Graph-based Neural Space Weather Forecasting

Daniel Holmberg[1,2*] **Ivan Zaitsev**[1] **Markku Alho**[1] **Ioanna Bouri**[1] **Fanni Franssila**[1]
Haewon Jeong[2] **Minna Palmroth**[1,3] **Teemu Roos**[1]

[1]University of Helsinki [2]UC Santa Barbara [3]FMI Space and Earth Observation Centre

Abstract

Accurate space weather forecasting is crucial for protecting our increasingly digital infrastructure. Hybrid-Vlasov models, like Vlasiator, offer physical realism beyond that of current operational systems, but are too computationally expensive for real-time use. We introduce a graph-based neural emulator trained on Vlasiator data to autoregressively predict near-Earth space conditions driven by an upstream solar wind. We show how to achieve both fast deterministic forecasts and, by using a generative model, produce ensembles to capture forecast uncertainty. This work demonstrates that machine learning offers a way to add uncertainty quantification capability to existing space weather prediction systems, and make hybrid-Vlasov simulation tractable for operational use.

1 Introduction

Space weather describes conditions in near-Earth space driven by the solar wind and the internal dynamics of Earth's magnetosphere. These conditions threaten modern infrastructure by creating geomagnetically induced currents that disrupt power networks [1, 2], adversely affecting satellite operations [3, 4], and causing failures in electromagnetic communication [5]. Current operational forecasting relies on driving global magnetohydrodynamic (MHD) models like BATS-R-US [6] with real-time solar wind data from satellites orbiting the L1 Lagrange point, such as ACE [7]. However, this paradigm faces several challenges. First, MHD models approximate plasma as a fluid, omitting ion-kinetic processes that are only captured by higher-fidelity but computationally prohibitive simulations like hybrid-Vlasov models. Second, most operational forecasts are deterministic single-point predictions, which lack crucial uncertainty information. To address this, the need for ensemble forecasting has been highlighted [8], with existing approaches perturbing the solar wind inputs for physics-based models [9], or using machine learning for post-processing deterministic outputs [10].

To tackle these challenges, we take inspiration from recent breakthroughs in machine learning for atmospheric weather forecasting [11, 12, 13, 14, 15, 16], and adapt graph-based limited-area modeling [14] to the domain of space weather. We tailor established graph neural network (GNN) architectures [11, 13, 14] to a magnetospheric simulation grid in the noon-midnight meridional plane, excluding the circular inner boundary close to Earth, and incorporate the upstream solar wind as boundary condition. By doing so, we can produce high-fidelity forecasts of the magnetosphere's evolution more than 100 times faster on 1 GPU than the original simulation on 50 CPUs. We show that the framework is capable of both deterministic and probabilistic forecasting, with the latter using a latent-variable approach to generate ensembles that provide uncertainty information currently missing from most space weather models.

[*]Corresponding author: `dholmberg@ucsb.edu`

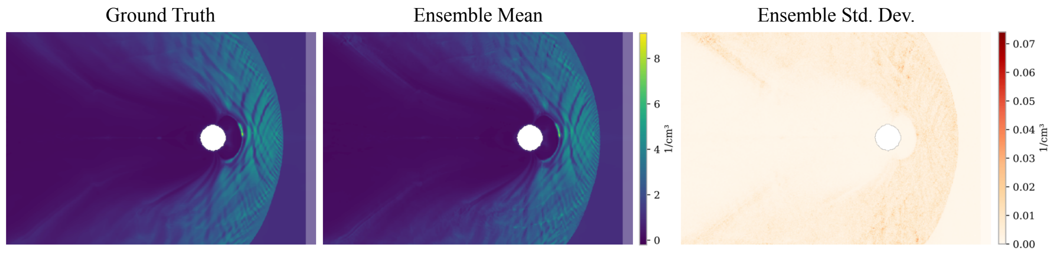


Figure 1: Example forecast of particle density 10 steps ahead, showing the faded inflow boundary.

2 Hybrid-Vlasov dataset

The data for this study were generated using Vlasiator [17], which performs global simulations of the solar wind's interaction with the Earth's magnetosphere in the hybrid-Vlasov formalism. Ions are treated as a velocity distribution function, f, that depends on position $\mathbf{x}$, velocity $\mathbf{v}$, and time t. Its evolution is dictated by the Vlasov equation, which describes how f changes due to the electric field $\mathbf{E}$ and magnetic field $\mathbf{B}$:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{x}} + (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f}{\partial \mathbf{v}} = 0. \tag{1}$$

Electrons are modeled as a massless, charge-neutralizing fluid where the number density n is equal for both ions and electrons ($n_i \simeq n_e \simeq n$). The electromagnetic fields are evaluated by solving the Maxwell-Darwin system: $\nabla \times \mathbf{E} = -\partial_t \mathbf{B},\ \nabla \times \mathbf{B} = \mu_0 \mathbf{J},\ \nabla \cdot \mathbf{B} = 0$. This system is closed by relating the fields to the moments of the ion distribution function through a generalized Ohm's law that includes the Hall term, which involves the current density $\mathbf{J}$ and the elementary charge e:

$$\mathbf{E} + \mathbf{v} \times \mathbf{B} = \frac{\mathbf{J} \times \mathbf{B}}{ne}. \tag{2}$$

The run was performed in a 2D-3V configuration (two spatial and three velocity dimensions) using a timestep $\Delta t = 1\,\mathrm{s}$ and spatial resolution of $600\,\mathrm{km}$. The simulation domain covers the noon-midnight meridional plane (the x-z plane in Geocentric Solar Ecliptic, or GSE, coordinates), extending from $-60\,R_E$ to $+30\,R_E$ along the x-axis and $\pm 30\,R_E$ along the z-axis. The solar wind is injected at the $+x$ boundary as a Maxwellian distribution, parametrized by an ion density of $\rho = 1/\mathrm{cm}^3$, wind velocity of $\mathbf{v} = (-750, 0, 0)\,\mathrm{km/s}$, and a plasma temperature of $T = 0.5\,\mathrm{MK}$. The interplanetary magnetic field is directed southward with $\mathbf{B} = (0, 0, -5)\,\mathrm{nT}$. These conditions result in an Alfvén Mach number of $M_A = 6.9$. Plasma can exit through the other edges of the domain, where Neumann boundary conditions are applied. The inner boundary is represented as a perfect conductor located at a radius of 3.7 R_E from the Earth's center. All vector quantities are given in GSE coordinates.

3 Method

Problem formulation We formulate the problem of space weather forecasting as mapping a set of initial magnetospheric states to a sequence of future states. The input consists of two consecutive states, $X^{-1:0} = (X^{-1}, X^0)$, to capture first-order dynamics [13], with the goal of predicting the future trajectory $X^{1:T} = (X^1, \ldots, X^T)$. Each magnetospheric state $X^t \in \mathbb{R}^{N \times d_x}$ is a high-dimensional tensor representing d_x physical variables present in the hybrid-Vlasov simulation across N grid locations in near-Earth space. The complete feature set is listed in Appendix B. Deterministic models tackle the problem by producing a single point, typically mean, estimate of the trajectory $X^{1:T}$, while probabilistic approaches aim to model the full conditional distribution $p(X^{1:T}|X^{-1:0})$.

Graph-based neural forecasting Using machine learning, the deterministic forecasting problem can be solved with an autoregressive model applied iteratively to produce a forecast. In the probabilistic case, we sample from the model's output distribution and repeat the process to generate an ensemble of trajectories. For both cases, we use a GNN based on an *encode-process-decode* architecture [18] where: 1) grid inputs are encoded onto the mesh representation; 2) a number of GNN layers process this latent representation; 3) the processed data is mapped back onto the original grid to produce the final prediction. Here the GNN predicts the next step as a residual update to the most recent input state, making it an easier learning task compared to predicting the next state directly. For the model to better handle the open boundary on Earth's dayside, we apply boundary forcing as an additional input in the region from $x = 27\,R_E$ to the domain edge at $x = 30\,R_E$, shown as the faded areas in Fig. 1. A static binary mask indicates which grid nodes to force by replacing with ground truth boundary data after each prediction. In an operational scenario this information would come from conditioning on L1 observations [6, 19], or from a heliospheric host model [20].

Mesh variations We consider three mesh architectures for the GNN processor: a *simple* mesh coarser than the simulation domain [11]; a *multiscale* mesh [13]; and a *hierarchical* mesh [14]. The multiscale and hierarchical meshes are constructed by recursively creating a sequence of graphs, $\mathcal{G}_L, \dots, \mathcal{G}_1$, at varying resolutions. The multiscale mesh, $\mathcal{G}_{MS}$, uses the nodes from the finest graph, $\mathcal{V}_1$, but connects them with edges from all levels, $\mathcal{E}_L \cup \cdots \cup \mathcal{E}_1$, allowing a single GNN layer to process information across all spatial scales simultaneously. In contrast, the hierarchical approach maintains L distinct graph levels, $\mathcal{G}_1, \dots, \mathcal{G}_L$, where the number of nodes decreases at each level. Information is passed between adjacent levels using dedicated inter-level graphs, $\mathcal{G}_{l,l+1}$ up and $\mathcal{G}_{l+1,l}$ down, to model different spatial scales separately. The mappings between grid and mesh nodes occur through bipartite grid-to-mesh $\mathcal{G}_{G2M}$ and mesh-to-grid $\mathcal{G}_{M2G}$ graphs.

Deterministic model We use the deterministic graph-based forecasting model *Graph-FM* [14] to generate point estimate forecasts through the autoregressive mapping $\hat{X}^t = f(X^{t-2:t-1})$. We test this model with each of the three mesh architectures, adapting its processor accordingly. When using the hierarchical mesh, a processing step is a complete sweep through the hierarchy. A sweep sequentially applies GNNs to propagate information up from the lowest level $\mathcal{G}_1$ to the highest $\mathcal{G}_L$ and then back down. All upward updates are facilitated by *propagation networks* [14], while the remaining updates use *interaction networks* [21], with all layers mapping to a latent dimensionality d_z. This design leverages the inductive bias of interaction networks to retain information, while propagation networks are more effective at forwarding new information through the graph. We train the deterministic models by minimizing a weighted mean square error (MSE) loss.

Probabilistic model For probabilistic space weather forecasting we employ the graph-based ensemble forecasting model, *Graph-EFM* [14]. To model the distribution $p(X^t|X^{t-2:t-1})$ for a single time step, Graph-EFM uses a latent random variable Z^t. This variable acts as a low-dimensional representation of the stochastic elements of the weather system that are not captured by the input states. By conditioning the prediction on this latent variable, the model can efficiently sample different, spatially coherent outcomes. The relationship is defined by the integral:

$$p(X^t|X^{t-2:t-1}) = \int p(X^t|Z^t, X^{t-2:t-1})p(Z^t|X^{t-2:t-1})dZ^t \tag{3}$$

where the term $p(Z^t|X^{t-2:t-1})$ is the latent map and the term $p(X^t|Z^t, X^{t-2:t-1})$ is the predictor. The latent map is a probabilistic mapping that takes the previous two weather states as input and produces the parameters for a distribution over the latent variable Z^t. Specifically, Graph-EFM uses GNNs to map the inputs to the mean of an isotropic Gaussian distribution, effectively encoding the state-dependent uncertainty into the latent space. The variance is kept fixed. This latent distribution is defined over the nodes $\mathcal{V}_L$ in the top, coarsest level of a given mesh graph, as follows:

$$p(Z^t|X^{t-2:t-1}) = \prod_{v \in \mathcal{V}_L} \mathcal{N}(Z^t_v|\mu_Z(X^{t-2:t-1})_v, I) \tag{4}$$

ensuring that the stochasticity is introduced at a low-dimensional, spatially-aware level. The predictor is a deterministic mapping that takes a specific sample of the latent variable Z^t, along with the previous weather states, to produce the next weather state $\hat{X}^t$. A sampled value of Z^t is injected at the top level of the graph hierarchy and its influence propagates down through the levels to produce a full, high-resolution, and spatially coherent forecast. The predictor is a deterministic mapping, g, that produces the next state $\hat{X}^t$ by adding a predicted residual update, $\tilde{g}$, to the previous state X^{t-1}:

$$\hat{X}^t = g(Z^t, X^{t-2:t-1}) = X^{t-1} + \tilde{g}(Z^t, X^{t-2:t-1}). \tag{5}$$

By first sampling a Z^t from the latent map and then passing it through the predictor, the model generates one possible future weather state. Repeating this process creates an arbitrarily large ensemble of forecasts. Drawing from the structure of a conditional Variational Autoencoder (VAE) [22, 23], we train the Graph-EFM by optimizing a variational objective equivalent to the negative Evidence Lower Bound (ELBO), which combines a Kullback-Leibler (KL) divergence regularizer with a reconstruction loss. Subsequently, we fine-tune the model by adding a Continuous Ranked Probability Score (CRPS) loss [24, 25, 26] to the objective, for calibration. Further details on the models and the loss functions are available in Appendix C.

4 Experiments

To evaluate our models, the Vlasiator simulation is causally split into training, validation, and test sets with durations of 10 minutes, 1 minute, and 1 minute, respectively. This ensures that the evaluation tests for meaningful generalization into the future. We train both the deterministic Graph-FM and the probabilistic Graph-EFM using the simple, multiscale, and hierarchical mesh architectures. All models are configured with a latent dimension of $d_z = 64$. For the simple and multiscale graphs, the processor consists of 4 processing layers. To ensure a fair comparison, the hierarchical Graph-FM model uses 2 processing layers, as its sweep mechanism effectively doubles the number of updates per layer. Graph-EFM is set to generate an ensemble size of 5. On a single GPU, the deterministic models predict the next step approximately 500 times faster than the original simulation running on 50 CPUs, while our probabilistic models are roughly 80 times faster. Appendix D contains more details on the training setup and computational complexity.

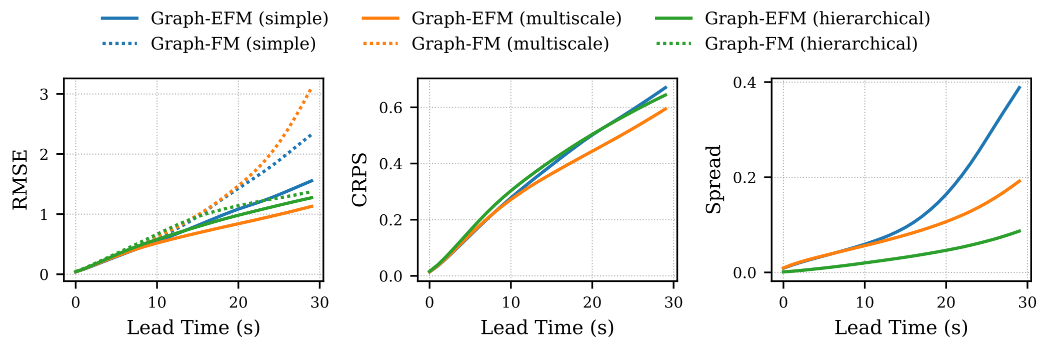


Figure 2: The mean of the normalized forecast RMSE, CRPS, and Spread over all variables.

Results Figure 1 displays a sample forecast from the hierarchical Graph-EFM, showing the ensemble mean and the ensemble standard deviation as a measure of forecast uncertainty. The performance of all models is summarized in Figure 2, which plots the normalized RMSE, CRPS, and Spread averaged over all variables. To normalize the metrics as unitless scores, we divide each by the standard deviation of its corresponding variable. The RMSE measures the accuracy of the deterministic forecast (or the ensemble mean), while the CRPS provides a comprehensive assessment of the probabilistic forecast's accuracy and calibration. In the determistic setting the hierarchical graph architecture accumulates less error than the simple and multiscale variations. The probabilistic models are quite comparable in performance. However, they tend to be underdispersed. In a well-calibrated forecasting system the ensemble spread should match the error [27], here the spread is smaller than the error. The limited sample size in our training data is most likely the main culprit here, but this is also a known characteristic of models trained primarily with a variational objective and in limited-area weather modeling [14], analogous to our magnetospheric setup in space. We also performed fairly conservative CRPS fine-tuning here, which is designed to alleviate underdispersion. Additional details on the metrics and per-variable results can be found in Appendices E and F.

5 Conclusion and outlook

In this work, we demonstrate that graph-based neural emulators can learn the complex dynamics of near-Earth space from a hybrid-Vlasov simulation, and provide exciting new uncertainty quantification capability for space weather prediction. Future development should focus on extending the model to three spatial dimensions, and depending on the simulation, a refined grid, for which GNNs are well suited. Training on a larger sample size coupled to varying real world input conditions covering multiple solar cycles would be ideal. This can also enable the use of larger timesteps, circumventing much of the cumulative error inherent to autoregressive emulators. Finally, looking into the physical realism of the forecasts and incorporating constraints, such as the divergence freeness of the magnetic field, is another promising future direction to follow.

Acknowledgements

This work was funded by the Research Council of Finland under the FAISER project (grant nos. 361901, 361902). D.H. acknowledges support from the Fulbright-KAUTE Foundation Award for conducting research at UCSB. Computing resources were provided by the LUMI supercomputer, owned by the EuroHPC Joint Undertaking and hosted by CSC–IT Center for Science.

References

[1] Léonard Bolduc. GIC observations and studies in the Hydro-Québec power system. *Journal of Atmospheric and Solar-Terrestrial Physics*, 64(16):1793–1802, 2002.

[2] Andrew P Dimmock, L Rosenqvist, J-O Hall, A Viljanen, Emiliya Yordanova, I Honkonen, Mats André, and EC Sjöberg. The GIC and geomagnetic response over Fennoscandia to the 7–8 September 2017 geomagnetic storm. *Space Weather*, 17(7):989–1010, 2019.

[3] Robin Gubby and John Evans. Space environment effects and satellite design. *Journal of Atmospheric and Solar-Terrestrial Physics*, 64(16):1723–1733, 2002.

[4] Yongliang Zhang, Larry J Paxton, Robert Schaefer, and William H Swartz. Thermospheric conditions associated with the loss of 40 Starlink satellites. *Space Weather*, 20(10):e2022SW003168, 2022.

[5] DN Baker, Eamonn Daly, Ioannis Daglis, John G Kappenman, and Mikhail Panasyuk. Effects of space weather on technology infrastructure. *Space Weather*, 2(2), 2004.

[6] A Glocer, M Fok, X Meng, G Toth, N Buzulukova, S Chen, and K Lin. CRCM + BATS-R-US two-way coupling. *Journal of Geophysical Research: Space Physics*, 118(4):1635–1650, 2013.

[7] Edward C Stone, AM Frandsen, RA Mewaldt, ER Christian, D Margolies, JF Ormes, and F Snow. The advanced composition explorer. *Space Science Reviews*, 86(1):1–22, 1998.

[8] Sophie A Murray. The importance of ensemble techniques for operational space weather forecasting. *Space Weather*, 16(7):777–783, 2018.

[9] Steven Karl Morley, Daniel T Welling, and Jesse Richard Woodroffe. Perturbed input ensemble modeling with the space weather modeling framework. *Space Weather*, 16(9):1330–1347, 2018.

[10] Enrico Camporeale, Xiangning Chu, OV Agapitov, and Jacob Bortnik. On the generation of probabilistic forecasts from deterministic models. *Space Weather*, 17(3):455–475, 2019.

[11] Ryan Keisler. Forecasting global weather with graph neural networks. *arXiv preprint arXiv:2202.07575*, 2022.

[12] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3D neural networks. *Nature*, 619(7970):533–538, 2023.

[13] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

[14] Joel Oskarsson, Tomas Landelius, Marc Peter Deisenroth, and Fredrik Lindsten. Probabilistic weather forecasting with hierarchical graph neural networks. In *Advances in Neural Information Processing Systems*, 2024.

[15] Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, et al. Probabilistic weather forecasting with machine learning. *Nature*, 637(8044):84–90, 2025.

[16] Erik Larsson, Joel Oskarsson, Tomas Landelius, and Fredrik Lindsten. Diffusion-LAM: Probabilistic limited area weather forecasting with diffusion. In *ICLR 2025 Workshop on Tackling Climate Change with Machine Learning*, 2025.

[17] S Von Alfthan, D Pokhotelov, Y Kempf, S Hoilijoki, I Honkonen, A Sandroos, and M Palmroth. Vlasiator: First global hybrid-Vlasov simulations of Earth's foreshock and magnetosheath. *Journal of Atmospheric and Solar-Terrestrial Physics*, 120:24–35, 2014.

[18] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, 2020.

[19] Ilja Honkonen, Max van de Kamp, Theresa Hoppe, and Kirsti Kauristie. Over 20-year global magnetohydrodynamic simulation of Earth's magnetosphere. *Space Weather*, 20(11):e2022SW003196, 2022.

[20] Anwesha Maharana, W Douglas Cramer, Evangelia Samara, Camilla Scolini, Joachim Raeder, and Stefaan Poedts. Employing the coupled EUHFORIA-OpenGGCM model to predict CME geoeffectiveness. *Space Weather*, 22(5):e2023SW003715, 2024.

[21] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, 2016.

[22] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Internationcal Conference on Learning Representations*, 2014.

[23] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, 2015.

[24] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

[25] Stephan Rasp and Sebastian Lerch. Neural networks for postprocessing ensemble weather forecasts. *Monthly Weather Review*, 146(11):3885–3900, 2018.

[26] Lorenzo Pacchiardi, Rilwan A Adewoyin, Peter Dueben, and Ritabrata Dutta. Probabilistic forecasting with generative networks via scoring rule minimization. *Journal of Machine Learning Research*, 25(45):1–64, 2024.

[27] Vincent Fortin, Mabrouk Abaza, Francois Anctil, and Raphael Turcotte. Why should ensemble spread match the RMSE of the ensemble mean? *Journal of Hydrometeorology*, 15(4):1708–1713, 2014.

A Future work

Emulating kinetic-scale physics This work emulates the spatial moments of the ion Velocity Distribution Functions (VDFs), which Vlasiator solves fully. A natural progression would be to forecast the entire VDFs over time, rather than just their moments. Here one could take inspiration from previous work on emulating gyrokinetic simulations [28], where a hierarchical vision transformer was used for next step prediction of the 5D distribution function for adiabatic electrons.

Addressing error accumulation Neural emulators struggle with error accumulation on long rollouts, causing simulated trajectories to diverge. This happens because the model increasingly operates "out of distribution" with each autoregressive step. To combat this, techniques like thermalization can significantly extend stable rollout times by controlling error growth [29]. Another promising method to tackle cumulative error involves training models to forecast a future state in a single step while ensuring the temporal consistency of the forecast for each ensemble member [30].

Mesh refinement Simulating with hybrid-Vlasov models in three dimensions is computationally so expensive that mesh refinement becomes essential. This allows for higher spatial resolution in critical regions, like the bow shock and the magnetotail reconnection site near Earth, while using lower resolution in less important areas such as inflow and outflow boundaries [31]. This results in irregular data structures, making GNNs a future-proof model for modeling such data. However, recently physics-motivated adaptive refinement has been tested [32], where the refinement itself is driven by physical conditions. This presents an additional interesting challenge for neural emulators.

Foundation models Recent community initiatives are gathering diverse physics simulations to create large-scale datasets for machine learning [33, 34]. Such datasets can be used to train foundation models, which can learn across scales or multiple physical systems [35, 36, 37, 38]. Space weather data would be a valuable addition to these growing datasets. Conversely, space weather prediction models could potentially be improved by training on diverse simulation data, allowing them to leverage insights from various physical domains and potentially generalize better.

B Dataset details

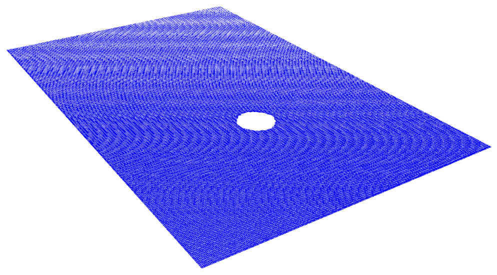
The data used for this study is simulated using Vlasiator and contains the variables listed in Table 1.

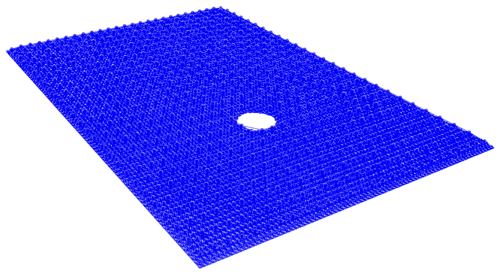
Table 1: Summary of all variables and static fields in the Vlasiator dataset.

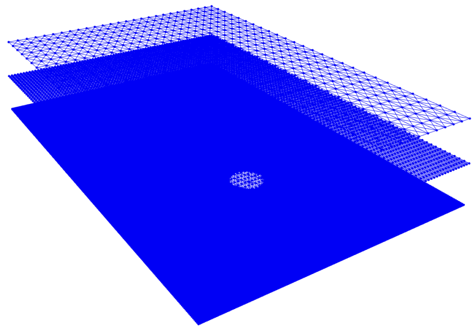
	Abbreviation	**Unit**
Variables		
Magnetic field x-component	B_x	nT
Magnetic field y-component	B_y	nT
Magnetic field z-component	B_z	nT
Electric field x-component	E_x	mV/m
Electric field y-component	E_y	mV/m
Electric field z-component	E_z	mV/m
Velocity field x-component	v_x	km/s
Velocity field y-component	v_y	km/s
Velocity field z-component	v_z	km/s
Particle number density	ρ	1/cm^3
Plasma temperature	T	MK
Plasma pressure	P	nPa
Static fields		
Coordinate in x	x	R_E
Coordinate in z	z	R_E
Radial distance from Earth	r	R_E

C Model details

The 671×1006 (z, x) data grid excludes 5124 inner boundary nodes. We construct graphs by recursively downsampling the grid, placing each coarser-level node at the center of a 3×3 finer-level node square as seen in Fig. 3. We compare three mesh architectures: a simple single-level graph, a three-level multiscale graph, and a three-level hierarchical graph, with full statistics in Table 2. The multiscale version connects all levels into a single heterogeneous mesh to capture both short- and long-range interactions, while the hierarchical version uses distinct, uniform levels to process information at different spatial scales separately. For each node, an MLP encodes static features in Table 1, concatenated with previous states. For a complete explanation on update rules in the GNNs see [14]. All MLPs use one hidden layer with Swish activation [39] and layer normalization [40].


(a) Simple Graph


(b) Multiscale Graph


(c) Graph Layers

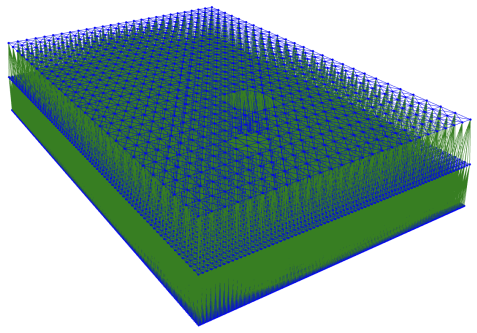

(d) Hierarchical Graph

Figure 3: Quadrilateral mesh variations used by the GNNs. Nodes and same-level edges in blue, and inter-level edges in green. Node size corresponds to degree in the multiscale graph.

Table 2: Number of nodes and edges in the different graphs.

Graph		**Nodes**	**Edges**
Simple	$\mathcal{G}_{\text{S}}$	58592	465584
Multiscale	$\mathcal{G}_{\text{MS}}$	58592	522054
Hierarchical	$\mathcal{G}_0$	58592	465584
	$\mathcal{G}_{0,1}/\mathcal{G}_{1,0}$	-	58592
	$\mathcal{G}_1$	6510	51032
	$\mathcal{G}_{1,2}/\mathcal{G}_{2,1}$	-	6510
	$\mathcal{G}_2$	723	5438
	Total	65825	587156
$\mathcal{G}_{\text{G2M}}$		-	1411687
$\mathcal{G}_{\text{M2G}}$		-	2679608
Grid		669902	-

Deterministic objective We train the deterministic models by minimizing a weighted MSE:

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{d_x} \omega_i \lambda_i \left(\hat{X}_{n,i} - X_{n,i} \right)^2 . \tag{6}$$

The loss is weighted by two terms: λ_i is the inverse variance of the time differences for variable j, which normalizes the contribution of variables with different dynamic ranges. The second term, ω_i, is a variable-specific weight. While often used in atmospheric forecasting to prioritize variables by altitude, we set it uniformly to $1/d_x$.

Probabilistic objective The probabilistic model's single-step prediction has a structure analogous to a conditional VAE, and it is trained by optimizing a variational objective derived from the ELBO:

$$\begin{aligned} \mathcal{L}_{\text{Var}} \left(X^{t-2:t-1}, X^t, F^t \right) = \lambda_{\text{KL}} D_{\text{KL}} q \left(Z^t \middle| X^{t-2:t-1}, X^t \right) p \left(Z^t \middle| X^{t-2:t-1} \right) \\ - \mathbb{E} \left[q \left(Z^t \middle| X^{t-2:t-1}, X^t \right) \right] \textstyle\sum_{n=1}^{N} \sum_{i=1}^{d_x} \log \mathcal{N} X^t_{v,i} g \left(Z^t, X^{t-2:t-1} \right)_{v,i} \sigma^2_{v,i} . \end{aligned} \tag{7}$$

This objective consists of two terms. The first is the KL divergence, which acts as a regularizer, encouraging the approximate posterior distribution q to remain close to the prior p. The second term is the expected negative log-likelihood, or reconstruction loss, which trains the predictor g to accurately reconstruct the true state X^t from a latent sample Z^t. The hyperparameter λ_{KL} balances these two terms to prevent the model from collapsing to a deterministic prediction. As both distributions are Gaussian, the KL divergence has a closed-form solution. The variance $\sigma^2_{n,i}$ is an output of the predictor network. The model is further fine-tuned using the CRPS loss, which is minimized only when the predicted distribution matches the empirical data distribution. We use an unbiased two-sample estimator for the CRPS loss, summed over all grid points and variables:

$$\mathcal{L}_{\text{CRPS}} = \sum_{n=1}^{N} \sum_{i=1}^{d_x} \frac{1}{2} \left(\left| \hat{X}_{n,i} - X_{n,i} \right| + \left| \check{X}_{n,i} - X_{n,i} \right| - \left| \hat{X}_{n,i} - \check{X}_{n,i} \right| \right) \tag{8}$$

where $\hat{X}^t$ and $\check{X}^t$ are two independent ensemble members (forecasts) generated by the model. The final training loss combines both objectives: $\mathcal{L} = \mathcal{L}_{\text{Var}} + \lambda_{\text{CRPS}} \mathcal{L}_{\text{CRPS}}$.

D Training details

All models are trained using the AdamW optimizer [41] with parameters $\beta_1 = 0.9$, $\beta_2 = 0.95$, a weight decay of 0.01, and an effective batch size of 8. The specific training schedules for the deterministic and probabilistic models are detailed in Table 3. The deterministic Graph-FM model follows a standard training schedule with a learning rate decay after 300 epochs. The probabilistic Graph-EFM model is trained in three distinct stages. First, we train the model as a simple autoencoder with both λ_{KL} and λ_{CRPS} set to 0 to teach the encoder and decoder to reconstruct the fields from the latent space without probabilistic constraints. In the second stage, we introduce the KL divergence term to train the full variational model on the single-step distribution. Finally, we fine-tune the model with the CRPS loss to improve the calibration of the ensemble.

Table 3: Training schedules for the deterministic and probabilistic models.

Model	**Epochs**	**Learning Rate**	λ_{KL}	λ_{CRPS}
Graph-FM	300	10^{-3}	-	-
	200	10^{-4}	-	-
Graph-EFM	200	10^{-3}	0	0
	300	10^{-3}	1	0
	50	10^{-4}	1	10^{-3}

Computational complexity Table 4 shows the times it took to train the different models, and their inference speeds on the test set. Training was done on 8 AMD MI250X GPUs with 2 workers for the dataloader. The actual training time is therefore the GPU hours in the table divided by 8. Inference was performed on 1 AMD MI250X GPU and is measured as the time it takes for one next-step prediction. Graph-EFM produces an ensemble size of 5 here. For comparison, the Vlasiator simulation takes 4–5 minutes on 50 AMD EPYC 7H12 CPUs to simulate 1 s of real time.

Table 4: Training and inference times for different models and graph variations.

Model	Graph	Training time (GPU h)	Inference time (s)
Graph-FM	Simple	101	0.47
	Multiscale	102	0.48
	Hierarchical	108	0.52
Graph-EFM	Simple	119	3.20
	Multiscale	122	3.31
	Hierarchical	131	3.45

E Metrics

To evaluate the performance of our forecasts, we use a set of standard metrics. For an ensemble forecast with M members (deterministic forecasts have $M = 1$), we denote the prediction for variable i at location n for a given sample s at time t as $\hat{X}_{n,i}^{s,t,m}$, with the corresponding ground truth being $X_{n,i}^{s,t}$. RMSE is calculated by averaging the squared error over all S forecasts in the test set and all N grid locations:

$$\mathrm{RMSE}_{t,i} = \sqrt{\frac{1}{SN}\sum_{s=1}^{S}\sum_{n=1}^{N}\left(\overline{X}_{n,i}^{s,t} - X_{n,i}^{s,t}\right)^2} \tag{9}$$

$$\text{where} \quad \overline{X}_{n,i}^{s,t} = \frac{1}{M}\sum_{m=1}^{M}\hat{X}_{n,i}^{s,t,m}. \tag{10}$$

The CRPS is calculated for ensemble forecasts to assess the overall quality of a probabilistic forecast by comparing the entire predictive distribution to the single ground truth observation. Lower values indicate better performance. We use a finite-sample estimate [42], computed as:

$$\begin{aligned}\mathrm{CRPS}_{t,i} = \frac{1}{SN}\sum_{s=1}^{S}\sum_{n=1}^{N}\Bigg(&\frac{1}{M}\sum_{m=1}^{M}\left|\hat{X}_{n,i}^{s,t,m} - X_{n,i}^{s,t}\right| \\ &- \frac{1}{2M(M-1)}\sum_{m=1}^{M}\sum_{m'=1}^{M}\left|\hat{X}_{n,i}^{s,t,m} - \hat{X}_{n,i}^{s,t,m'}\right|\Bigg).\end{aligned} \tag{11}$$

The spread quantifies the uncertainty expressed by the ensemble. It is defined as the root mean square deviation of the ensemble members from the ensemble mean.

$$\mathrm{Spread}_{t,i} = \sqrt{\frac{1}{SMN}\sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{n=1}^{N}\left(\overline{X}_{n,i}^{s,t} - \hat{X}_{n,i}^{s,t,m}\right)^2} \tag{12}$$

F Additional results

Figure 4 shows that for deterministic models, the error in the simple and multiscale architectures grows more rapidly than in the hierarchical version. The ensemble models perform similarly to each other in terms of their mean error. The CRPS scores in Fig. 5 are also quite comparable. The ensemble spread shown in Fig. 6 is consistently lower than the error for all models, indicating that the forecasts are underdispersed. When these two values are equal, the forecasting system is considered well-calibrated.

Visual inspection of an example test set forecast 10 steps ahead produced by the hierarchical Graph-EFM model in Fig. 7–18 reveals that it successfully captures complex and detailed magnetospheric dynamics. The ensemble standard deviation correctly localizes the highest uncertainty to physically active regions such as the bow shock, magnetotail, and the tail lobes. However, we also observe the emergence of spurious structures not present in the ground truth, for example in the predicted B_y

and v_y components within the northern tail lobe. We attribute these artifacts primarily to the limited sample size of the training dataset.

To address these points, future work should focus on training with a larger dataset, which would make performance differences between models clearer and reduce artifacts. The ensemble calibration could be improved by increasing the CRPS weight during fine-tuning to enhance the spread. Finally, incorporating a rollout-based loss, i.e. optimizing over many future timesteps, can be a useful strategy to improve long-term stability of the emulators [13].

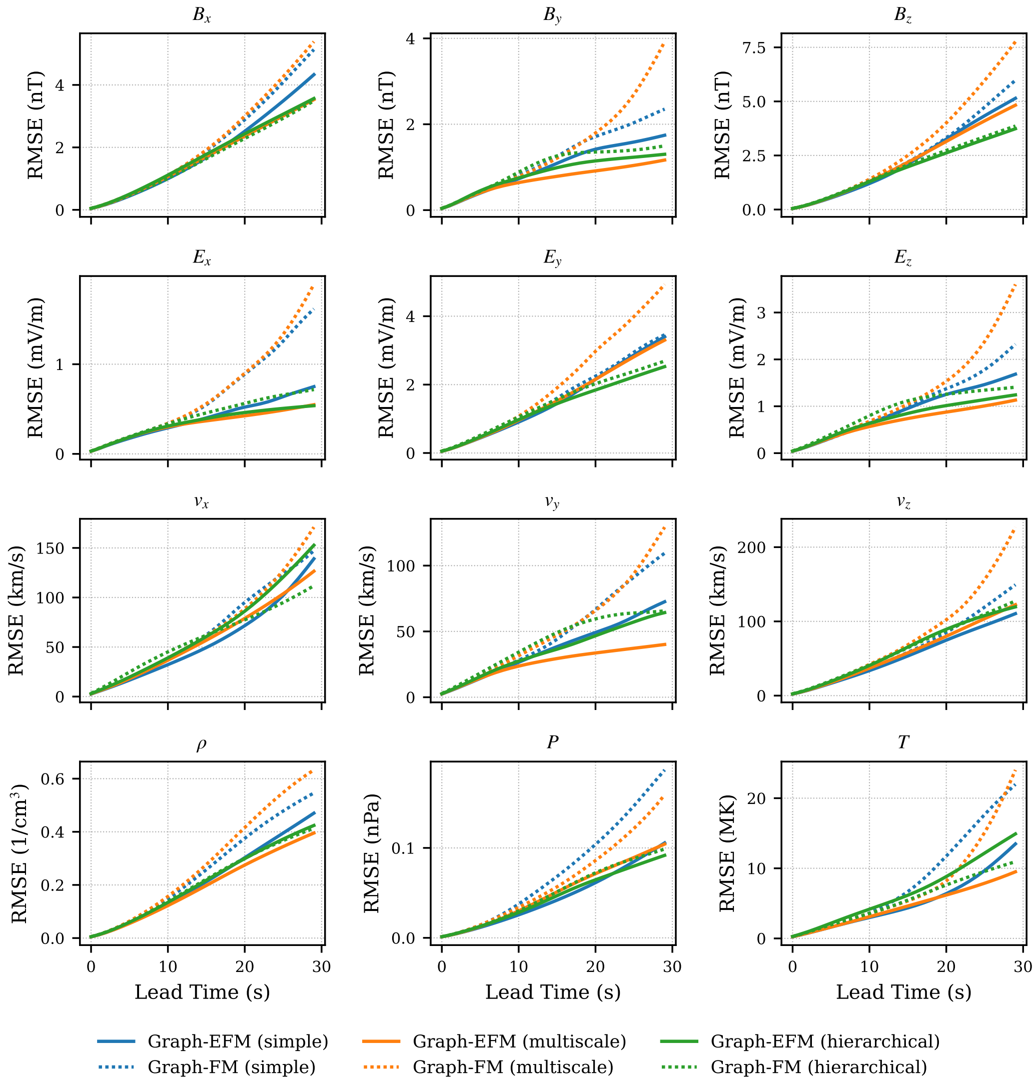


Figure 4: RMSE for all predicted variables as a function of forecast lead time.

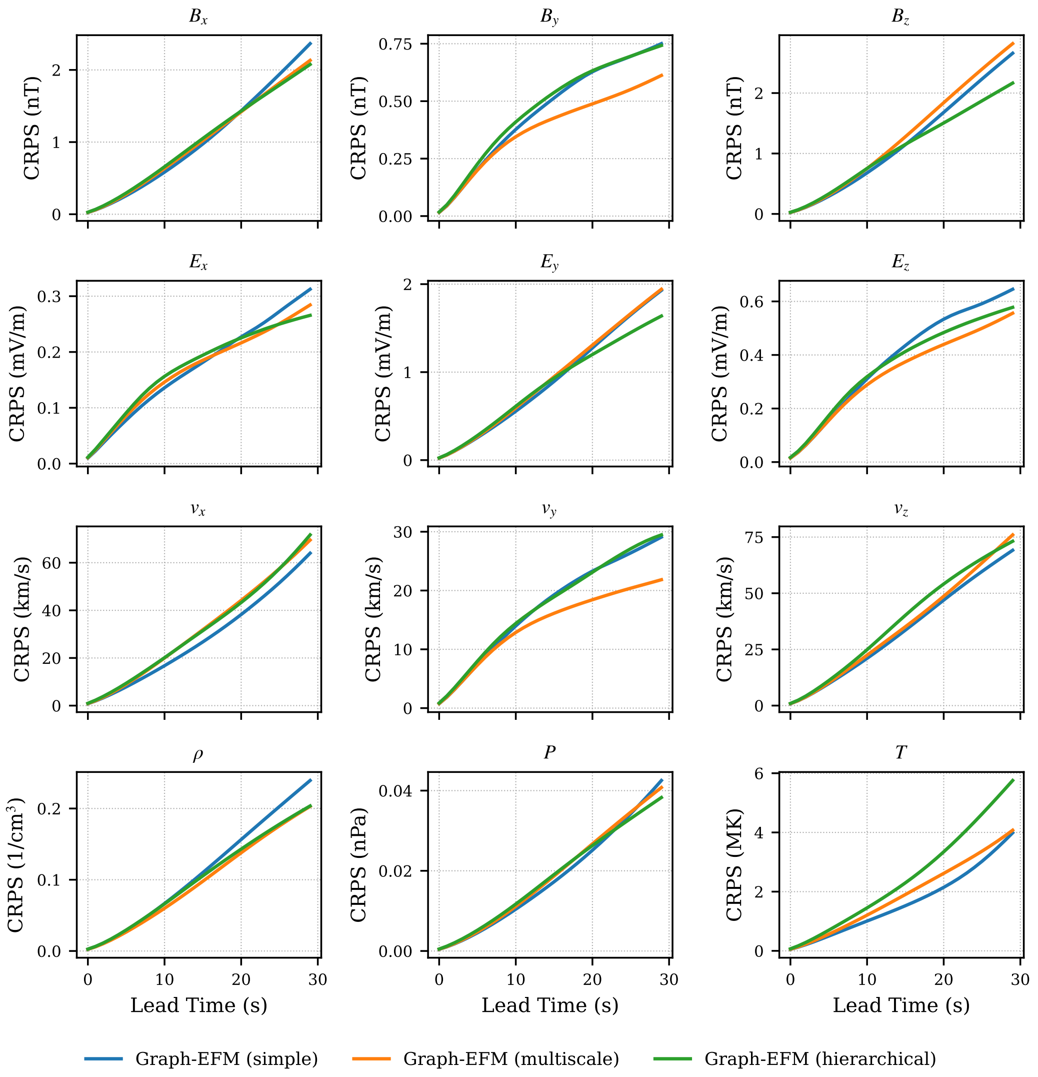


Figure 5: CRPS for all predicted variables as a function of forecast lead time.

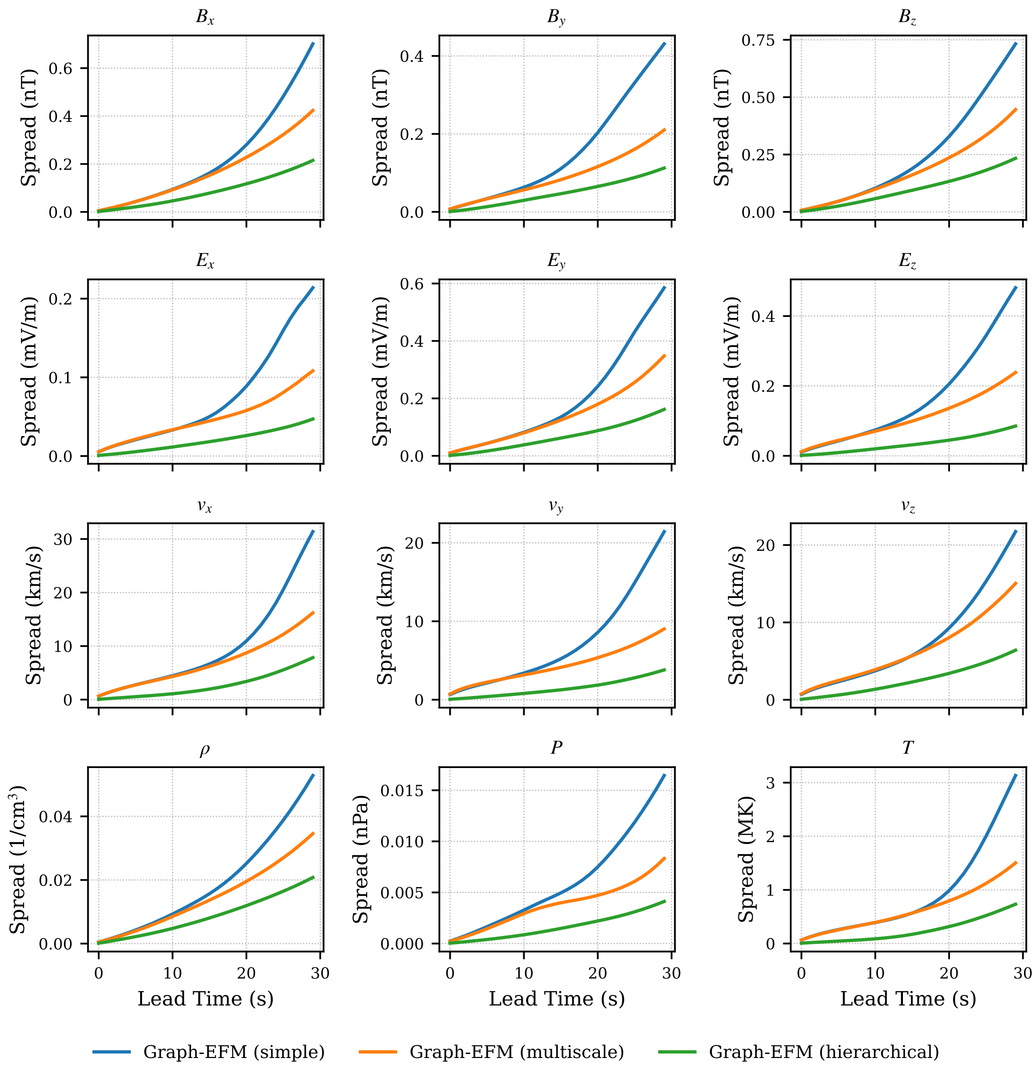


Figure 6: Ensemble spread for all predicted variables as a function of forecast lead time.

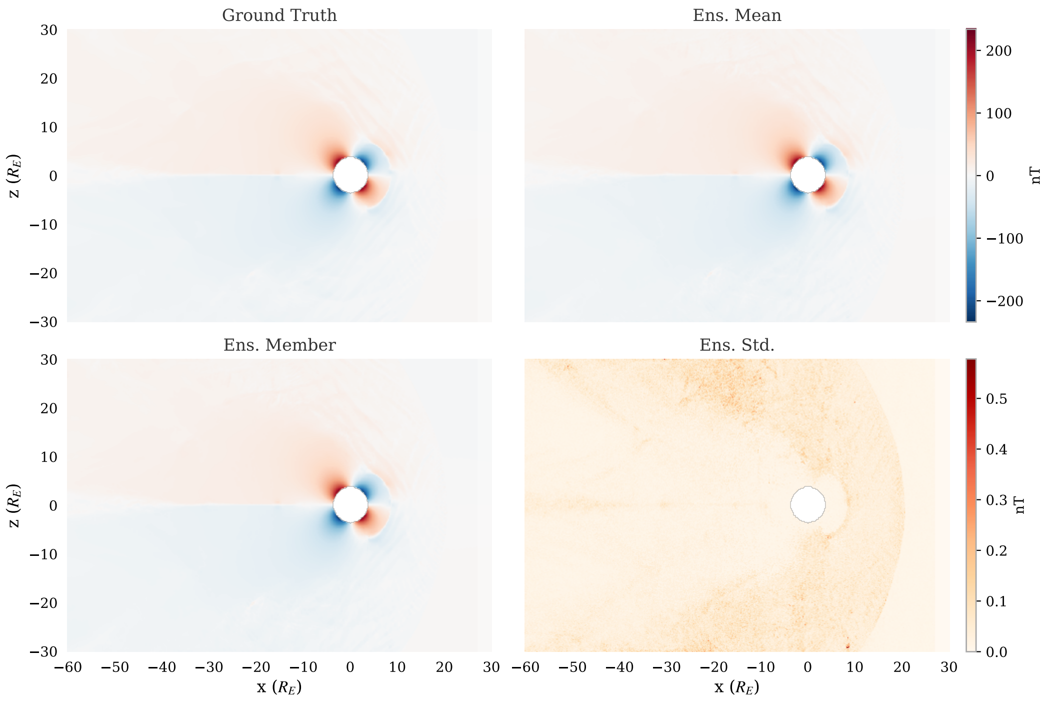


Figure 7: Magnetic field component B_x at timestep 10 from Graph-EFM (hierarchical).

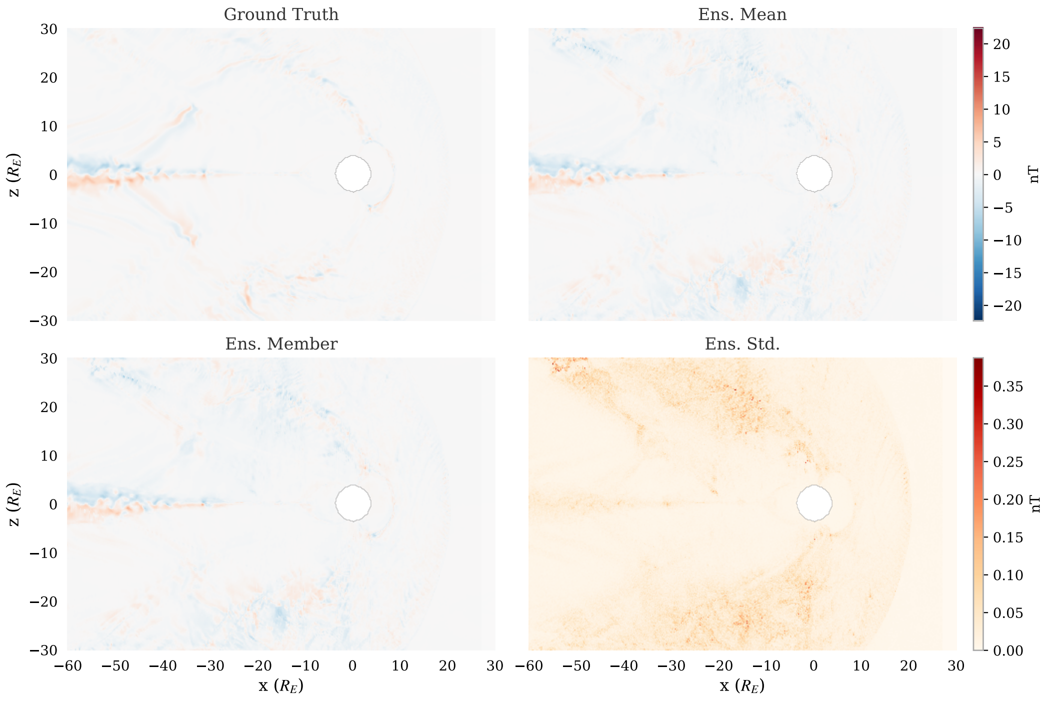


Figure 8: Magnetic field component B_y at timestep 10 from Graph-EFM (hierarchical).

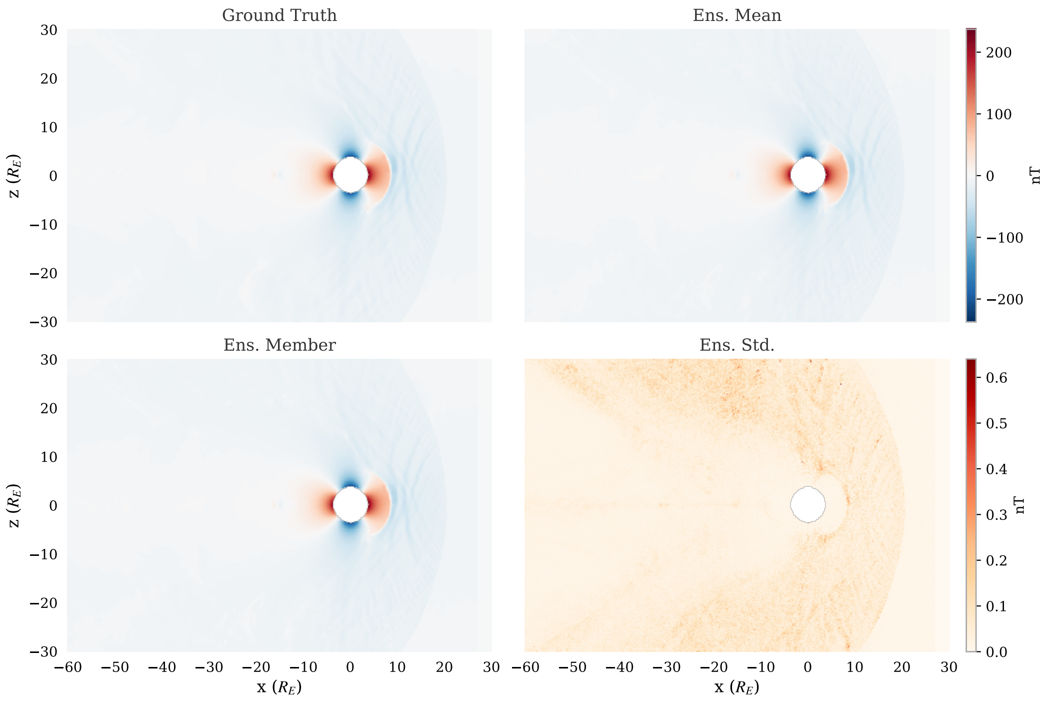


Figure 9: Magnetic field component B_z at timestep 10 from Graph-EFM (hierarchical).

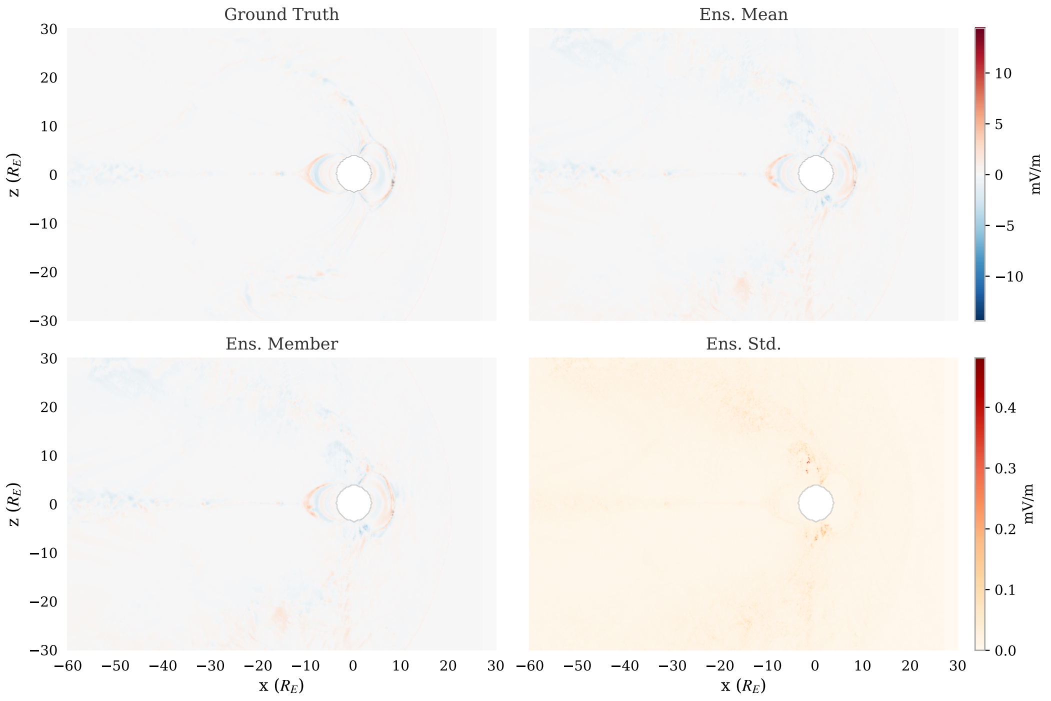


Figure 10: Electric field component E_x at timestep 10 from Graph-EFM (hierarchical).

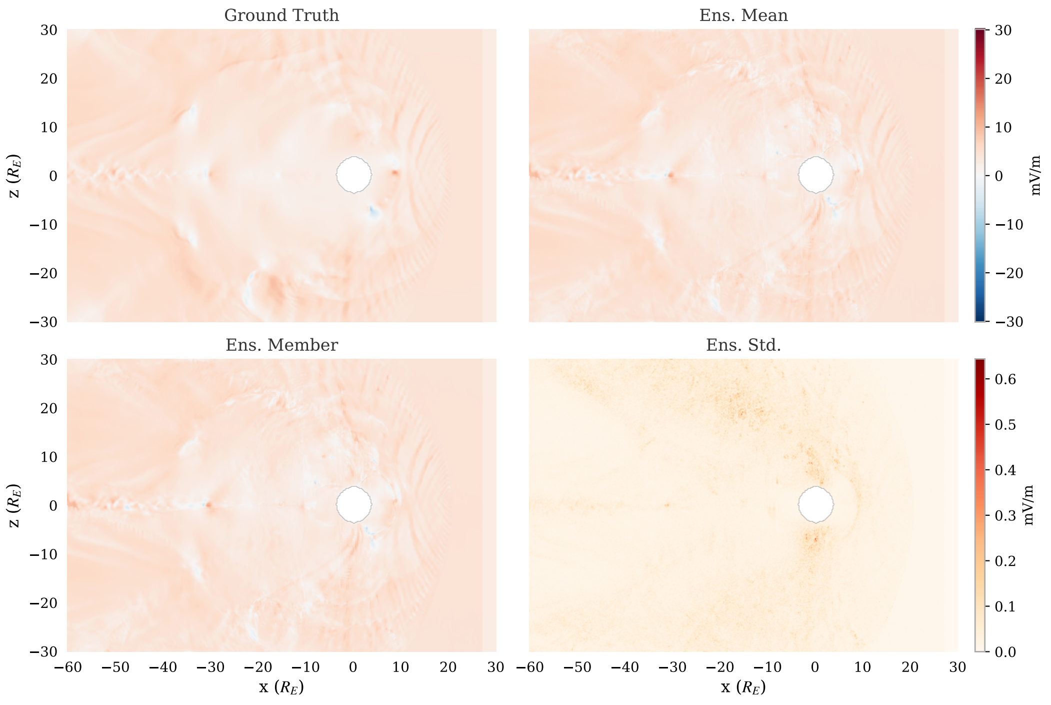


Figure 11: Electric field component E_y at timestep 10 from Graph-EFM (hierarchical).

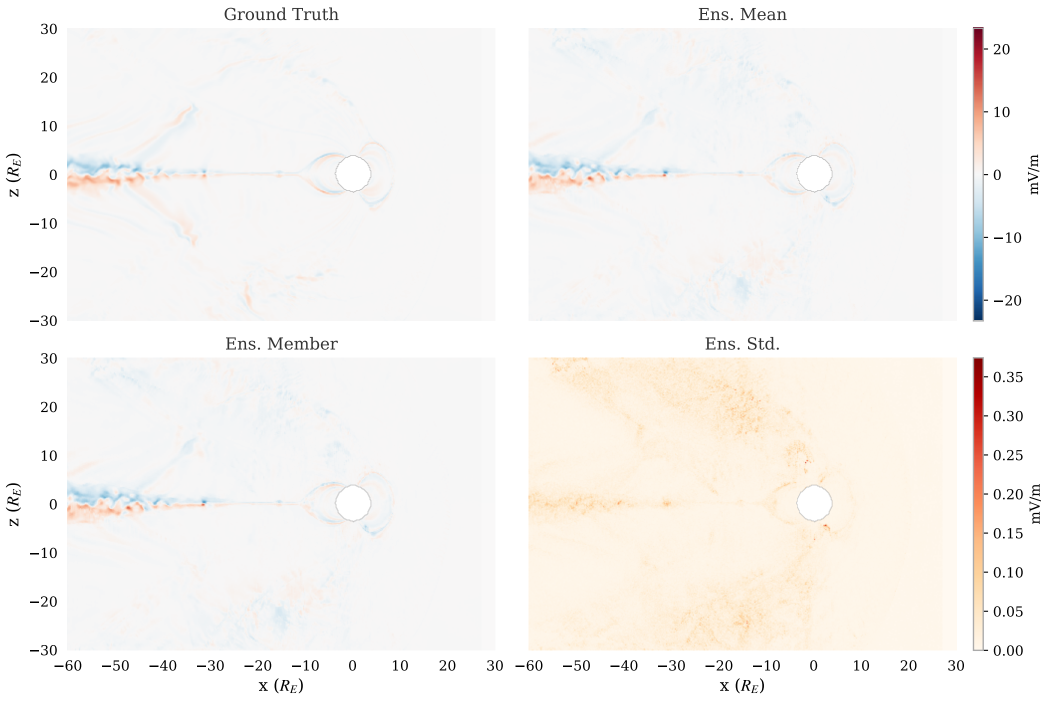


Figure 12: Electric field component E_z at timestep 10 from Graph-EFM (hierarchical).

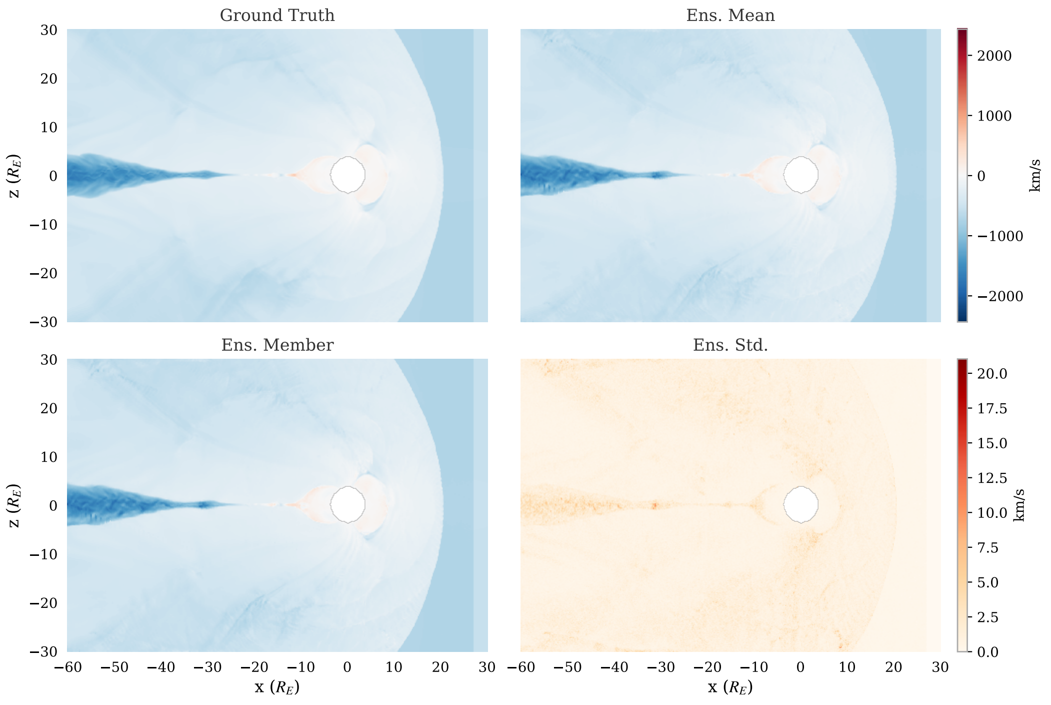


Figure 13: Velocity component v_x at timestep 10 from Graph-EFM (hierarchical).

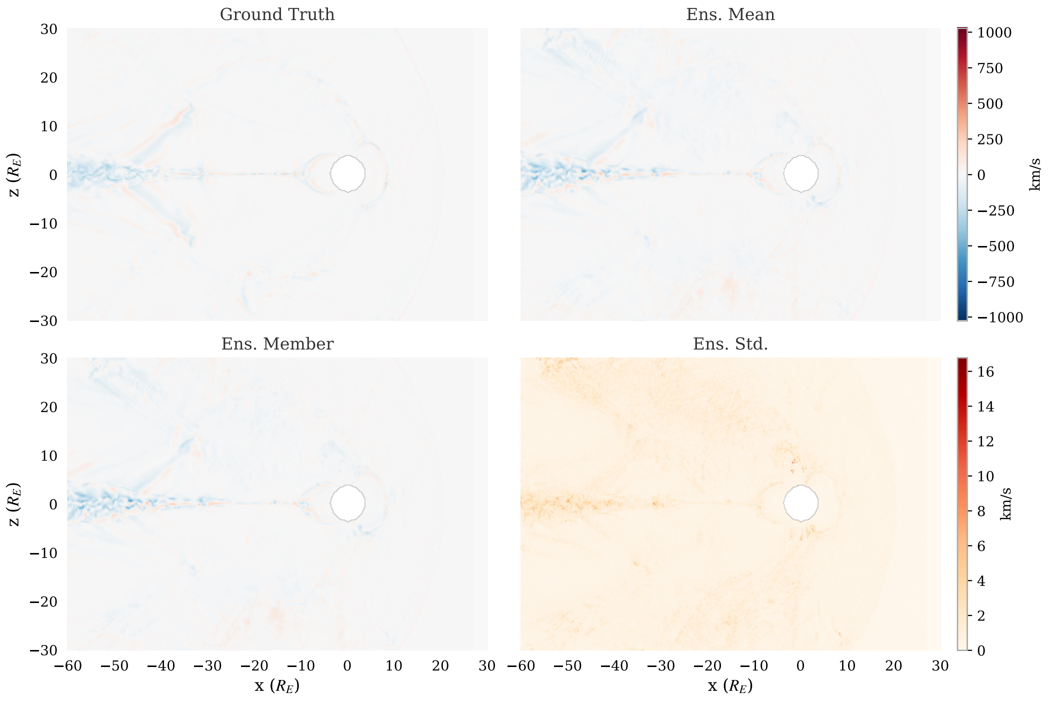


Figure 14: Velocity component v_y at timestep 10 from Graph-EFM (hierarchical).

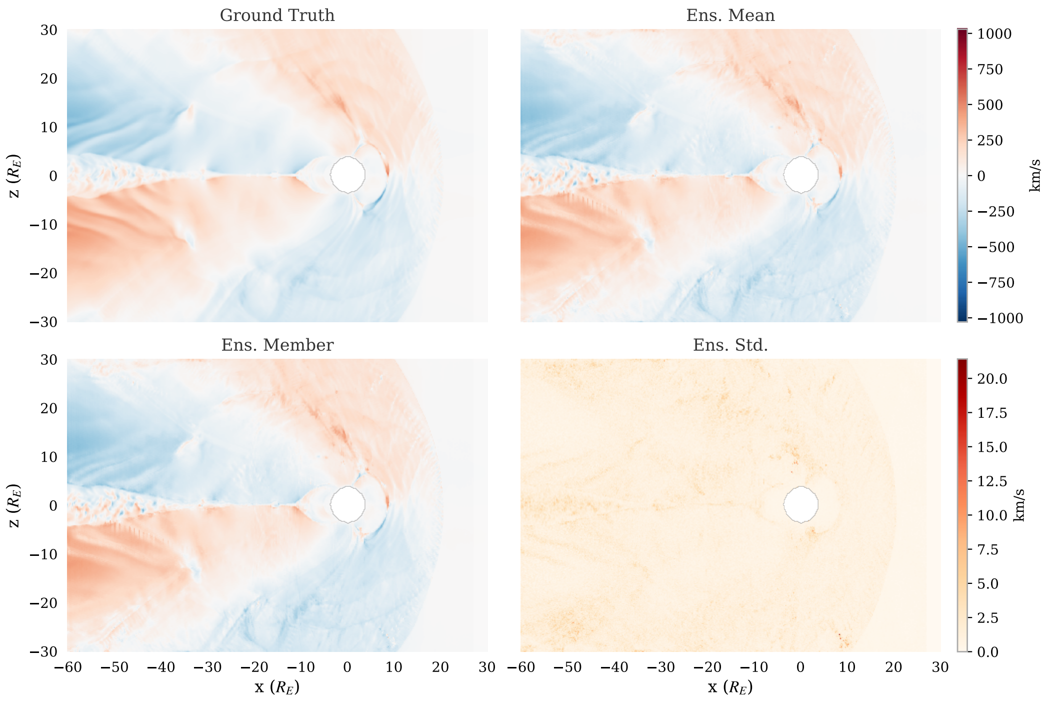


Figure 15: Velocity component v_z at timestep 10 from Graph-EFM (hierarchical).

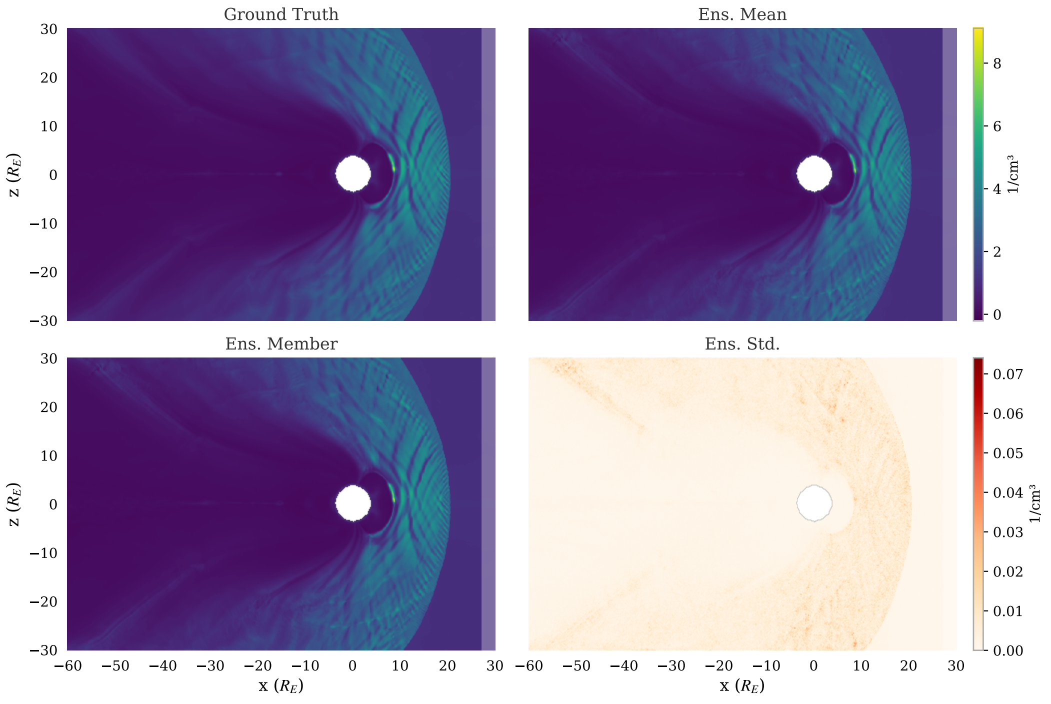


Figure 16: Particle number density ρ at timestep 10 from Graph-EFM (hierarchical).

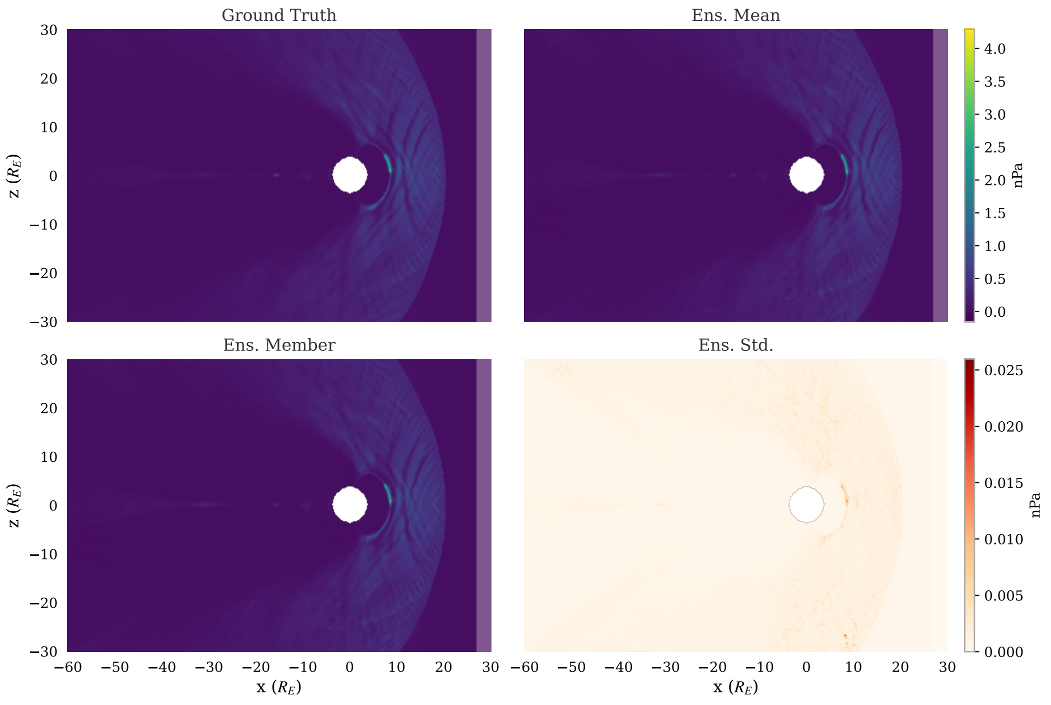


Figure 17: Plasma pressure P at timestep 10 from Graph-EFM (hierarchical).

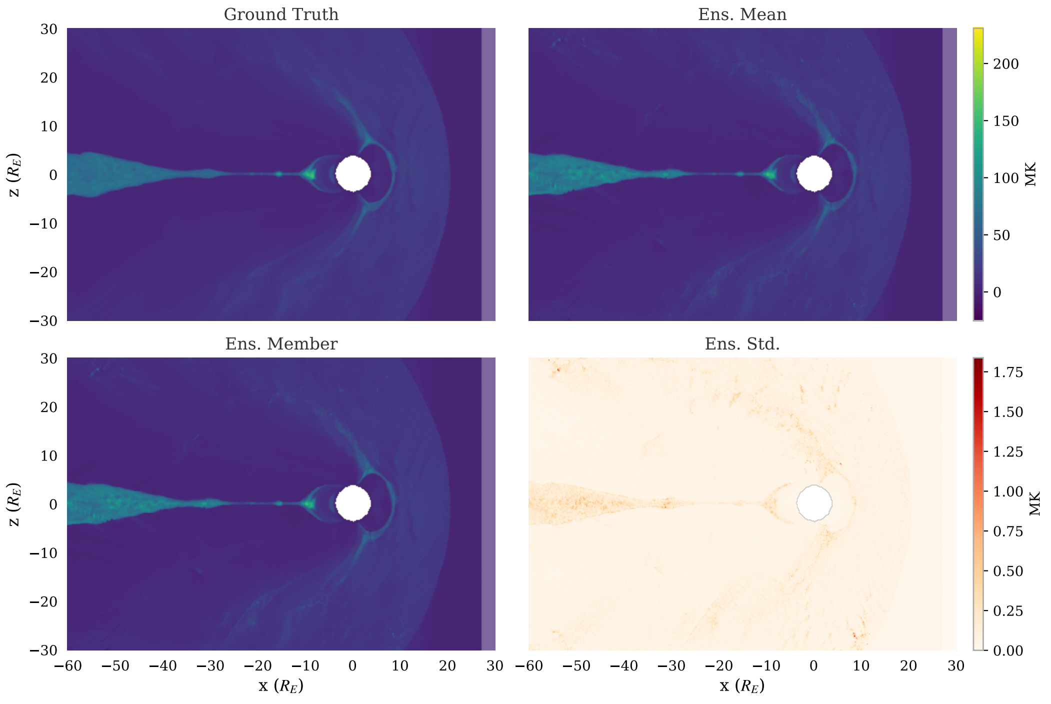


Figure 18: Plasma temperature T at timestep 10 from Graph-EFM (hierarchical).

Appendix references

[28] Gianluca Galletti, Fabian Paischer, Paul Setinek, William Hornsby, Lotenzo Zanisi, Naomi Carey, Stanislas Pamela, and Johannes Brandstetter. 5D neural surrogates for nonlinear gyrokinetic simulations of plasma turbulence. In *ICLR 2025 Workshop on Tackling Climate Change with Machine Learning*, 2025.

[29] Chris Pedersen, Laure Zanna, and Joan Bruna. Thermalizer: Stable autoregressive neural emulation of spatiotemporal chaos. In *International Conference on Machine Learning*, 2025.

[30] Martin Andrae, Tomas Landelius, Joel Oskarsson, and Fredrik Lindsten. Continuous ensemble weather forecasting with diffusion models. *International Conference on Learning Representations*, 2025.

[31] Urs Ganse, Tuomas Koskela, Markus Battarbee, Yann Pfau-Kempf, Konstantinos Papadakis, Markku Alho, Maarja Bussov, Giulia Cozzani, Maxime Dubart, Harriet George, et al. Enabling technology for global 3D + 3V hybrid-Vlasov simulations of near-Earth space. *Physics of Plasmas*, 30(4), 2023.

[32] Leo Kotipalo, Markus Battarbee, Yann Pfau-Kempf, and Minna Palmroth. Physics-motivated cell-octree adaptive mesh refinement in the Vlasiator 5.3 global hybrid-Vlasov code. *Geoscientific Model Development*, 17(16):6401–6413, 2024.

[33] Eirini Angeloudi, Jeroen Audenaert, Micah Bowles, Benjamin M Boyd, David Chemaly, Brian Cherinka, Ioana Ciucă, Miles Cranmer, Aaron Do, Matthew Grayling, et al. The multimodal universe: Enabling large-scale machine learning with 100 TB of astronomical scientific data. In *Advances in Neural Information Processing Systems*, 2024.

[34] Ruben Ohana, Michael McCabe, Lucas Meyer, Rudy Morel, Fruzsina Agocs, Miguel Beneitez, Marsha Berger, Blakesly Burkhart, Stuart Dalziel, Drummond Fielding, et al. The Well: A large-scale collection of diverse physics simulations for machine learning. In *Advances in Neural Information Processing Systems*, 2024.

[35] Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, et al. Multiple physics pretraining for spatiotemporal surrogate models. In *Advances in Neural Information Processing Systems*, 2024.

[36] Harris Abdul Majid, Pietro Sittoni, and Francesco Tudisco. Solaris: A foundation model of the sun. In *NeurIPS 2024 Workshop on Foundation Models for Science*, 2024.

[37] Cristian Bodnar, Wessel P Bruinsma, Ana Lucic, Megan Stanley, Anna Allen, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan A Weyn, Haiyu Dong, et al. A foundation model for the Earth system. *Nature*, 641(8065):1180–1187, 2025.

[38] Sujit Roy, Johannes Schmude, Rohit Lal, Vishal Gaur, Marcus Freitag, Julian Kuehnert, Theodore van Kessel, Dinesha V. Hegde, Andrés Muñoz-Jaramillo, Johannes Jakubik, et al. Surya: Foundation model for heliophysics. *arXiv preprint arXiv:2508.14112*, 2025.

[39] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[40] Jimmy Ba, Jamie Kiros, and Geoffrey Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[41] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[42] Michaël Zamo and Philippe Naveau. Estimation of the continuous ranked probability score with limited information and applications to ensemble weather forecasts. *Mathematical Geosciences*, 50(2):209–234, 2018.