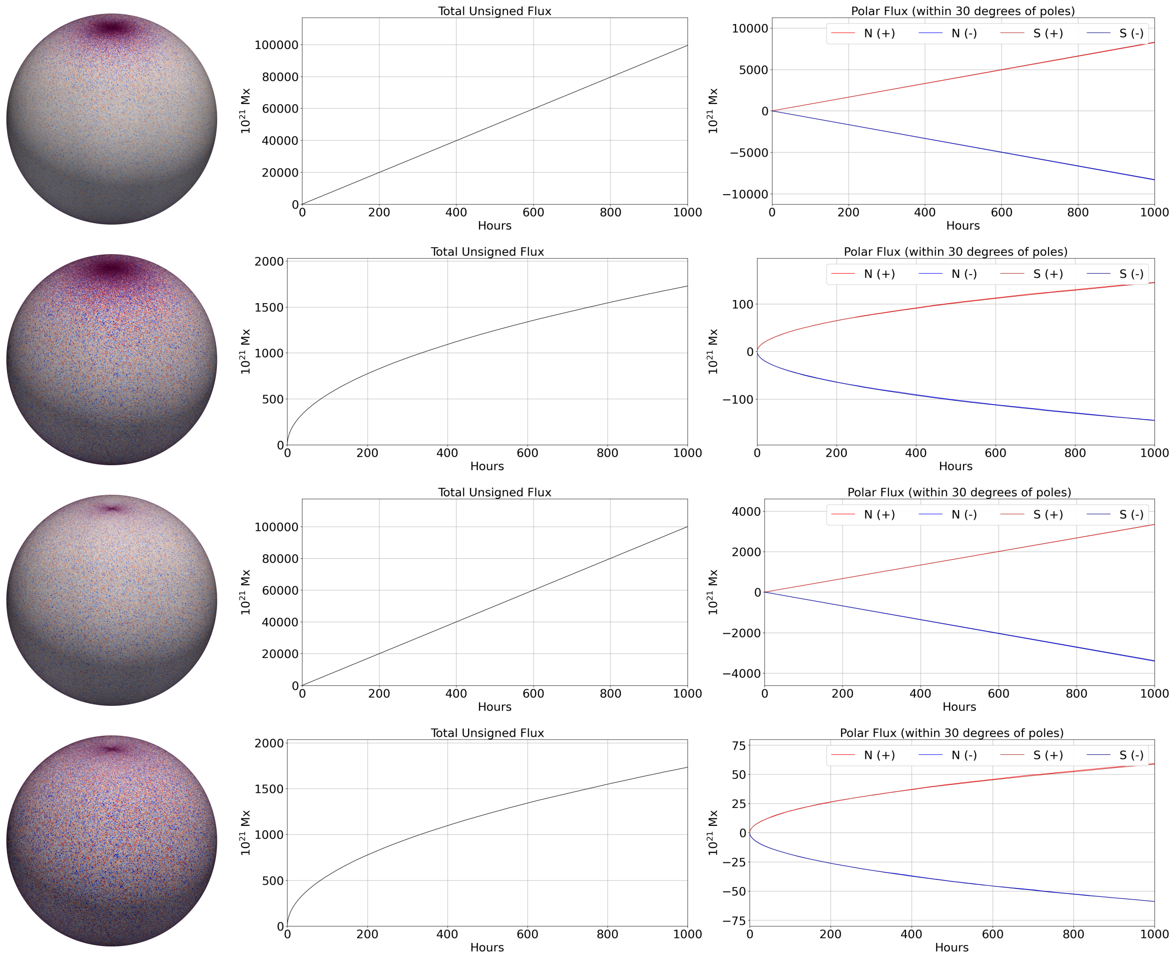


Figure 3. Random flux emergence source term runs for method 1 (M1) and method 2 (M2) with an end time of 1000 hours. A total unsigned flux rate of 100×10^{21} Mx/hr is set, and no other parts of the SFT model are active (no flows and no diffusion). The results for M1 with lifetimes of infinity and 0.3 hours, and M2 with lifetimes of infinity and 0.3 hours are shown top to bottom. For each run, the final map is shown (at a b_0 angle of 30 degrees) along with the total unsigned flux, and the polar flux within 30 degrees (left to right). The final map for runs with infinite lifetimes are shown on a color scale from -1000 to 1000 Gauss, while those with 0.3 hour lifetimes are shown from -100 to 100 Gauss. The latitudinal dependence of the resulting field for method 1 is apparent.

2.6. *Analysis*

There are a variety of derived quantities that are useful when analyzing full-Sun magnetic maps. In HipFT we compute several of these quantities (described below) and output them at a chosen cadence in a text file per realization we refer to as a 'run history'. Python scripts are also provided to compute these quantities on maps that have already been written to disk, which can be used to generate history files on observational data products or other SFT models after converting their maps to the HipFT file and grid format.

The derived quantities written to the history files are the total positive and negative flux given by

$$\int_\Omega B_r \, d\Omega, \qquad (B_r > 0) \qquad \text{and} \qquad \int_\Omega B_r \, d\Omega, \qquad (B_r < 0),$$

respectively, where $\Omega = \sin\theta \, d\theta \, d\phi$, the total fluxes limited to the polar regions as defined by a user selected latitude cutoff, the area of the selected polar regions, the minimum, maximum, and minimum magnitude of B_r over the whole map, and the strengths of the equatorial and axial dipole moments (Wang & Sheeley 1991). The strength of equatorial dipole moment is calculated as

$$H(t) = \sqrt{h_1^2(t) + h_2^2(t)}, \qquad (16)$$

where

$$h_1(t) = \frac{3}{4\pi} \int_\Omega B_r \, \sin\theta \, \cos\phi \, d\Omega, \tag{16a}$$

$$h_2(t) = \frac{3}{4\pi} \int_\Omega B_r \, \sin\theta \, \sin\phi \, d\Omega, \tag{16b}$$

while the strength of the axial dipole moment is calculated as

$$D(t) = \frac{3}{4\pi} \int_\Omega B_r \, \cos\theta \, d\Omega. \tag{17}$$

When running validation tests, we compute the difference between the HipFT solution and the true analytic solution using an absolute value version of the Hanna and Heinold (HH) metric (Hanna & Heinold 1986)

$$\mathrm{HH}_{||} = \sqrt{\frac{\sum_{i=1}^{N} |X_i - Y_i|^2}{\sum_{i=1}^{N} |X_i||Y_i|}}, \tag{18}$$

where N is the total number of grid points, X is the computed final map, and Y is the analytic solution map. The $\mathrm{HH}_{||}$ metric has been shown to be better for testing numerical solutions than the more commonly used normalized root mean-square error (Mentaschi et al. 2013).

HipFT contains several post-processing scripts that utilize these analysis outputs, including using them to compute additional derived quantities as described in Sec. 5.3.

3. NUMERICAL METHODS

HipFT uses a variety of numerical methods for each part of the integration of Eq. 1. In this section, we describe the implemented schemes, starting with a description of the initial conditions for the tests that will be used to validate the schemes. We culminate with the validations shown in Sec. 3.8.

3.1. *Test cases*

We use two test initial conditions, each of which have analytic time-dependent solutions. We can also set a constant angular velocity in ϕ corresponding to a full rotation, allowing us to test the advection schemes with any initial condition, including full-Sun magnetic maps. To normalize the tests, each is run to an end time of 672 hours (approximately one Carrington rotation) and, for analytic solutions, use a function amplitude of one. The ϕ velocity for rigid rotation tests is set to $v_\phi = 1.80766\,\mathrm{km/s}\,\sin\theta$.

The first test case is known as the "soccer ball" function[7]. It is an analytic time-dependent solution for the diffusion operator on the surface of a sphere, which can be combined with a full rigid rotation ϕ velocity to test advection-diffusion schemes. It is described as

$$u(\theta,\phi,t) = e^{-42\,\nu\,t}\left(Y_6^0(\theta,\phi) + \sqrt{\frac{14}{11}}\,Y_6^5(\theta,\phi)\right), \tag{19}$$

where ν is the chosen diffusion coefficient (in our tests we use $\nu = 500\,\mathrm{km}^2/\mathrm{s}$), and $Y_l^m(\theta,\phi)$ are the tesseral spherical harmonics. To compute the spherical harmonics, we have implemented the First Modified Forward Column Recursion method of Holmes & Featherstone (2002) based on the MATLAB code `spherefun.sphharm` from the ChebFun package (Driscoll et al. 2014).

The second test solution is a pair of modified Gaussians of opposite polarity described as

$$\begin{aligned} B_r(\theta,\phi,t) = & -\frac{1}{\sin\theta}\exp\left[-\frac{(\theta-\pi/4-v_\theta\,t)^2}{\sigma} - \frac{(\phi-\pi/2-v_\phi\,t)^2}{\sigma}\right] \\ & +\frac{1}{\sin\theta}\exp\left[-\frac{(\theta-\pi/4-v_\theta\,t)^2}{\sigma} - \frac{(\phi-3\pi/2-v_\phi\,t)^2}{\sigma}\right]. \end{aligned} \tag{20}$$

[7] https://www.chebfun.org/examples/sphere/SphereHeatConduction.html

The $1/\sin\theta$ term in front ensures an exact solution in θ over time (see Appendix C for details). Although, in general, the term creates a divergence at the pole, our test has two equal and opposite polarities approaching the pole, which should cause the flux to cancel, mitigating the issue. We use $\sigma = 0.03$ and set v_ϕ such that the Gaussian profiles travel a full rotation. For v_θ, we set its value such that the Gaussians travel an angular distance of $\Delta\theta = \pi/2$, making an end solution that should be the same as the initial solution flipped in latitude. By setting (or not) the velocities, we can test advection in the ϕ and θ directions independently or together.

The initial and analytic solution maps for the two test cases are shown in Fig. 4. For all test cases, we use the

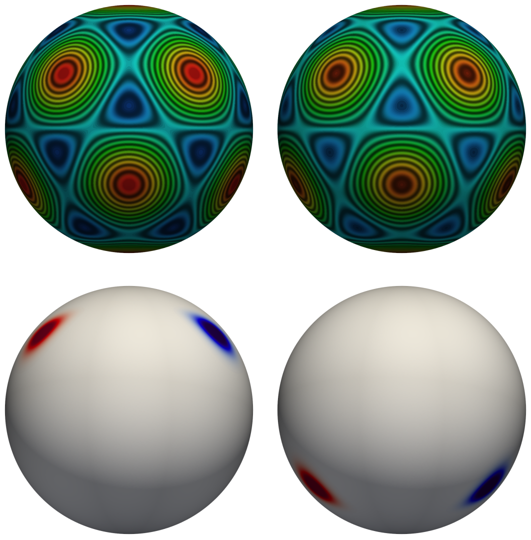


Figure 4. Test cases used for HipFT validations. The top row is the soccer ball test of Eq. 19, while the bottom row is the Gaussians test of Eq. 20. For each row, the initial condition is shown on the left, and the solution after 672 hours is shown on the right (for the Gaussians test, the initial condition is shown at a b_0 angle of $30°$, while the end solution is shown at a b_0 of $-30°$). The solutions are shown for the default resolution of 512×1024.

$\mathrm{HH}_{||}$ metric of Eq. 18 to compare the final computed output map with the analytic solution.

3.2. *Grid*

HipFT uses a logically-rectangular non-uniform spherical surface grid shown in Fig. 5. The main grid (where B_r resides) is defined over $\phi \in [0, 2\pi]$ and $\theta \in [0, \pi]$, which includes a one-point overlap in the ϕ periodic boundary (i.e. both the $\phi = 0$ and $\phi = 2\pi$ values are in the files). For internal calculations, an additional ϕ point is added, yielding a two-point overlap in the ϕ periodic boundary (shown in the lower left panel of Fig. 5 as the light blue square). The flow velocities and the advection fluxes (see Sec. 3.6.1) are placed on 'half' grids that are staggered to the B_r field in their corresponding direction (as shown in the right panel of Fig. 5 as the blue and red shaded squares). The diffusion viscosity (see Sec. 3.7) is defined on the 'half-half' grid (green shaded squares in the right panel of Fig. 5).

During the computation, when needed, the ϕ direction is 'seamed' by copying the overlapping point values as shown by the arrows in the left panel of Fig. 5.

For production runs of OFT, we currently use a default resolution of $\phi \times \theta = 1024 \times 512$. The resolution in HipFT is set based on the initial input map, and any data assimilation and/or flow maps must match that resolution (the OFT repository contains tools needed to re-bin and process maps to any desired resolution and can be used prior to running HipFT). Alternatively, the resolution can be set using input parameters for the case of a zero-value starting map or for validation test cases.

3.3. *Units*

HipFT internally uses spatial units of $R_\odot = 6.96 \times 10^{10}$ cm/s and time units of hours. It includes a module with preset constants to easily convert inputs into the internal code units. For instance, to convert the flow velocities from

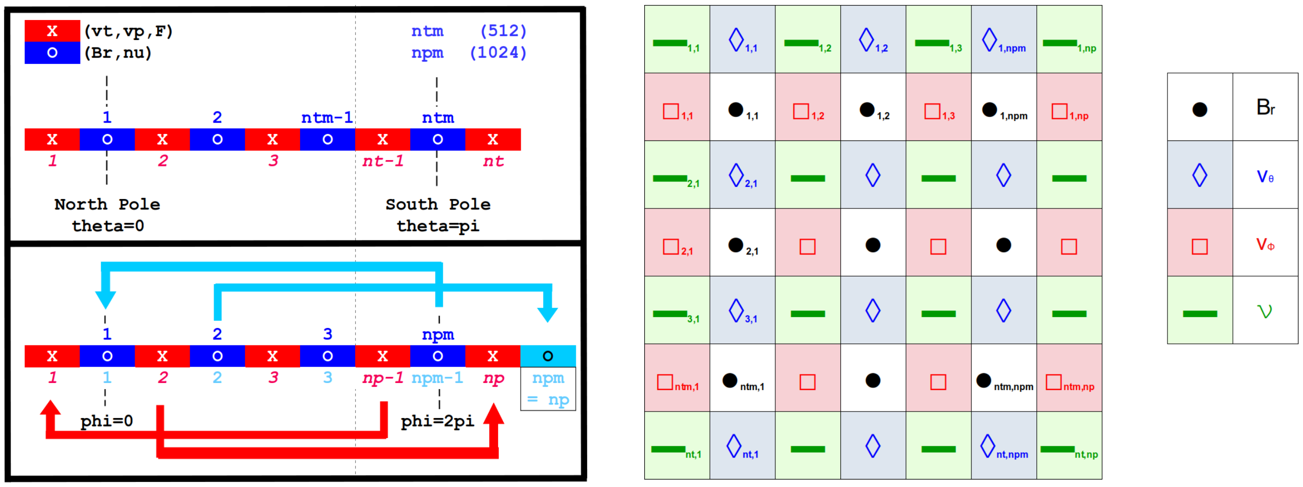


Figure 5. Computational grid layout of HipFT. In the left image, the light blue ϕ grid limit is used within the code, while the dark blue limits are used for the input and output maps. The two-point overlap and seam directions are indicated by the light blue arrows. The staggering of the field, velocity components, and diffusivity are shown in the right image.

m/s to $R_\odot$/hr we have m_s_to_rs_hr $= 5.172413793103448 \times 10^{-6}$, and for converting the diffusivity ν from km^2/s to $R_\odot^2$/hr we have km2_s_to_rs2_hr $= 7.43162901307967 \times 10^{-9}$.

3.4. *Time steps*

The time steps in HipFT are determined by a combination of stability limits, input/output times, and user choices. At each step, the time step is initially set to the remaining time for the simulation. It is then reduced through a series of checks which include the advection flow CFL stability condition (see Sec. 3.6.2), the time to the next requested map output, the time to the next data assimilation, the time to the next input flow, and the time to the next random flux generation (depending on which features in the code are being used). This method ensures that features like data assimilation and map output are performed exactly at the times proscribed, while also maintaining stable integration of the model. Additionally, various user options (such as a minimum and maximum time step) are available for additional control.

3.5. *Operator splitting*

At each time step, the various components of Eq. 1 are integrated separately (i.e. operator split) in the following sequence:

$$\begin{aligned} B_r^* &= A(B_r^n, \Delta t), \\ B_r^{**} &= D(B_r^*, \Delta t), \\ B_r^{n+1} &= S(B_r^{**}), \end{aligned} \tag{21}$$

where A is the advection term, D is the diffusion term, S is the source term, n is the current step number, and Δt is the current step's time step (the data assimilation occurs at the end of the advance). This sequence of operator splitting can introduce a $O(\Delta t)$ error (Simpson & Landman 2008). Another option is to use Strang splitting (MacNamara & Strang 2016) defined as

$$\begin{aligned} B_r^* &= S(B_r^n, \Delta t/2) \\ B_r^{**} &= A(B_r^*, \Delta t/2), \\ B_r^{***} &= D(B_r^{**}, \Delta t), \\ B_r^{****} &= A(B_r^{***}, \Delta t/2), \\ B_r^{n+1} &= S(B_r^{****}, \Delta t/2), \end{aligned} \tag{22}$$

which has a lower $O(\Delta t^2)$ splitting error. Using Strang splitting can be computationally more expensive due to the additional advection and source advances in each step. However, this is only when using a constant time step (not set by the flow CFL). When using the maximum allowed flow CFL time step, the CFL stable time step is now twice as large (due to the $\Delta t/2$ step sizes), cutting the total number of steps in half, negating the cost of the extra advection and source advances. Moreover, if diffusion is active, there are now half as many diffusion advances (each with a time step twice as large). Since the computation time of the super time stepping PTL scheme does not grow linearly with the time step (see Sec. 3.7.2), this can make the overall run faster, especially since diffusion can be a large portion of the total run time. Therefore, the time cost (or savings) of using Strang splitting is problem specific. In terms of accuracy, we have found that often the spatial errors dominate the total error, so Strang splitting does not significantly improve the accuracy overall. Therefore, the choice to use Strang splitting in HipFT is typically based on computational cost. As an example, in Table 1, we show the results of running the ϕ-rotation soccer ball advection-diffusion test case of Eq. 19 with and without Strang splitting, using both a small fixed time step and a time step set by the stable flow CFL. We see that with a fixed time step, the run is slower using Strang splitting, but when using the maximum allowed CFL

Fixed Time Step

	Δt	Wall Clock	Error
No Strang	0.50 hr	23.8 sec	4.4×10^{-5}
Strang	0.50 hr	26.7 sec	4.0×10^{-5}

Maximum Stable Flow CFL Time Step

	Δt	Wall Clock	Error
No Strang	0.62 hr	21.1 sec	4.8×10^{-5}
Strang	1.25 hr	15.6 sec	4.8×10^{-5}

Table 1. Effects of Strang splitting in HipFT for the ϕ-rotation diffusion-advection test case of Eq. 19. The run was performed with and without Strang splitting using a fixed time step, and using the maximum allowed stable flow CFL time step. The wall clock time (run on an NVIDIA RTX 3090Ti GPU) and the final $\mathrm{HH}_{||}$ solution error is shown. Due to the reduction in the number of diffusion advances, using Strang splitting is faster when using the maximum stable time step, but slower when using a fixed time step.

time step, the run is faster. In all cases, the total errors are comparable. As the advantage of using Strang splitting is problem specific, it is disabled by default in HipFT.

3.6. *Advection*

3.6.1. *Advection spatial schemes*

To descretize the advection term in Eq. 1, we use a finite-difference of the form

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

where F_θ and F_ϕ are computed either with an Upwinding/Central or WENO3-CS(h) scheme as described below.

Upwinding (UW) and Central Difference (CD)

Here, we set

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

where

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

and $\alpha_{\mathrm{uw}} \in [0.5, 1]$. Setting $\alpha_{\mathrm{uw}} = 0.5$ makes the scheme a central difference scheme which is second-order accurate, but when combined with explicit time stepping schemes, is unconditionally unstable (Strikwerda 2004). This can be countered by adding enough diffusion to the model to stabilize the method (as is done in the AFT model). The default value in HipFT when using this method is $\alpha_{\text{uw}} = 1$ (upwind) which results in a first-order accurate discretization ($O(\Delta\theta)$,$O(\Delta\phi)$) and adds a significant amount of diffusivity to the solution.

Weighted Essentially Non-Oscillatory, third-order scheme with grid-step epsilon (WENO3-CS(h))

In order to have a more accurate advection integration and avoid the implicit diffusion added by the upwind scheme, we have implemented a Weighted Essentially Non-Oscillatory (WENO) scheme (Liu et al. 1994). We chose to implement a WENO3 scheme as a balance of solution quality and complexity. Special consideration is needed due to the use of non-homogeneous velocities and a non-uniform grid (Smit et al. 2005; Shadab et al. 2019). For the flux, we use a localized Lax-Friedrichs (LLF) spitting (Shu 2009; Li 2006). The use of spherical coordinates poses a challenge for WENO implementations, specifically in the theta direction. Here, we treat the theta direction in the same manner as a Cartesian dimension which is a commonly used option even though it may potentially reduce the accuracy (Li 2006; Mignone 2014). While WENO3 schemes are formally 3rd-order accurate in smooth solution regions, they can be less accurate in the presence of critical points and discontinuities without adding additional grid cells to the scheme (Baeza et al. 2020). However, Cravero & Semplice (2015) showed that by using the cell spacing for the required 'small' constant ϵ_w (used to avoid a division by zero) in the formulation, the WENO3 scheme can retain its 3rd-order accuracy even close to local extrema on nonuniform grids. We denote this scheme as WENO3-CS(h) and is the one implemented in HipFT. While using other values of ϵ_w can make the standard WENO3 scheme more accurate, knowing in advance what value is best is difficult, making the WENO3-CS(h) scheme easier to use and more robust. A full description of the scheme is given in Appendix B, where in Fig. 15, the convergence for the Upwind, WENO3 with a constant ϵ_w, and WENO3-CS(h) schemes are compared.

In Fig. 6 we show the qualitative difference between the upwind and WENO3-CS(h) schemes in a realistic case of applying meridional and differential rotation flows to a magnetic map. We see that the upwind scheme is much more

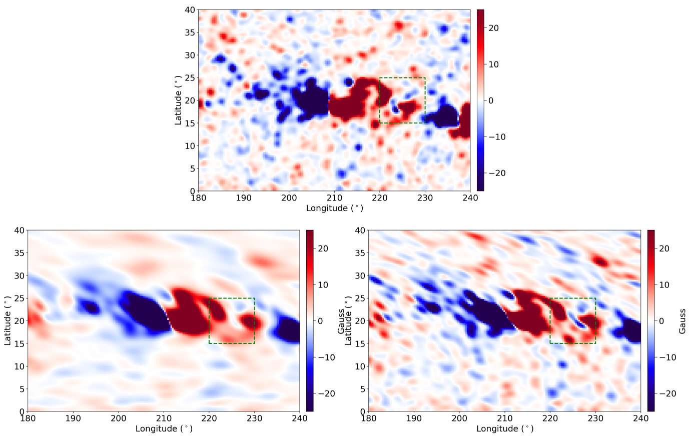


Figure 6. Qualitative comparison of the Upwind and WENO3-CS(h) scheme. An initial magnetic map (top) was integrated in HipFT for 672 hours with default analytic differential and meridional flows (no diffusion) with the upwind scheme (bottom left) and the WENO3-CS(h) (bottom right). A zoomed-in portion of the maps is shown, highlighting how the upwind scheme is much more diffusive than the WENO3-CS(h) scheme. This can cause qualitative structural changes, such as the elimination of the small parasitic polarity highlighted in the green square.

diffusive than the WENO3-CS(h) scheme, leading to the washing out of detailed structures, while the WENO3-CS(h) retains them.

Boundary conditions

For transport on a spherical surface, there are no physical boundaries, but mathematically, boundaries appear because of the periodicity in ϕ and the geometric singularity at the poles in θ. In the ϕ direction, we invoke a periodic boundary condition over the 2-cell overlap described in Sec. 3.2. Since the grid is not split across the θ or ϕ domain over MPI ranks (see Sec. 4), the periodic cells are readily accessible without needing inter-process communication.

In θ, the poles are handled specially. Since the WENO3-CS(h) scheme has a 5-point stencil, we use the upwind scheme at the cells adjacent to the poles. Even though this can reduce the accuracy of the scheme in the θ direction, is it a simple solution that typically does not degrade the accuracy of solutions with little structure at the pole. For the poles themselves, we utilize a finite-volume approach described in Appendix A.

3.6.2. *Advection temporal schemes*

Forward-Euler (FE)
The forward Euler scheme is first-order accurate in time ($O(\Delta t)$) and described as

$$B_r^{n+1} = B_r^n - \Delta t\, F_A(B_r^n) \tag{23}$$

where

$$F_A(B_r^n) = \nabla_s \cdot (\mathbf{v}_s^n\, B_r^n).$$

To ensure stability, the time step must be bound by the CFL condition, which in this case is approximated by

$$\Delta t < \frac{1}{2}\left[\frac{|v_\theta|}{\Delta\theta} + \frac{|v_\phi|}{\sin\theta\,\Delta\phi}\right]^{-1}, \tag{24}$$

where the velocities are taken from the maximum of the flanking staggered values in each direction. We also apply a 'safety factor' of ~ 0.9 to the limit.

Strong Stability Preserving Runge-Kutta (SSPRK)
When using the WENO3-CS(h) scheme from Sec. 3.6.1, the use of a higher-order time-stepping scheme is required for stability, and the use of a total value diminishing (TVD) scheme is typically used for its additional desirable properties (Lunet et al. 2017). HipFT provides two TVD scheme options: the third-order, three stage, strong stability preserving Runge-Kutta (SSPRK(3,3)) scheme and the third-order, four stage SSPRK(4,3) scheme (Spiteri & Ruuth 2002; Gottlieb et al. 2011). The SSPRK(3,3) (also know as RK3-TVD) and SSPRK(4,3) are described as

$$\begin{array}{c}
\hline
\text{SSPRK(3,3)} \\
\hline
f^{(1)} = f^n - \Delta t\, F_A(f^n) \\
f^{(2)} = \frac{3}{4} f^n + \frac{1}{4} f^{(1)} - \frac{1}{4}\Delta t\, F_A(f^{(1)}) \\
f^{n+1} = \frac{1}{3} f^n + \frac{2}{3} f^{(2)} - \frac{2}{3}\Delta t\, F_A(f^{(2)}) \\
\hline
\text{SSPRK(4,3)} \\
\hline
f^{(1)} = f^n - \frac{1}{2}\Delta t\, F_A(f^n) \\
f^{(2)} = f^{(1)} - \frac{1}{2}\Delta t\, F_A(f^{(1)}) \\
f^{(3)} = \frac{2}{3} f^n + \frac{1}{3} f^{(2)} - \frac{1}{6}\Delta t\, F_A(f^{(2)}) \\
f^{n+1} = f^{(3)} - \frac{1}{2}\Delta t\, F_A(f^{(3)})
\end{array} \tag{25}$$

The SSPRK(4,3) adds an additional evaluation stage, while remaining third-order accurate. Its advantage lies in that, while the stable time step limit for the SSPRK(3,3) is the same as Eq. 24 used for the upwind scheme, the limit for the SSPRK(4,3) scheme is twice as large (Gottlieb et al. 2011), i.e.

$$\Delta t < \left[\frac{|v_\theta|}{\Delta\theta} + \frac{|v_\phi|}{\sin\theta\,\Delta\phi}\right]^{-1}. \tag{26}$$

Therefore, adding $\sim 30\%$ more computation is off-set by needing 50% less time steps (if the overall calculation time step is being set by the flows). This advantage is increased when combined with diffusion, as the diffusion term is expensive to calculate per time step. For example, running the soccer ball diffusion test case of Eq. 19 at 512×1024 resolution with rigid rotation in ϕ, using the maximum allowed time step with SSPRK(3,3) takes 38.5 seconds on an RTX 3090Ti GPU, while for SSPRK(4,3), it takes 26.7 seconds (30% faster). The $\mathrm{HH}_{||}$ (see Eq. 18) between the analytic solution and the simulation for the two runs is nearly the same (4.3×10^{-5} for SSPRK(3,3) and 4.8×10^{-5} for SSPRK(4,3)) showing that the speedup of the SSPRK(4,3) scheme does not significantly effect the solution accuracy. Therefore, HipFT uses the SSPRK(4,3) scheme by default.

3.7. *Diffusion*

3.7.1. *Diffusion spatial schemes*

To descretize the diffusion term in Eq. 1, we use the second-order central finite difference

$$
\begin{aligned}
&\nabla_s \cdot (\nu(\theta,\phi)\, \nabla_s B_r) \approx \\
&\frac{1}{\sin\theta_j\, \Delta\theta_j}\left[\frac{\nu_{j+\frac{1}{2},k+\frac{1}{2}} + \nu_{j+\frac{1}{2},k-\frac{1}{2}}}{2}\, \sin\theta_{j+\frac{1}{2}}\, \frac{B_{r:j+1,k} - B_{r:j,k}}{\Delta\theta_{j+\frac{1}{2}}}\right. \\
&\qquad\qquad \left. - \frac{\nu_{j-\frac{1}{2},k+\frac{1}{2}} + \nu_{j-\frac{1}{2},k-\frac{1}{2}}}{2}\, \sin\theta_{j-\frac{1}{2}}\, \frac{B_{r:j,k} - B_{r:j-1,k}}{\Delta\theta_{j-\frac{1}{2}}}\right] \\
&+ \frac{1}{\sin^2\theta_j\, \Delta\phi_k}\left[\frac{\nu_{j+\frac{1}{2},k+\frac{1}{2}} + \nu_{j-\frac{1}{2},k+\frac{1}{2}}}{2}\, \frac{B_{r:j,k+1} - B_{r:j,k}}{\Delta\phi_{k+\frac{1}{2}}}\right. \\
&\qquad\qquad \left. - \frac{\nu_{j+\frac{1}{2},k-\frac{1}{2}} + \nu_{j-\frac{1}{2},k-\frac{1}{2}}}{2}\, \frac{B_{r:j,k} - B_{r:j,k-1}}{\Delta\phi_{k-\frac{1}{2}}}\right],
\end{aligned}
\tag{27}
$$

where since the diffusivity ν is located on a staggered grid in both the θ and ϕ direction, it needs to be averaged to the correct location in the stencil. In order to apply the operator in a simple and efficient manner, and to allow the easy computation of the stable Euler time step size limit (see Sec. 3.7.2), we represent the operator as applied to the inner cells (without boundary conditions) as a symmetric self-adjoint sparse matrix in a custom DIA format (Bell & Garland 2008), where the matrix coefficients are given by

$$
\begin{array}{|c|c|}
\hline
a_{j,k-1} & \dfrac{\nu_{j+\frac{1}{2},k-\frac{1}{2}} + \nu_{j-\frac{1}{2},k-\frac{1}{2}}}{2\,\Delta\phi_{k-\frac{1}{2}}\,\Delta\phi_k\,\sin^2\theta_j} \\
\hline
a_{j-1,k} & \dfrac{(\nu_{j-\frac{1}{2},k+\frac{1}{2}} + \nu_{j-\frac{1}{2},k-\frac{1}{2}})\,\sin\theta_{j-\frac{1}{2}}}{2\,\Delta\theta_{j-\frac{1}{2}}\,\Delta\theta_j\,\sin\theta_j} \\
\hline
a_{j,k} & -\left[a_{j-1,k} + a_{j+1,k} + a_{j,k-1} + a_{j,k+1}\right] \\
\hline
a_{j+1,k} & \dfrac{(\nu_{j+\frac{1}{2},k+\frac{1}{2}} + \nu_{j+\frac{1}{2},k-\frac{1}{2}})\,\sin\theta_{j+\frac{1}{2}}}{2\,\Delta\theta_{j+\frac{1}{2}}\,\Delta\theta_j\,\sin\theta_j} \\
\hline
a_{j,k+1} & \dfrac{\nu_{j+\frac{1}{2},k+\frac{1}{2}} + \nu_{j-\frac{1}{2},k+\frac{1}{2}}}{2\,\Delta\phi_{k+\frac{1}{2}}\,\Delta\phi_k\,\sin^2\theta_j} \\
\hline
\end{array}
\tag{28}
$$

Boundary conditions
In the ϕ direction, we invoke a periodic boundary condition over the 2-cell overlap as described in Sec. 3.6.1. For the poles, we utilize the same finite-volume approach used in the advection scheme as described in Appendix A.

3.7.2. *Diffusion temporal schemes*

A major difficulty in integrating diffusion operators explicitly is the restrictive stable time step, making the computation expensive and often impractical. One way to avoid this is to use implicit methods, such as backward Euler scheme solved with iterative Krylov solvers (e.g. the preconditoned conjugate gradient (PCG) method) (Saad 2003) and/or multi-grid methods (Briggs et al. 2000). However, these methods can be complicated to implement and parallelize efficiently (especially on GPUs). Additionally, when used in cases where the time step is very large compared to the explicit Euler time step limit, they can become too inaccurate for production use (Dawes 2021).

Extended stability Runge-Kutta methods (also known as 'Super Time Stepping' (STS)) are a class of methods that are explicit and unconditionally stable (Verwer 1996). They are straight-forward to implement and easy to parallelize, while being competitive in computational performance to implicit methods. For example, in Caplan et al. (2017), we compared the solution and performance of the second-order RKL2 scheme (Meyer et al. 2014) compared to a PCG solver for a thermodynamic MHD model and found that the RKL2 method had equal or better performance to the PCG solvers, while yielding overall similar solution results. However, some solution oscillations were found in regions of high gradients in small grid cells (where the explicit Euler time step was very small compared to the overall time step). In Caplan et al. (2024a), a practical time step limit (PTL) was developed to be able to achieve much less oscillatory solutions for both STS (in that case the RKG2(3/2) scheme (Skaras et al. 2021)) and PCG methods for large time steps. It was found that using the PTL with RKG2(3/2) has similar performance to the PCG method, but yielded better solutions.

HipFT implements both RKL2 and RKG2(3/2) schemes for the diffusion operator (the explicit forward Euler scheme is also included for reference and testing). See Section 3 and Appendix B of Caplan et al. (2017) for details on the implementation of RKL2, and Appendix A of Caplan et al. (2024a) for RKG2(3/2). Both schemes are combined with the PTL sub-cycling procedure described in Caplan et al. (2024a). In practice, the PTL cycling is typically not triggered for most use cases of HipFT, with the notable exception of running HipFT as a map smoother (see Sec. 6.3).

The RKL2 and RKG2(3/2) yield similar accuracy and computational efficiency. For example, running the test case 2 of Eq. 19 with diffusion only (no flows) using the RKL2 scheme took 0.45 seconds for the diffusion advance (on an NVIDIA RTX 3090Ti GPU) with a final $\mathrm{HH}_{||}$ error of 8.0×10^{-5}, while using the RKG2(3/2) scheme took 0.55 seconds with a final $\mathrm{HH}_{||}$ error of 6.3×10^{-5}. So in this case, the RKL2 is a little faster than RKG2 but also a little less accurate. However, here the PTL condition was not activated. In some cases, the RKG2(3/2) scheme could activate less PTL cycles than RKL2, counterbalancing the performance differences. For reference, the same run using explicit forward Euler took 320 seconds with a final error of 1.3×10^{-5}. The error is less due to the much smaller time step (1.8×10^{-4} hr) where the STS methods ran in one full 672 hr time step. In HipFT, the RKG2(3/2) scheme is used by default.

3.8. *Validation of default methods*

Here, we show validations of the numerical schemes in HipFT using the test cases described in Sec. 3.1. We focus on the default/recommended methods chosen based on the discussion and tests described in the previous sections. The default advection method is the CS-WENO3(h) spatial scheme of Sec. 3.6.1 combined with the SSPRK(4,3) temporal scheme from Sec. 3.6.2. In Fig. 7, we show convergence results for running the test case 1 of Eq. 20 for a resolutions spanning $n_\phi = 256$ to $n_\phi = 4096$. For each resolution, we perform runs with a constant angular velocity in ϕ, a constant velocity in θ, and with both velocities active. We see that the code exhibits 3rd order accuracy as expected. The errors in the ϕ-direction are higher than those in the θ direction, due to the much higher velocities used in ϕ. In the θ direction, we see the convergence starting to diverge from third-order at the highest resolution tested. This may be due to the use of first-order upwinding for the pole-adjacent cells in the θ-direction or the second-order polar boundary conditions.

The default diffusion method is the finite central difference spatial scheme of Eq. 27, combined with the RKG2(3/2) temporal scheme adaptively cycled with the PTL condition described in Sec. 3.7.2. In Fig. 8, we show convergence results for running the initial 'soccer ball' profile of Eq. 19 for 672 hours with a diffusion coefficient of $500\,\mathrm{km}^2/\mathrm{s}$. We also show the same solution including advection with a constant angular velocity in ϕ that yields a full rotation (with no Strang splitting). We also show the error for runs using only advection over a full rotation with the soccer ball initial condition. We see that the errors associated with advection are significantly larger than those for diffusion. However, since the CS-WENO3(h) scheme exhibits third-order accuracy, the errors intersect at a high resolution, resulting in the diffusion error dominating. This transition happens at an extremely high resolution of $n_\phi = 8192$, which is considerably larger than a typical HipFT run. If such high resolutions eventually become needed, a 4th-order spatial diffusion operator would be straight-forward to implement.

The validation tests performed in this paper all used a uniform grid. However, HipFT is capable of using a non-uniform grid, which can be useful to coarsen the grid near the poles to increase the flow CFL time step limit and for performing high-resolution evolutions of localized regions on the Sun, such as active regions or coronal holes. Since the current default use cases of HipFT use a uniform grid, we leave the formal validation of running on non-uniform grids for a future publication.

4. CODE IMPLEMENTATION AND PERFORMANCE

HipFT is written in modern Fortran (Curcic 2020) including features of the new Fortran 2023 standard (ISO 2023). It is designed to be easily modified and expanded, with a modular structure and clear, descriptive names

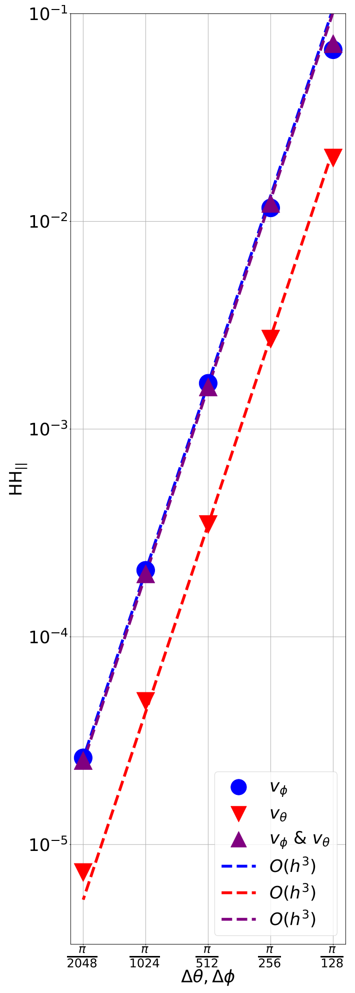


Figure 7. Convergence of test case 2 (Eq. 20) using the default numerical schemes of HipFT. The blue marks are for a ϕ-rotation with $v_\theta = 0$, the red marks are for a constant v_θ, with $v_\phi = 0$, and the purple marks are for runs using both v_ϕ and v_θ. For runs with a zero-value velocity component, the resolution of the zero-velocity direction is set to 512 for n_θ and 1024 for n_ϕ. The third-order convergence of the WENO3-CS(h) scheme can be seen in all three runs.

for variables, function, and subroutines. It has been tested to work on a variety of compilers and hardware targets, including x86 and Arm CPUs, as well as NVIDIA and Intel GPUs.

4.1. *Parallelism*

HipFT uses a hybrid parallelization scheme, where MPI is used to spread groups of map realizations across compute units (within or across distributed nodes), while Fortran's `do concurrent` (DC) is used to compute a group of realizations in parallel within a compute unit. This can take the form of multi-threading on multi-core CPUs, or parallel computation on GPUs. For example, when only modeling one realization, no MPI parallelism is used, and the code is run either multi-threaded across local CPU cores or on a single GPU. When using MPI to run many realizations across multiple GPUs, the code assumes the code is launched with 1 MPI rank per GPU to set the GPU device number. For multi-core CPUs, the number of MPI ranks and parallel threads should be set carefully using optimal affinity options, which can vary from system to system. For example, on an AMD EPYC ROME CPU with 4 NUMA domains, it is optimal to set the CPU to 4xNUMA mode and run HipFT with 16 threads per NUMA domain with 4 MPI ranks (one per NUMA). These kinds of affinity considerations are a common complication in running hybrid MPI+multi-threading applications (Arul & Huang 2015), which is mostly avoided when running HipFT on GPUs.

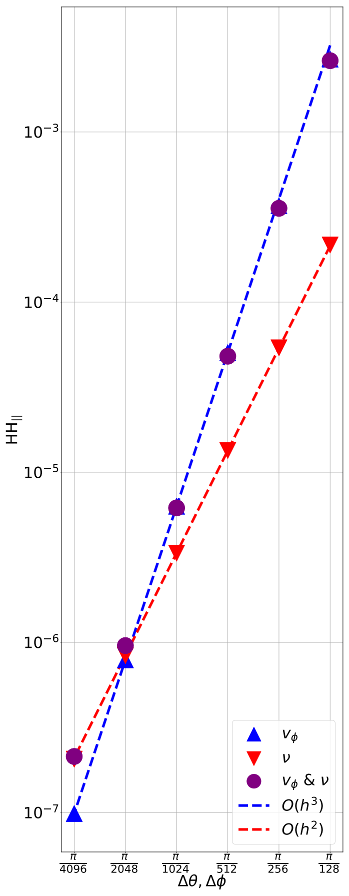


Figure 8. Convergence of test case 1 (Eq. 19) using the default numerical schemes of HipFT. The purple marks are with diffusion and advection, the red marks are with diffusion only, and the blue marks are with advection only. The advection error dominates the total error for most resolutions. However, since the diffusion error is second-order while the advection error is third-order, eventually at very high resolution, the diffusion error starts to dominate.

A key issue when running codes on GPUs is data management. Most GPUs have their own local memory separate from the CPU's memory. Accessing/transferring data from one to the other is often much slower than the memory speed of the GPU, sometimes by orders of magnitude. Two strategies to avoid this overhead are to either manually move the data to and from the GPU in a way that minimizes the number and size of the transfers, or use an automatic memory management system. Automatic memory management can be a compiler/software feature, run-time feature, and/or driver-level feature. Some GPUs and APUs have special hardware that allow the CPU and GPU to directly access each others' memory, often referred to as 'unified memory', which can be used to greatly improve performance of automatic memory management.

In order to keep HipFT as portable as possible, we manually manage the data movement between GPU and CPU using the OpenMP API, specifically OpenMP Target data directives (Deakin & Mattson 2023). This allows HipFT to run efficiently on compilers/hardware that do not support an automatic memory management system. For more information on the GPU parallelization of HipFT and its portability across GPU vendors, see Caplan et al. (2024b).

While we leave the details of building the code to the installation guide in the HipFT repository, it is important to point out that the ubiquitous GCC Fortran compiler `gfortran` does not currently support directly multi-threading DC loops. However, it does contain an automatic multi-threading option activated during compilation to parallelize HipFT. We refer the user to the sample build scripts in the repository for more details.

4.2. *Performance*

Here, we test the performance of HipFT on both server and consumer CPUs and GPUs. For the first test, we use the multi-realization example run provided in the github repository in the `examples/flux_transport_1rot_flowAa_diff_r8` folder. This run uses analytic flows, flow attenuation, and diffusion to evolve a starting HMI Carrington map for 674 hours at the default 1024×512 resolution. Eight realizations are computed, varying the diffusivity and flow attenuation values, allowing us to run the code with up to 8 MPI ranks.

In Fig. 9 we show timing results for various server CPUs and GPUs, as well as some consumer GPUs. The portion of time each part of the HipFT calculation required is indicated. Due to large variations in file system speeds across the systems, we omit the I/O time from the plots, but note that the total time for I/O on a locally-attached drive for these runs was small (less than 1 second). We see that runs on a single professional GPU are faster than those on

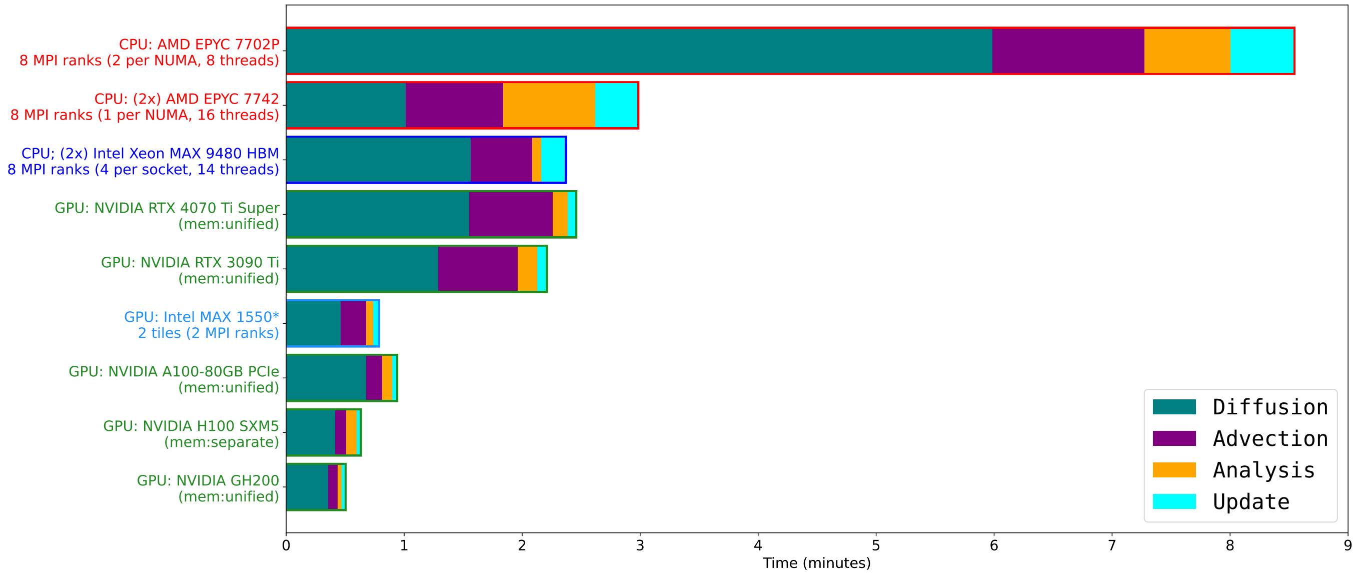


Figure 9. Timing results for running HipFT on the example case `flux_transport_1rot_flowAa_diff_r8` provided in the git repository on a variety of CPU and GPU hardware. The time taken for each part of the code is indicated (and I/O time is not included). Only a single CPU node or single GPU is tested here. For CPUs, the eight realizations of the run is spread across the sockets and/or NUMA domains using MPI, with the number of threads per MPI rank indicated. On NVIDIA GPUs, the memory management method used for each GPU is indicated (see Caplan et al. (2024b) for details). For the Intel GPU, MPI is used to spread the realizations across compute tiles.

the multi-core CPUs. We highlight the result of being able to run on both NVIDIA and Intel GPUs, demonstrating the performance portability of the Fortran standard implementation (see Caplan et al. (2024b) for details). We also note that the inexpensive consumer GPUs ran as fast as a modern two-socket CPU server with over 100 cores. For reference, the same run performed on a modern desktop CPU (Ryzen 9700X) with 8 dual-threaded cores (running with 8 MPI ranks, 2 threads per rank) took $\sim$ 23 minutes. Therefore, adding an inexpensive consumer GPU to a modern personal desktop computer can yield over an $8\times$ speedup of HipFT (even more for an older desktop CPU).

As described in Sec. 4.1, HipFT can use MPI to spread realizations across compute units. To test how HipFT can scale for many realizations, we add redundant realizations to the example used above for a total of 128 realizations and run it across multiple GPUs within a compute node, and across GPU/CPU nodes. The resulting timings for a variety of CPUs and GPUs are shown in Fig. 10. We see that the code scales well across realizations for multiple GPUs (both NVIDIA and Intel) within a node, and across CPU and GPU nodes. The scaling across GPU nodes starts to taper off at larger number of nodes, but the overall performance is much higher than the CPU nodes.

HipFT can be run at any practical resolution, but the higher the resolution, the more computationally expensive the integration steps become, and more importantly, the smaller the stable advection time step becomes, yielding many more computational steps. The non-uniform grid capabilities of HipFT can be used to reduce the computational time, by coarsening the grid near the poles (increasing the stable flow time step) and/or focusing the high-resolution to limited sections of the map. A full description and testing of these capabilities is beyond the scope of the current paper. However, in order to get a feel on the computation time at very high resolutions in the worst case, we test HipFT at 4096×2048 resolution. We modify the HipFT git repository's example run

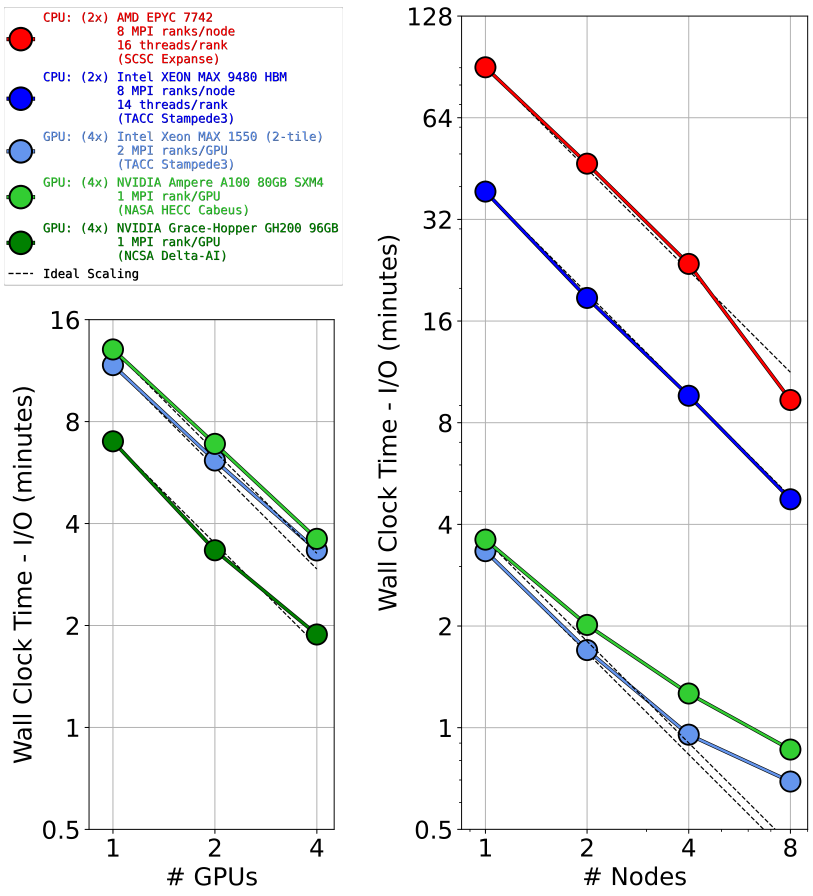


Figure 10. Timing results for running HipFT on the example case `flux_transport_1rot_flowAa_diff_r8`, but with a total of 128 realizations. Due to differences in the file systems of the machines tested, we exclude the I/O time. For CPUs, the realizations are spread optimally across the sockets and/or NUMA domains per node as indicated. The most performative memory management method for each GPU is used. For the Intel GPUs, the realizations are spread across each GPUs two compute tiles (see Caplan et al. (2024b) for details). The scaling results for GPUs within a single node are shown on the left, and the results across CPU and GPU nodes is shown on the right. The CPU and Intel GPU results used the Intel IFX compiler 2025.0.0, while the NVIDIA GPU results used the nvfortran compiler 24.7/9.

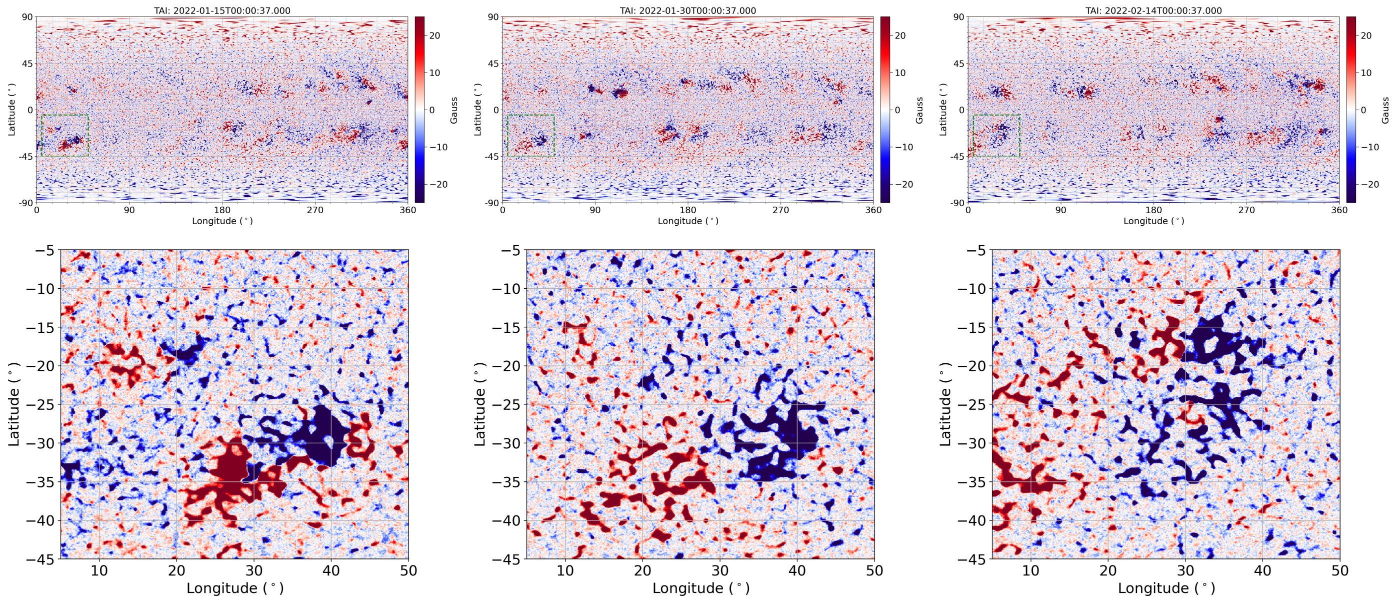
`flux_transport_1yr_flowCAa_diff300_assimdata_rfe` to run at 4096×2048 resolution, set the added random flux amount $\Phi/\text{hr} = 650 \times 10^{21}$ Mx/hr (to compensate for the higher unsigned flux at higher resolution), set the diffusion to $\nu = 200\,\text{km}^2/\text{s}$, and only run it for two Carrington rotations (56 days). We also generated the required high-resolution MagMAP and ConFlow data. The run completed in hours on an NVIDIA A100 80GB PCIe GPU, which is a much larger computational time than that of the default resolution of HipFT ($\sim$ 6 minutes). However, the run time is still over 50× faster than real-time, opening the door for high resolution studies in the future. In Fig. 11, we show maps from the high resolution run with zoomed-in views of an active region. Exploration and analysis of high-resolution HipFT runs like these will be discussed in a future publication.

5. CODE USE

For full instructions on running the code and all input options, see the documentation included in the github repository. Here, we provide a brief overview of the basic input, output, and post processing features of HipFT.

5.1. *Inputs*

HipFT uses a Fortran ‘namelist’ for input parameters. This is a lightly formatted text file containing the desired input parameters and their values. One key advantage of using a namelist is that only input parameters differing from



TAI: 2022-01-15T00:00:37.000
TAI: 2022-01-30T00:00:37.000
TAI: 2022-02-14T00:00:37.000
Latitude (°)
Longitude (°)
Gauss

Figure 11. Map output of HipFT running at 4096 × 2048 resolution. The run is adapted from the example case `flux_transport_1yr_flowCAa_diff300_assimdata_rfe` in the git repository, with the diffusion lowered to $\nu = 200\,\text{km}^2/\text{s}$, the random flux amplitude risen to $\Phi/\text{hr} = 650 \times 10^{21}\text{Mx/hr}$, and the simulation time lowered to 56 days. High resolution versions of the required MagMAP database of assimilation data and convective flows from ConFlow were produced for the run. The maps of the run are shown at ~ 15 (left), ~ 30 (middle), and ~ 45 (right) days into the simulation. The active region highlighted by the green boxes is shown zoomed-in in the bottom row.

the default values need to be specified, enabling concise, easy-to-read input files for most use cases. Additionally, the code writes out the full namelist of all parameters to a file after reading the input, providing a complete record of every parameter used in the run.

Other possible inputs for HipFT include 2D HDF5 (Group 2024) map files for the initial map, as well as options for a custom spatially-dependent diffusion coefficient, source term, and/or flows. When using data assimilation and sequences of flows from file (such as when using MagMAP and/or ConFlow output), a CSV file is read in which directs HipFT to load the correct files at the desired times. HipFT has the option to multiplicatively flux balance the input map and/or assimilated data. Note that there are currently no interpolation/re-binning capabilities in the HipFT code, so any necessary map processing for inputs must be done externally (tools to do this are provided in the OFT repository).

5.2. *Outputs*

Numerous outputs are provided by HipFT. When the code is launched, it outputs the namelist of parameters that will be used in the run, as well as the initial map. For validation runs, the analytic solution is also written out. When using multiple realizations, a text file is outputted listing all parameters for each realization in a table for reference.

During the run, basic information about the current run time, time step, etc. are written to the terminal in order to monitor the run progress. The analysis quantities described in Sec. 2.6 are written to a history file for each realization. Additionally, quantities related to the numerical methods (e.g. time step, stability limits, diffusion PTL cycles, etc.) are written to a separate history file across realizations. All history files are appended and closed at each step, allowing them to be plotted (see Sec. 5.3) throughout the run. If map output was activated, the maps are written at their chosen cadence and a list of the maps' output simulation time and filenames are written to a text file. For multiple realization runs, the maps are written as a 3D HDF5 file, with each slice corresponding to a realization. If there is only 1 realization, the code can either write the maps as 3D HDF5 files with a unitary third dimension, or as 2D HDF5 files. In addition to the maps, the code can also be set to output the flows at the same cadence as the maps. This can be useful to see what the combined flows in the code were at specific times (for example when combining analytic flow profiles with flows from file and flow attenuation).

When the run is complete, HipFT outputs a text file that shows timings for each part of the code for each MPI rank, including a summary of all ranks. This summary is also outputted to the terminal at the end of a run. Additionally, the code writes out the final map (the output of the initial and final maps are independent of whether or not there is a map output cadence set). For all map output, an option to multiplicativly balance the flux is available.

5.3. *Post processing and analysis*

In the HipFT repository, we have provided several python scripts to help analyze and process the results of a HipFT run. This includes plotting the quantities (and derived quantities) in the history files, plotting the output maps and making a movie of them, and generating and plotting a 'butterfly diagram'. Additional scripts are provided that add UTC/TAI dates and times to the run outputs and plots, extract realization slices, print history summaries, compare runs to each other, compare maps to each other, read map data into python `numpy` arrays, and get map values of a series of (θ, ϕ) in-situ points through interpolation. To facilitate comparisons to other models (or to analyze data derived maps), we provide a script that generates a HipFT history file from a sequence of HDF5 maps. Scripts to run the validation tests of Sec. 3.8 are also included. For details on how to use the various tools, see the help documentation for each script by calling them with the '-h' flag.

In addition to the HipFT tools described above, the OFT repository contains scripts and tools that can easily post-process HipFT input/output maps. These scripts allow for operations such as binning to a different resolution in a flux-balance preserving manner, flux balancing, and/or smoothing. These tools can also be used to remesh HipFT output, which can be useful for models and analysis requiring lower resolution, smoothed maps, and/or alternative grids. For example, once processed, the maps generated from HipFT can be directly used with the SWiG empirical solar wind generator model[8] and/or the CORHEL-CME thermodynamic coronal mass ejection generator model[9].

6. USE CASES

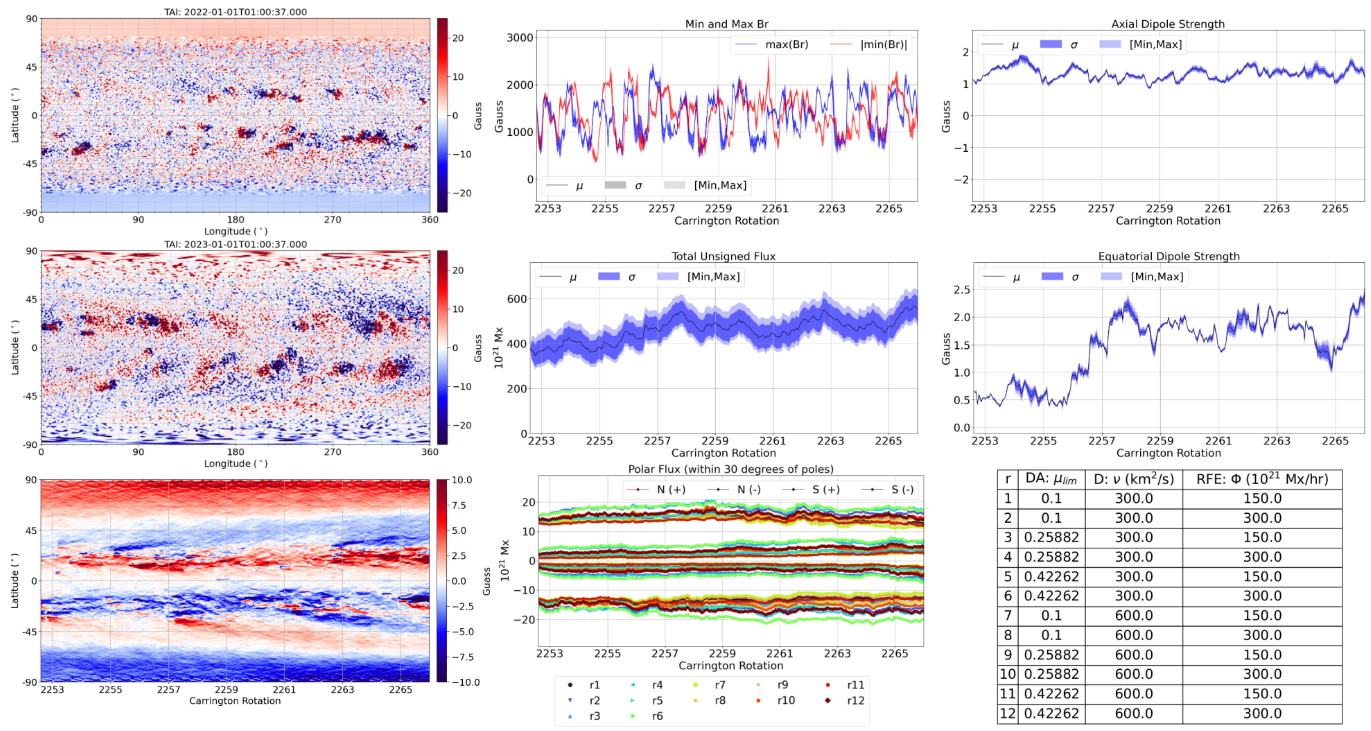
The use of HipFT in the context of the OFT model, including production runs and detailed analysis, will be described in a forthcoming publication (Upton 2025). In this section, we briefly highlight some use cases of HipFT.

6.1. *Flux transport with convective flows, data assimilation, and random flux emergence*

One primary use of HipFT is to run flux transport simulations in order to provide full-Sun magnetic maps. To facilitate an example production run of this kind, we have provided a ready-to-use set of convective flow ConFlow simulation data (28 days at 15 minute cadence) as well as a full year of HMI B_r data at 1 day cadence processed with MagMAP (see Sec. 7). The HipFT input file to run a full year simulation using this data is provided in the `examples/flux_transport_1yr_flowCAa_diff300_assimdata_rfe` folder of the HipFT repository. The run uses the default suggested values for this kind of simulation, including flow attenuation (Eq. 5), diffusion, data assimilation and random flux emergence.

Here, we show results of running the one year example case, adding some multiple realization parameters. Specifically, we set the data assimilation μ cutoff (see Eq. 9) to $5.74°$, $15°$, and $25°$ away from disk limb, the diffusivity ν to $300\,\text{km}^2/\text{s}$ and $600\,\text{km}^2/\text{s}$, and the random flux emergence total unsigned flux per hour to 150×10^{21} Mx/hr and 300×10^{21} Mx/hr. The combination of these parameters results in 12 realizations. In Fig. 12, we show some of the post processing outputs from the run including derived quantities, maps, and butterfly diagram plots. We see variations across realization combinations, with some combinations showing similarities while others can differ significantly. The use of multi-realization runs like this is a useful tool in exploring the dynamics of flux transport. A thorough analysis of a full solar cycle run (with a different set of realization parameters) will be described in a forthcoming publication (Upton 2025).

[8] https://github.com/predsci/swig
[9] https://ccmc.gsfc.nasa.gov/models/CORHEL-CME~1



r	DA: μ_{lim}	D: ν (km^2/s)	RFE: Φ (10^{21} Mx/hr)
1	0.1	300.0	150.0
2	0.1	300.0	300.0
3	0.25882	300.0	150.0
4	0.25882	300.0	300.0
5	0.42262	300.0	150.0
6	0.42262	300.0	300.0
7	0.1	600.0	150.0
8	0.1	600.0	300.0
9	0.25882	600.0	150.0
10	0.25882	600.0	300.0
11	0.42262	600.0	150.0
12	0.42262	600.0	300.0

Figure 12. Samples of post processing output for the example case `flux_transport_1yr_flowCAa_diff300_assimdata_rfe` included in the HipFT repository, but with 12 realizations spanning diffusion, data assimilation, and amount of added random flux quantities. The initial and final B_r maps for realization 1 are shown in the top and middle left respectively with the butterfly diagram shown in the bottom left. Various derived quantities and metrics are shown in the center and right columns, with the center right showing all realizations, while the others show realization summary plots. The bottom right shows the table of realization parameters generated by the post processing.

6.2. *Generating time-dependent boundary conditions for an MHD model*

The full Sun maps generated from a HipFT run of the kind described in Sec. 6.1 can be used to drive time-dependent MHD models of the solar corona and heliosphere (Mason et al. 2023; Lionello et al. 2023; Feng et al. 2023; Downs et al. 2024). For example, Downs et al. (2024) used HipFT to generate several months of full-Sun maps at one hour cadence, and used them to drive the lower boundary condition of a thermodynamic MHD model for 32 days, assimilating data in near real time. The simulation was used to generate a running prediction of how the corona would look during the 2024 total solar eclipse[10]. In order to avoid strong jumps in the boundary from emerging active regions, a custom ramped weight map was used during data assimilation. The maps were then processed using a flux-preserving re-meshing scheme as well as B_r-dependent smoothing to ensure the model could resolve the evolving map structures. In Fig. 13 we show a time sequence of HipFT processed maps along with forward modeled EUV images of the time-dependent MHD model. These images use the same viewer position but each snapshot is separated by several days, illustrating how the solution evolves as new data from the sequence of HipFT maps is continuously assimilated. For full details and results of this novel simulation, including details on how HipFT was utilized to generate the boundary conditions, see Downs et al. (2024).

6.3. *Flux-preserving smoothing tool*

The efficiency of the diffusion advance in HipFT allows it to be used as a fast map smoothing tool. Smoothing magnetic maps is often necessary for models to properly resolve the magnetic structure (Caplan et al. 2019). While local filter algorithms can be used, they can suffer from aliasing artifacts and flux imbalance. By using a surface diffusion advance, these difficulties are avoided. The grid-based diffusivity shown in Sec. 2.2 allows the use of the

[10] https://www.predsci.com/eclipse2024

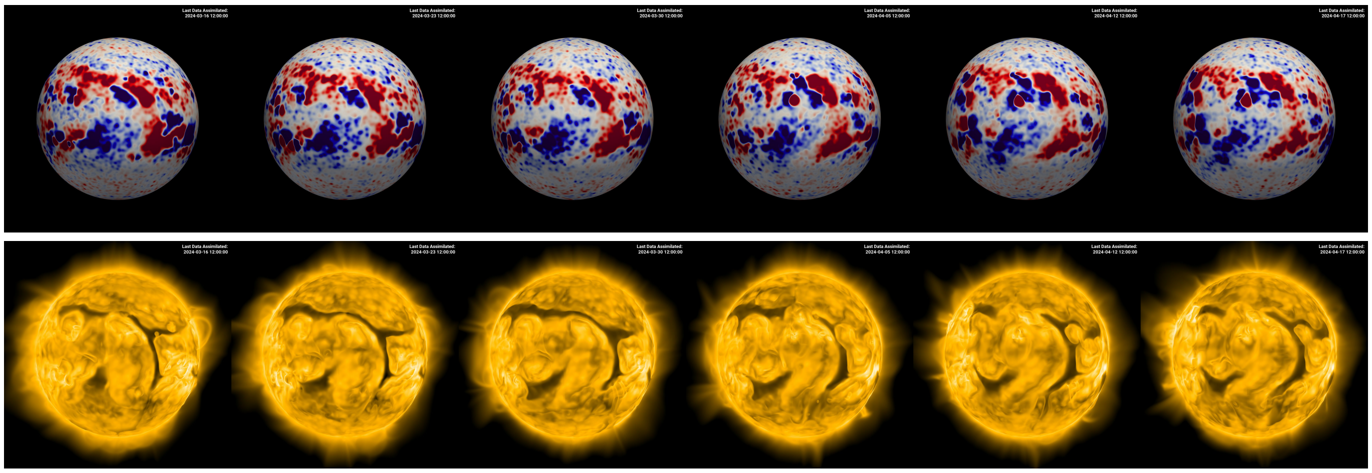

Last Data Assimilated: 2024-03-16 12:00:00
Last Data Assimilated: 2024-03-23 12:00:00
Last Data Assimilated: 2024-03-30 12:00:00
Last Data Assimilated: 2024-04-05 12:00:00
Last Data Assimilated: 2024-04-12 12:00:00
Last Data Assimilated: 2024-04-17 12:00:00

Figure 13. Example using HipFT to drive a time-dependent MHD model (Downs et al. 2024). The top row shows the processed HipFT B_r maps, while the bottom row shows the corresponding forward-modeled EUV images derived from the data-driven MHD model. The view is from Earth's perspective on April 8th, 2024 18:42 UT, and the model snapshots have timestamps of 12 UT on 3/16, 3/23, 3/30, 4/5, 4/12, and 4/17 2024 respectively. The time-evolving model exhibits dynamics similar to those observed on the Sun that cannot be generated by non-evolving static relaxation models. Images taken from www.predsci.com/eclipse2024.

minimum amount of smoothing necessary for different parts of the grid; however, a constant diffusivity can also be used. This map smoothing capability of HipFT is used in OFT's map processing tools. In Fig. 14, we show the result of using HipFT to smooth a map with grid-based diffusivity. This smoothing is very fast (in this case, the 512×1024 map took 0.25 seconds on a RTX 3090Ti GPU). We note that the PTL time step limit described in Sec. 3.7.2 is critical in obtaining an accurate solution robustly in these runs as there is no stability limit on the time step (since the STS diffusion advance is unconditionally stable).

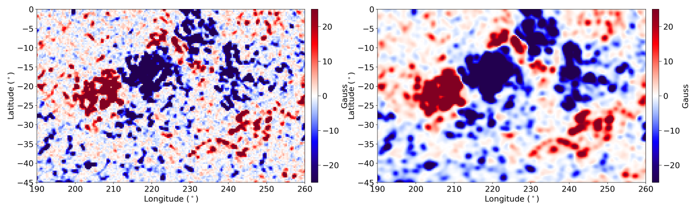


Figure 14. Example of using HipFT as a flux-preserving magnetogram smoother. A zoomed-in portion of the last output map for realization 1 of the test used in Sec. 6.1 is shown on the left, while the same map is shown on the right after smoothing the map using the grid-based diffusivity of Eq. 7 with $\alpha_\nu = 0.5$.

7. AVAILABILITY

HipFT is open source and publicly available and developed on GitHub at https://github.com/predsci/HipFT. The repository includes detailed instructions on building and running the code on multiple types of systems. As the computational core of OFT, the HipFT repository is also included as a submodule of the OFT GitHub repository at https://github.com/predsci/OFT. We have also set up a Zenodo data set at https://zenodo.org/records/11205509 where a Carrington rotation of ConFlow convective flow maps and one full year of MagMAP HMI data for data assimilation is provided. This allows testing HipFT in a true scientific context independent of installing and running ConFlow and MagMAP. HipFT is continually developed along with the overall OFT code suite, with public suggestions, updates, and modifications being welcome.

8. SUMMARY

We have introduced the open source High-performance Flux Transport code HipFT, which serves as the computational core of the Open-source Flux Transport (OFT) model. It is a Fortran code that implements highly accurate and efficient numerical methods to advance advection, diffusion, sources, and data assimilation. It is designed to be modular, incorporating numerous model and numerical method options, and allows users to write extensions easily. HipFT can compute multiple realizations across many model parameters within a single run to create ensembles of maps for uncertainty quantification and parameter studies. By writing the code using Fortran standard language parallelism (using the `do concurrent` construct), it can run in parallel on multi-core CPUs and on GPUs. MPI is used to parallelize across realizations, allowing the code to run across multiple multi-CPU and multi-GPU compute nodes. We have described the model, numerical methods, code implementation, and analysis tools of HipFT. The numerical methods were validated on known solutions, and the performance on CPUs and GPUs was tested. The code is available and developed on GitHub, and is included as a submodule in the OFT model suite.

The OFT model suite contains two other important codes needed for realistic runs of HipFT, which are ConFlow and MagMAP. ConFlow is used to generate time-dependent super granular convective tangential surface flows and will be describe in detail in a forthcoming paper (Attie 2025). MagMAP is a python library that is used to obtain, process, and prepare observations of the solar surface magnetic field for use in data assimilation in HipFT. A full description of the OFT model and its use, including details of the MagMAP code will be described in a forthcoming paper (Upton 2025).

ACKNOWLEDGMENTS

Work at Predictive Science Inc. was supported by the NASA LWS Strategic Capabilities Program (grant 80NSSC22K0893), the NSF PREEVENTS program (grant ICER1854790), and the NSF/NASA SWQU program (grants AGS 2028154 and 80NSSC20K1582). It also utilized the Cabeus system at NASA's High-End Computing Capability (HECC) through NASA grant's 80NSSC20K0192's NASA Advacned Supercomputing division request SMD-24-72380598, as well as the Expanse system at the San Diego Supercomputing Center (SDSC), the Stampede3 system at the Texas Advanced Computing Center (TACC), and the Delta-AI system at the National Center for Supercomputing Applications (NCSA) through allocation TG-MCA03S014 from the Advanced the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

C.N.A. was supported in part by the NASA SWQU grant listed above and the NASA competed Heliophysics Internal Scientist Funding Model (ISFM).

C.J.H. is partially supported by Air Force Office of Scientific Research (AFOSR) tasks 22RVCOR012 and 25RVCOR001. The views expressed are those of the authors and do not reflect the official guidance or position of the United States Government, the Department of Defense or of the United States Air Force.

APPENDIX

A. POLAR BOUNDARY CONDITIONS FOR ADVECTION AND DIFFUSION OPERATORS

For the polar boundary condition for the advection and diffusion operator, we use a finite volume method. We want to evaluate $\nabla\cdot F$ at the pole, where for advection, $F_a = \vec{v}\,B_r$, and for diffusion, $F_d = \nu\,\nabla B_r$. From the divergence theorem, we have

$$\int\int \nabla\cdot F\,dA = \int_C F\,ds.$$

We treat the region at the pole out to the first internal half-mesh point ($\theta = \Delta\theta/2$) as a single cell. F is computed at the half-mesh points around the pole using the upwinding scheme for advection (see Sec. 3.6.1), and central differencing for diffusion (see Sec. 3.7.1). Since only the flux in the theta direction contribute to the flux in and out of the polar cell, we ignore the ϕ direction in F. The line integral above is evaluated as

$$\int_C F\,ds = \int_0^{2\pi} F\,\sin\theta\,d\phi,$$

which at $\theta = \Delta\theta/2$, numerically is integrated along ϕ as

$$\int_0^{2\pi} F\,\sin\theta\,d\phi = \sin\left(\frac{\Delta\theta}{2}\right)\sum_{k=2}^{N_\phi-1} F_k\Delta\phi_k.$$

For the surface integral, since the polar cap is treated here as a single cell, the F is spatially independent within the cell, so it can be pulled out of the integral to yield

$$\int\int_{\text{pole}} \nabla\cdot F\,dA \approx (\nabla\cdot F)_{\text{pole}}\int\int dA = (\nabla\cdot F)_{\text{pole}}\int_0^{2\pi}\int_0^{\Delta\theta/2}\sin\theta\,d\theta\,d\phi.$$

Evaluating the integral yields

$$\int_0^{2\pi}\int_0^{\Delta\theta/2}\sin\theta\,d\theta\,d\phi = 2\,\pi\,\left[-\cos\theta\right]_0^{\Delta\theta/2} = 2\,\pi\left[1-\cos\left(\frac{\Delta\theta}{2}\right)\right].$$

Combining this with the line integral above, and using small angle approximations, we get the polar operator

$$\nabla\cdot(\vec{v}\,B_r)_{\text{pole}} \approx \frac{\sin\left(\frac{\Delta\theta}{2}\right)}{2\,\pi\left[1-\cos\left(\frac{\Delta\theta}{2}\right)\right]}\sum_{k=2}^{N_\phi-1} F_k\Delta\phi_k \approx \frac{2}{\pi\,\Delta\theta}\sum_{k=2}^{N_\phi-1} F_k\Delta\phi_k$$

where F_k is evaluated on the half mesh points next to the pole.

B. A 3RD-ORDER WEIGHTED ESSENTIALLY NON-OSCILLATORY SCHEME ON A NON-UNIFORM GRID

The WENO3-CS(h) (Cravero & Semplice 2015) is implemented using a left and right flux as in the upwinding scheme, but with different calculations of the flux F. Since $F_{i+1/2}$ is a right shifted version of $F_{i-1/2}$, only the calculation of $F_{i-1/2}$ is required to compute along each dimension over the whole domain. $F_{i-1/2}$ is calculated by first separating the left and right moving fluxes as:

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

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

$$p_0^- = (1 + D^{\mathrm{c/cm}}_{i-3/2})\,\mathrm{LM}_{i-1} - D^{\mathrm{c/cm}}_{i-3/2}\mathrm{LM}_{i-2}, \qquad p_0^+ = (1 + D^{\mathrm{c/cp}}_{i})\,\mathrm{LP}_{i} - D^{\mathrm{c/cp}}_{i}\mathrm{LP}_{i+1},$$
$$p_1^- = D^{\mathrm{c/cm}}_{i}\,\mathrm{LM}_{i-1} + D^{\mathrm{c/cp}}_{i-3/2}\mathrm{LM}_{i}, \qquad p_1^+ = D^{\mathrm{c/cp}}_{i-3/2}\,\mathrm{LP}_{i} + D^{\mathrm{c/cm}}_{i}\mathrm{LP}_{i-1},$$

where the D values 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:

$$w_0^- = \frac{D^{\mathrm{p/t}}_{i-3/2}}{(\epsilon_w + \beta_0^-)^2}, \qquad w_1^- = \frac{D^{\mathrm{cm/t}}_{i-3/2}}{(\epsilon_w + \beta_1^-)^2}, \qquad w_0^+ = \frac{D^{\mathrm{m/t}}_{i-1/2}}{(\epsilon_w + \beta_0^+)^2}, \qquad w_1^+ = \frac{D^{\mathrm{cp/t}}_{i-1/2}}{(\epsilon_w + \beta_1^+)^2}, \tag{B1}$$

where the βs are called 'smoothness indicators' which describes the smoothness of the region, and ϵ_w is a small value to avoid division by zero. In other WENO3 schemes, the value of ϵ_w is often set to an arbitrary value smaller than the typical solution value (Shu 2009). However, the WENO3-CS(h) scheme used here sets $\epsilon_w = h$ where h is the local cell spacing. This modification avoids the drop in accuracy near critical points, keeping the scheme 3rd-order. In Fig. 15, we show convergence plots for the Upwind, WENO3 with a constant $\epsilon_w = 10^{-12}$, and WENO3-CS(h) schemes running test case 1 (Eq. 20) with $v_\theta = 0$. We see that the WENO3-CS(h) scheme displays third-order accuracy, while using the standard WENO3 scheme with $\epsilon_w = 10^{-12}$ only exhibits second-order accuracy.

The βs are defined as:

$$\beta_0^- = 4\,(D^{\mathrm{c/cm}}_{i-3/2}\,(\mathrm{LM}_{i-1} - \mathrm{LM}_{i-2}))^2, \qquad \beta_0^+ = 4\,(D^{\mathrm{c/cp}}_{i}\,(\mathrm{LP}_{i+1} - \mathrm{LP}_{i}))^2,$$
$$\beta_1^- = 4\,(D^{\mathrm{c/cp}}_{i-3/2}\,(\mathrm{LM}_{i} - \mathrm{LM}_{i-1}))^2, \qquad \beta_1^+ = 4\,(D^{\mathrm{c/cm}}_{i}\,(\mathrm{LP}_{i} - \mathrm{LP}_{i-1}))^2,$$

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

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

Fig. 16 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 (LLF) (Shu 2009) is used:

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

where α_i is the maximum velocity taken over a relevant range of the domain (Li 2006):

$$\alpha_i = \max\{|v_{i-5/2}|, |v_{i-3/2}|, |v_{i-1/2}|, |v_{i+1/2}|, |v_{i+3/2}|, |v_{i+5/2}|\}$$

Fig. 16 shows a flow chart schematic for each WENO3 component that resides on the grid and what previous calculation on the grid each subsequent component relies on.

C. DERIVATION OF ANALYTIC THETA SOLUTION

For advection in the θ direction in spherical coordinates, we have:

$$\frac{\partial f}{\partial t} = -\nabla_s \cdot (v_\theta\, B_r) = -\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}(\sin\theta\, B_r\, v_\theta).$$

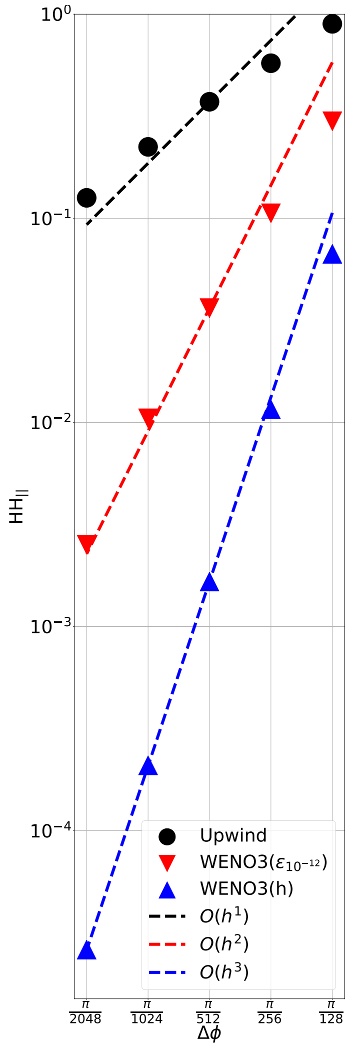


Figure 15. Convergence of the ϕ-rotation test case 1 (Eq. 20) with a fixed θ-resolution of 512, and with $v_\theta = 0$ for upwinding, WENO3 with a constant $\epsilon_w = 10^{-12}$, and WENO3-CS(h). We see that the WENO3(ϵ_w) exhibits 2nd-order accuracy, while the WENO3-CS(h) exhibits 3rd-order accuracy.

If we choose v_θ to be constant this becomes

$$\frac{\partial f}{\partial t} = -v_\theta \frac{1}{\sin\theta}\frac{\partial}{\partial\theta}(\sin\theta\, B_r) \tag{C3}$$

A translational function of the form $B_{r0}(\theta - v_\theta\, t)$ won't be a solution due to the terms involving $\sin\theta$. Instead, we define

$$B_r(\theta, t) = \frac{1}{\sin\theta}\, B_{r0}(\theta - v_\theta\, t).$$

Calling $g \equiv \theta - v_\theta\, t$, we have

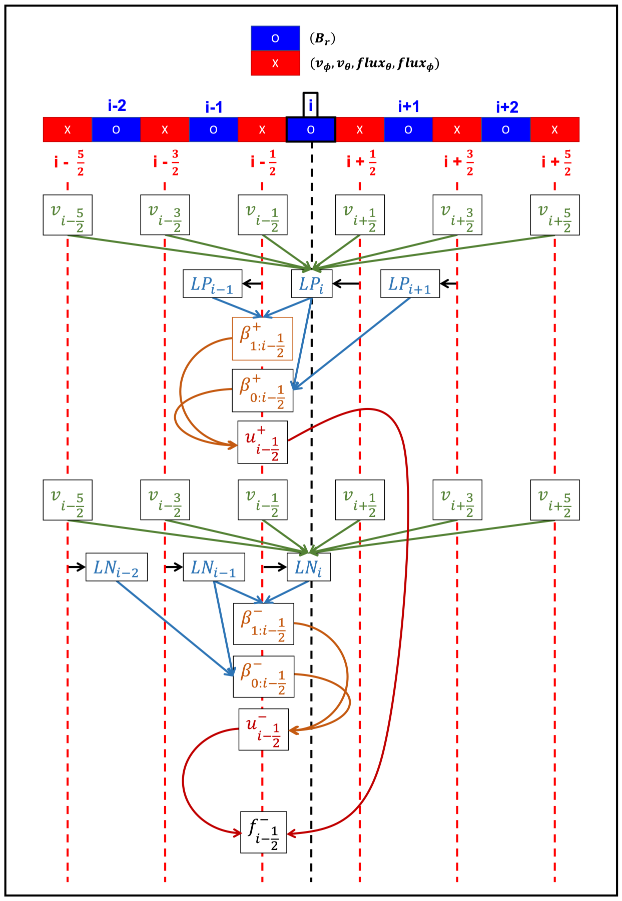
$$B_r(\theta, t) = \frac{1}{\sin\theta}\, B_{r0}(g),$$

which when inserted into Eq. C3 yields

$$\frac{\partial Br}{\partial t} = -\frac{v_\theta}{\sin\theta}\frac{\partial B_{r0}(g)}{\partial g}.$$

The spatial term can be written as

$$-v_\theta \frac{1}{\sin\theta}\frac{\partial}{\partial\theta}(B_{r0}(g)),$$



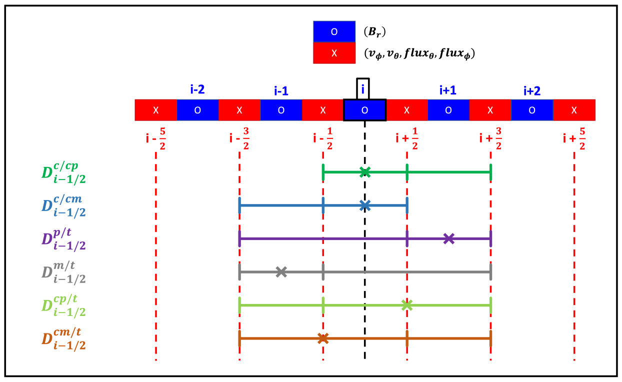


Figure 16. Top: Schematic of the WENO3-CS(h) scheme described in this section, showing the quantities used to calculate the left-side flux f_{i-1} and where they reside on the grid. Bottom: Schematic showing the values and locations of the non-uniform grid spacing used to compute the D grid factors in the scheme.

where we have

$$\frac{\partial B_{r0}(g)}{\partial\,\theta} = \frac{\partial B_{r0}(g)}{\partial\,g}.$$

Inserting this into Eq. C3, we see that both sides equate to $-(v_\theta/\sin\theta)\,\partial B_{r0}/\partial g$ so any profile $B_{r0}(\theta)$ makes $(1/\sin\theta)\,B_{r0}(\theta - v_\theta\,t)$ a solution.

REFERENCES

Altschuler, M. D., & Newkirk, G. 1969, SoPh, 9, 131

Arge, C. N., Henney, C. J., Hernandez, I. G., et al. 2013, in AIP Conference Proceedings (AIP), 11–14

Arge, C. N., Henney, C. J., Koller, J., et al. 2010, in AIP Conference Proceedings (AIP)

Arul, J. M., & Huang, C.-C. 2015, in 2015 International Symposium on Next-Generation Electronics (ISNE), Vol. 55 (IEEE), 1–4

Athalathil, J. J., Vaidya, B., Kundu, S., Upendran, V., & Cheung, M. C. 2024, The Astrophysical Journal, 975, 258

Attie, R. 2025, In preparation

Babcock, H. W. 1961, ApJ, 133, 572

Baeza, A., Bürger, R., Mulet, P., & Zorío, D. 2020, SIAM Journal on Scientific Computing, 42, A1028–A1051

Barnes, G., DeRosa, M. L., Jones, S. I., et al. 2023, ApJ, 946, 105

Bell, N., & Garland, M. 2008, Efficient sparse matrix-vector multiplication on CUDA, Tech. rep., Nvidia Technical Report NVR-2008-004, Nvidia Corporation

Briggs, W. L., McCormick, S. F., et al. 2000, A multigrid tutorial (Siam)

Caplan, R. M., Downs, C., & Linker, J. 2019, in AGU Fall Meeting Abstracts, Vol. 2019, SH43E

Caplan, R. M., Johnston, C. D., Daldoff, L. K. S., & Linker, J. A. 2024a, Journal of Physics: Conference Series, 2742, 012020

Caplan, R. M., Mikić, Z., Linker, J. A., & Lionello, R. 2017, Journal of Physics: Conference Series, 837, 012016

Caplan, R. M., Stulajter, M. M., Linker, J. A., et al. 2024b, in Proceedings of the SC '24 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis, SC-W '24 (New York, NY, USA: Association for Computing Machinery), 1904

Chapman, G. A., & Boyden, J. E. 1986, ApJL, 302, L71

Chen, R., Zhao, J., Webber, S. H., et al. 2022, The Astrophysical Journal, 941, 197

Cravero, I., & Semplice, M. 2015, Journal of Scientific Computing, 67, 1219–1246

Curcic, M. 2020, Modern Fortran: Building efficient parallel applications (Manning Publications)

Dash, S., DeRosa, M. L., Dikpati, M., et al. 2024, The Astrophysical Journal, 975, 288

Dawes, A. 2021, Computers & Fluids, 214, 104762

Deakin, T., & Mattson, T. G. 2023, Programming Your GPU with OpenMP: Performance Portability for GPUs (MIT Press)

DeVore, C. R., Boris, J. P., & Sheeley, Jr., N. R. 1984, SoPh, 92, 1

Downs, C., Linker, J. A., Caplan, R. M., et al. 2024, Science, Under revision

Driscoll, T. A., Hale, N., & Trefethen, L. N. 2014, Chebfun Guide (Pafnuty Publications)

Feng, X. 2019, Magnetohydrodynamic modeling of the solar corona and heliosphere (Springer)

Feng, X., Lv, J., Xiang, C., & Jiang, C. 2023, Monthly Notices of the Royal Astronomical Society, 519, 6297

Gombosi, T. I., van der Holst, B., Manchester, W. B., & Sokolov, I. V. 2018, Living Reviews in Solar Physics, 15

Gottlieb, S., Ketcheson, D., & Shu, C.-W. 2011, Strong Stability Preserving Runge-Kutta and Multistep Time Discretizations (WORLD SCIENTIFIC), 84

Group, T. H. 2024, Hierarchical Data Format, version 5

Hanna, S. R., & Heinold, D. W. 1986, Simple Statistical Methods for Comparative Evaluation of Air Quality Models (Boston, MA: Springer US), 441

Harvey, J., Hill, F., Hubbard, R., et al. 1996, Science, 272, 1284

Hathaway, D. H., Teil, T., Norton, A. A., & Kitiashvili, I. 2015, The Astrophysical Journal, 811, 105

Hathaway, D. H., Upton, L. A., & Mahajan, S. S. 2022, Frontiers in Astronomy and Space Sciences, 9, 1007290

Hathaway, D. H., Williams, P. E., Rosa, K. D., & Cuntz, M. 2010, The Astrophysical Journal, 725, 1082

Henney, C. J., Hock, R. A., Schooley, A. K., et al. 2015, Space Weather, 13, 141

Hess Webber, S., Chen, R., DeRosa, M. L., Upton, L., & Zhao, J. 2020, in AGU Fall Meeting Abstracts, Vol. 2020, SH002

Hickmann, K. S., Godinez, H. C., Henney, C. J., & Arge, C. N. 2015, Solar Physics, 290, 1105

Holmes, S. A., & Featherstone, W. E. 2002, Journal of Geodesy, 76, 279

Howard, R., & Harvey, J. 1970, Solar Physics, 12, 23

ISO. 2023, Programming languages — Fortran Part 1: Base language, Standard, International Organization for Standardization, Geneva, CH

Jiang, J., Hathaway, D. H., Cameron, R. H., et al. 2014, Space Science Reviews, 186, 491–523

Keller, C. U., Harvey, J. W., & Giampapa, M. S. 2003a, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 4853, Innovative Telescopes and Instrumentation for Solar Astrophysics, ed. S. L. Keil & S. V. Avakyan, 194

Keller, C. U., Harvey, J. W., & Solis Team. 2003b, in Astronomical Society of the Pacific Conference Series, Vol. 307, Solar Polarization, ed. J. Trujillo-Bueno & J. Sanchez Almeida, 13

Knizhnik, K. J., Weberg, M. J., Provornikova, E., et al. 2024, The Astrophysical Journal, 964, 188

Leighton, R. B. 1964, ApJ, 140, 1547

Li, S. 2006, in CSC, Citeseer, 177

Liewer, P., González Hernández, I., Hall, J., Lindsey, C., & Lin, X. 2014, Solar Physics, 289, 3617

Linker, J. A., Caplan, R. M., Downs, C., et al. 2017, ApJ, 848, 70

Lionello, R., Downs, C., Mason, E. I., et al. 2023, The Astrophysical Journal, 959, 77

Liu, X.-D., Osher, S., & Chan, T. 1994, Journal of computational physics, 115, 200

Lunet, T., Lac, C., Auguste, F., et al. 2017, Monthly Weather Review, 145, 3817–3838

Mackay, D., & Van Ballegooijen, A. 2006, The Astrophysical Journal, 641, 577

Mackay, D., & Yeates, A. 2012, Living Reviews in Solar Physics, 9

MacNamara, S., & Strang, G. 2016, Operator Splitting (Cham: Springer International Publishing), 95

Mason, E. I., Lionello, R., Downs, C., et al. 2023, The Astrophysical Journal Letters, 959, L4

Mentaschi, L., Besio, G., Cassola, F., & Mazzino, A. 2013, Ocean Modelling, 72, 53

Meyer, C. D., Balsara, D. S., & Aslam, T. D. 2014, Journal of Computational Physics, 257, 594

Mignone, A. 2014, Journal of Computational Physics, 270, 784

Mikić, Z., & Linker, J. A. 1996, in International Solar Wind 8, ed. e. a. Winterhalter, D., Vol. 382, AIP Conf. Proceedings, 104

Riley, P., Lionello, R., Linker, J. A., et al. 2011, Solar Physics, 274, 361–377

Rincon, F., & Rieutord, M. 2018, Living Reviews in Solar Physics, 15

Saad, Y. 2003, Iterative methods for sparse linear systems (Siam)

Scherrer, P. H., Schou, J., Bush, R., et al. 2012, Solar Physics, 275, 207

Schrijver, C. J., & DeRosa, M. L. 2003, Solar Physics, 212, 165

Schrijver, C. J., Title, A. M., van Ballegooijen, A. A., Hagenaar, H. J., & Shine, R. A. 1997, ApJ, 487, 424

Shadab, M. A., Balsara, D., Shyy, W., & Xu, K. 2019, Computers & Fluids, 190, 398

Sheeley, Jr., N. R. 2005, Living Reviews in Solar Physics, 2, 5

Shu, C.-W. 2009, SIAM review, 51, 82

Simpson, M. J., & Landman, K. A. 2008, Mathematics and Computers in Simulation, 77, 9

Skaras, T., Saxton, T., Meyer, C., & Aslam, T. D. 2021, Journal of Computational Physics, 425, 109879

Smit, J., van Sint Annaland, M., & Kuipers, J. 2005, Chemical engineering science, 60, 2609

Snodgrass, H. B. 1983, The Astrophysical Journal, 270, 288

Snodgrass, H. B., & Ulrich, R. K. 1990, The Astrophysical Journal, 351, 309

Solanki, S. K., del Toro Iniesta, J. C., Woch, J., et al. 2020, A&A, 642, A11

Spiteri, R. J., & Ruuth, S. J. 2002, SIAM Journal on Numerical Analysis, 40, 469–491

Stenflo, J. O. 1974, Solar Physics, 36, 495–515

Stenflo, J. O. 1977, A&A, 61, 797

Strikwerda, J. C. 2004, Finite Difference Schemes and Partial Differential Equations, Second Edition (Society for Industrial and Applied Mathematics), https://epubs.siam.org/doi/pdf/10.1137/1.9780898717938

Upton, L. 2025, In preparation

Upton, L., & Hathaway, D. H. 2013, The Astrophysical Journal, 780, 5

—. 2014, The Astrophysical Journal, 792, 142

Usmanov, A. V. 1993, SoPh, 146, 377

Verwer, J. G. 1996, Applied Numerical Mathematics, 22, 359

Wang, Y.-M., Nash, A., & Sheeley Jr, N. 1989, The Astrophysical Journal, 347, 529

Wang, Y. M., Nash, A. G., & Sheeley, Jr., N. R. 1989, Science, 245, 712

Wang, Y.-M., & Sheeley, Jr., N. R. 1991, ApJ, 375, 761

Wang, Y.-M., & Sheeley Jr, N. 1992, Astrophysical Journal, Part 1 (ISSN 0004-637X), vol. 392, no. 1, June 10, 1992, p. 310-319. Research supported by US Navy., 392, 310

Warren, H. P., Floyd, L. E., & Upton, L. A. 2021, Space Weather, 19, e02860

Wiegelmann, T., & Sakurai, T. 2021, Living Reviews in Solar Physics, 18

Worden, J., & Harvey, J. 2000, Sol. Phys., 195, 247

Yang, D., Heinemann, S., Cameron, R., & Gizon, L. 2024, Solar Physics, 299, 161

Yeates, A. R., Cheung, M. C. M., Jiang, J., Petrovay, K., & Wang, Y.-M. 2023, Space Science Reviews, 219