Progress On A Machine Learning Surrogate For The MuSHrooM Plasma Turbulence Code

Ruben Carpenter, Ian Gill, Lukass Kellijs
Yale University

Abstract

The extremely large quantity of computational resources and walltime required to perform gyrokinetic simulations of plasma turblence in fusion energy devices is a major barrier to both reactor design as well as basic plasma physics research. Recent advances in scientific machine learning techniques present an opportunity to massively speed up these simulations by training deep neural network on a relatively small amount of turbulence data to accurately continue a simulation at a much faster rate. In this work, we implement such a model using a CVAE-DNN structure, and evaluate it on electric potential data generated by the 2D gyrofluid turbulence code, MuSHrooM. We find that this model is able to accurately capture the Fourier spectrum of the MuSHrooM model on testing data, while requiring only a small fraction of the fraction of the compute time used to generate the original data. We also explore implementing a

1 Introduction

The foremost approach for achieving controlled fusion energy is the tokamak, a device that confines high-temperature plasma in a toroidal magnetic field configuration [9]. A key operational challenge is plasma turbulence, which drives complex microinstabilities and generates turbulent structures that significantly enhance anomalous transport processes. Advanced control of turbulence through optimized magnetic geometry and plasma management is critical to developing efficient, self-sustaining tokamak reactors.

Gyrokinetics is the primary theoretical framework for simulating turbulence in magnetically confined plasmas. This approach leverages a reduced-dimensional description by averaging over the rapid gyro-motion of charged particles around magnetic field lines, isolating the drift kinetics responsible for turbulent transport. However, gyrokinetic simulations remain inherently multiscale due to the large mass disparity between electrons and ions, with electrons requiring much finer spatial and temporal resolution. This results in a computationally demanding problem that often requires on the order of tens of millions of core-hours on high-performance computing systems.

Our objective is to develop a machine learning surrogate model for a particular gyrofluid simulation code, known as MuSHrooM, which evolves the state of a plasma under a system of four coupled PDEs on a doubly periodic 2D grid. This idea is based on the work (see the reference paper [4]) done by del Castillo-Negrete et al. on the simpler Hasegawa-Wakatani model [7].

In paricular, it should iteratively produce accurate predictions for the electric potential, ion temperature, ion density, electron temperature, and electron density at the next time step, given the initial state of all of these fields. Since MuSHrooM simulations can take hundreds of hours to run, the surrogate model could be trained on a fraction of the total data we wish to generate, and then generate the remaining data at multiple orders of magnitude speedup. Additionally, if one wished to test various initial conditions for the same plasma parameters, only one time intensive MuSHrooM simulation would be needed, and the surrogate model could perform the rest.

2 MuSHrooM and Training Data Generation

The MuSHrooM data we will be using for this project is generated on the MIT engaging cluster using 16 CPU nodes, each with 32 cores.

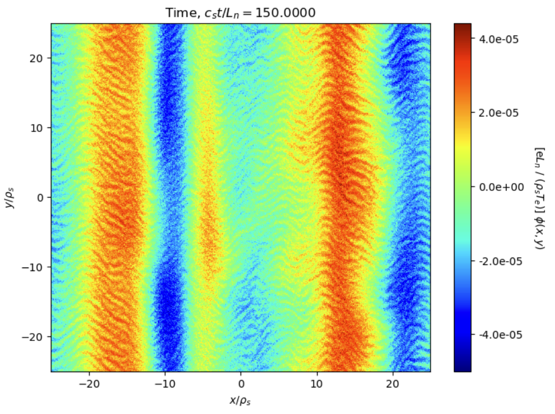

Figure 1: Example of a time snapshot of MuSHrooM generated electric potential

One set of data comes from a MuSHrooM simulation up to a time of $\frac{c_s t}{L_n} = 230$, outputting all five fields every 0.1 time units. To ensure that we have reached saturation, we use the last 1292 time slices for the training and testing of our model. The Fourier resolution on our simulation was 340 by 340, corresponding to a spatial resolution of 683 by 682. The exact parameters used in our simulation are listed in the appendix. Another set of data was generated in which the simulation evolved from $\frac{c_s t}{L_n} = 95$ to $\frac{c_s t}{L_n} = 115$, and all five fields were output every 0.01 time units. The former dataset was found to produce better results, so it was with this dataset that we proceeded.

3 Model Architecture

To demonstrate the feasibility of using machine learning for solving the coupled MuSHrooM equations, we begin by focusing on a simplified version of the problem. While the MuSHrooM equations govern the evolution of five coupled fields, for the sake of a proof of concept, we will train our surrogate model to predict the electric field $\phi(x, y, t)$ from only the $\phi(x, y, t-1)$, the electric at the previous time step.

This involves reducing the resolution of each slice to 64×64, and training a CVAE network to encode this data into a latent space of size 64, where we will train a fully connected network to predict the next latent space values given the current ones. Once this is done for ϕ, we can then implement the same structure for the other four fields. As we describe the next section, a large amount of this work has already been implemented.

There are two main problems with this approach: we lose a massive amount of resolution by reducing the size of our grid to 64×64, and we are not taking advantage of the natural coupling between the five fields we are investigating. There are two techniques from modern scientific machine learning which we are considering applying to remedy these issues.

4 GAIT Model

The GAIT (Generative Artificial Intelligence Turbulence) model [4] consists of a convolutional variational autoencoder (CVAE) and a coupled dense neural network (DNN).

4.1 CVAE network

The CVAE includes an encoder $\mathcal{E}$ that compresses input data to a latent representation and a decoder $\mathcal{D}$ that reconstructs the data [8, 3]. The encoder architecture (Table 1) implemented uses 5 convolutional layers with batch normalization to efficiently downsample and extract key features from the plasma data.

Table 1: Summary of Encoder $\mathcal{E}$ Network

Layer	**Input Shape**	**Output Shape**
Conv1	$1 \times 64 \times 64$	$32 \times 32 \times 32$
Conv2	$32 \times 32 \times 32$	$64 \times 16 \times 16$
Conv3	$64 \times 16 \times 16$	$128 \times 16 \times 16$
Conv4	$128 \times 16 \times 16$	$256 \times 8 \times 8$
Conv5	$256 \times 8 \times 8$	$256 \times 8 \times 8$
Flatten	$256 \times 8 \times 8$	16384
FC1 (μ)	16384	128
FC2 ($\log \sigma^2$)	16384	128

The output is then flattened and passed through two fully connected layers to produce the $N = 128$ dimensional latent space parameters: mean (μ) and log variance ($\log \sigma^2$). These outputs define a Gaussian distribution over the latent space from which the latent space representation z of the plasma data sample is then sampled.

This is done through reparametriation whereby the reparameterization function enables stochastic sampling from the latent space by generating a sample centered at the mean (μ) with variance ($\log \sigma^2$) in a differentiable manner. It calculates the standard deviation (σ) from $\log \sigma^2$, samples random noise (ϵ), and shifts this noise by μ, scaled by σ, as $z = \mu + \sigma \cdot \epsilon$. This generates points $\mathbf{z}$ in the N-dimensional latent space.

The decoder $\mathcal{D}$ in the CVAE model then reconstructs plasma field data from the latent representation, acting as a mirror-network of the encoder. A sigmoid activation is applied at the final layer to constrain the reconstructed output within a suitable range. This architecture effectively reverses the encoding process, transforming the latent distribution back into a detailed spatial representation of the plasma field.

4.2 DNN Network

The predictive nature of the GAIT model arises from the coupled DNN. After the encoder $\mathcal{E}$ and reparametrization is used to obtain a latent space representation z_i of an image ϕ_i in the simulation sequence, the DNN model is used to predict the next latent space representation z_{i+1} in the sequence, which through the decoder $\mathcal{D}$ then generates the next simulation step ϕ_{i+1}.

The DNN implemented in this paper (Table 2) is a simple fully connected network with 5 layers and the ReLu activation function.

Table 2: Summary of DNN

Layer	Input Shape	Output Shape
FC1	128	128
FC2	128	256
FC3	256	256
FC4	256	128
FC5	128	128

4.3 Model Training

The model training consists of two parts. First, the CVAE network is trained on a sample of $n_{train} = 1216$ images to ensure proper generation of latent-space representation z and reconstruction of ϕ images. Then the DNN network is trained using the generated latent space vectors z of the CVAE which form $(1216 - 1)$ pairs of sample and target (z_i, z_{i+1}).

Our loss function for the CVAE training sequence follows the form discussed in [4] and combines multiple terms to balance accurate reconstruction with regularization and smoothness of spatial features.

$$\begin{aligned}\mathcal{L} = w_\phi \cdot \sum ||\hat{x} - x||^2 \, + \, w_\nabla \cdot ||\nabla \hat{x} - \nabla x||^2 \\ + \, w_{\text{kl}} \cdot \left(-\frac{1}{2} \sum \left(1 + \log(\sigma^2) - \mu^2 - \sigma^2\right) \right)\end{aligned} \tag{1}$$

The total loss $\mathcal{L}$ in (1) is composed of the reconstruction loss, the gradient loss, and the KL divergence, each weighted by their respective coefficients $w_\phi = 1$, $w_\nabla = 10.0$, and $w_{\text{KL}} = 0.01$. The reconstruction loss $\sum ||\hat{x} - x||^2$ minimizes mean squared error (MSE) between x and its reconstruction $\hat{x}$, while the gradient loss $||\nabla \hat{x} - \nabla x||^2$ penalizes differences between the gradients of input and reconstructed output, promoting smoothness and alignment between them. Finally, the KL divergence term encourages the distribution of the latent space to match a standard normal distribution

The DNN uses a simple loss function (2) to minimize differences between the prediction of the next latent space representation $DNN(z_i)$ and the known target z_{i+1}.

$$\mathcal{L}_{\text{DNN}} = \sum ||z_{i+1} - DNN(z_i)||^2 \tag{2}$$

The CVAE and DNN models were both trained with 1000 epochs using a batch size of 32 and learning rate 1×10^{-3}.

5 Analysis of GAIT results

In this section, we present the results of our GAIT model for plasma turbulence prediction. Our model demonstrates the ability to generate time evolution predictions that closely resemble the ground truth. We generated animations of iterated GAIT predictions on both the training data (see here) and the test data (see here). Figures 2 and 3 display two representative samples from the training and test datasets, respectively.

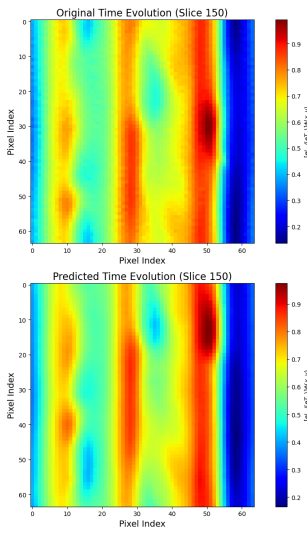

Figure 2: Training Data

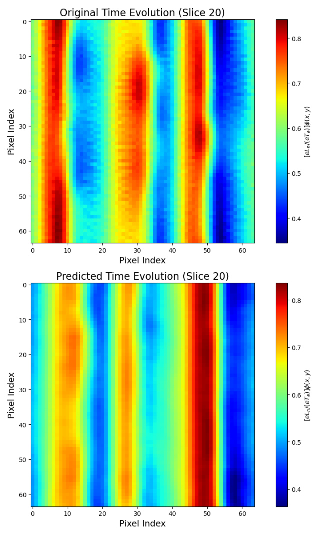

Figure 3: Test Data

It is noteworthy that while the MuSHrooM model requires approximately 50 hours to generate 1311 training time slices, GAIT achieves the same number of forward passes in just 40 seconds. This significant reduction in computation time showcases the efficiency of GAIT without compromising on predictive quality.

The dynamics of plasma flow are inherently complex, necessitating models that capture a wide range of physical phenomena. For this reason, a high-fidelity model like GAIT must not only replicate the broad behavior of plasma turbulence but also preserve its more intricate features. To rigorously validate GAIT's performance, we conduct Fourier Spectrum Analysis and Proper Orthogonal Decomposition (POD) tests.

5.1 Fourier Spectrum analysis (FSA)

Due to its turbulent nature, plasma flow often exhibits a broad spectrum of frequency components, and it is crucial that our model can reliably represent this broad range of frequencies. To assess the fidelity of our model in this regard, we performed a Fourier power spectrum analysis on the fluctuations in the electric field

$$\delta E(x, y, t) := E(x, y, t) - \langle E \rangle_{x,y}(t).$$

Recall that GAIT predicts the electric potential $\phi(x, y, t)$, so the electric field E can be computed via automatic differentation.

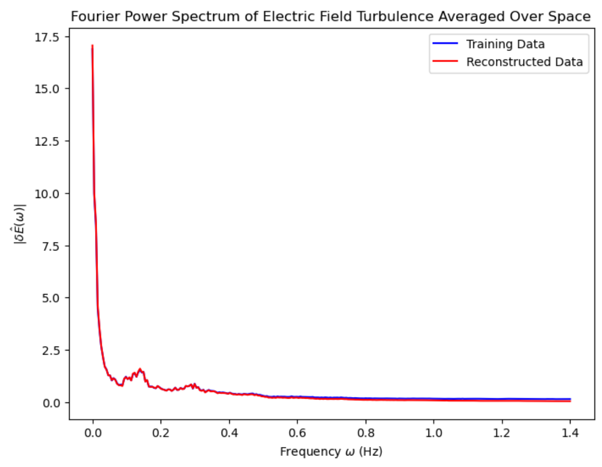

Figure 4: Fourier Power Spectrum of Electric Field Turbulence Averaged Over Space. Comparison for the training data predictions

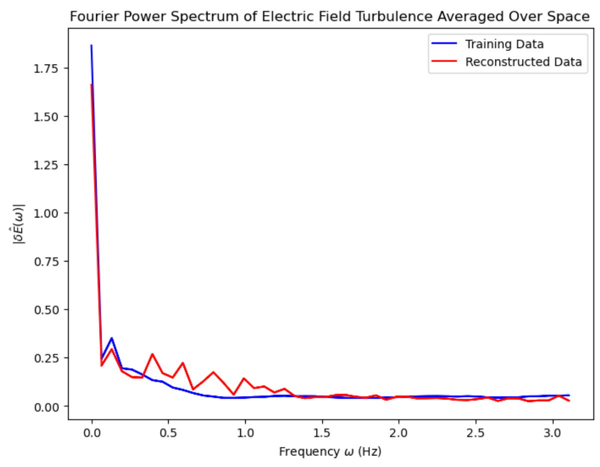

Figure 5: Fourier Power Spectrum of Electric Field Turbulence Averaged Over Space. Comparison for the training data predictions

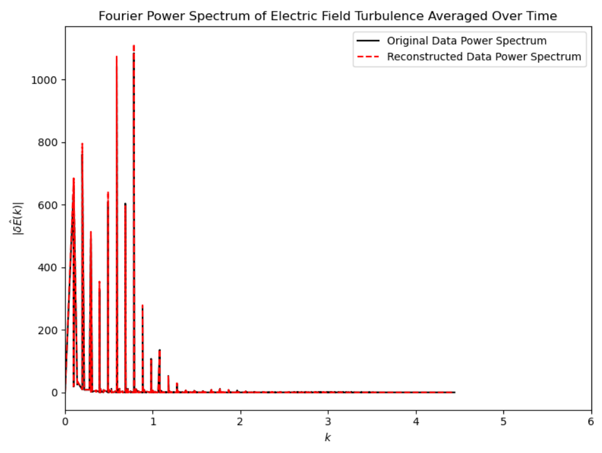

Figure 6: FSA analysis over space of the Electric field turbulent fluctuations, in all the testing data. The data for ϕ contained $T = 900$ time steps.

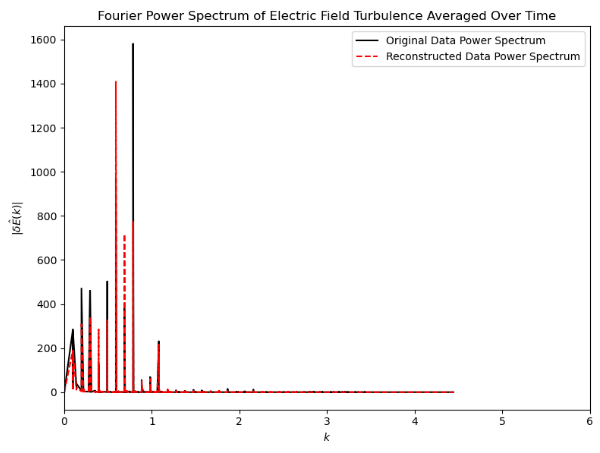

Figure 7: FSA analysis over space of the Electric field turbulent fluctuations, in all the training data. The data for ϕ contained $T = 900$ time steps.

In the first part of the analysis, we focus on the temporal fluctuations of the electric field. For each fixed spatial coordinate (x, y) we compute the Fourier transform of the electric field fluctuations over time t. The result is then averaged over all spatial coordinates to obtain the power spectrum as a function of frequency. Figures 4 and 5 show the comparison of the Fourier power spectra over time for the ground truth and the predictions, for both training and testing timeslices.

In the second part of the analysis, we examine the spatial frequency components of the electric field fluctuations. For each time slice t we compute the Fourier transform $|\widehat{\delta E}(\mathbf{k}, t)|$ over space. The result is the averaged over all time slices, which can be interpreted as a function of the modulus $\|\mathbf{k}\| = \sqrt{k_x^2 + k_y^2}$. Figures 6 and 7 show the comparison of the Fourier power spectra over space for the ground truth and the predictions, for both training and testing timeslices.

The results clearly demonstrate that for both tests, the agreement between the ground truth and our model's predictions is excellent. This indicates that our model not only captures the general trends of the system but also learns the more subtle aspects of the dynamics. We hypothesize that the observed spikiness in Figure 5 is primarily due to the limited amount of testing data, and that with a larger dataset, the spectra would become smoother

5.2 Proper Orthogonal Decomposition

We now turn to a complementary approach for assessing our model's ability to capture the intricate dynamics of plasma turbulence. Two of the most significant factors having a critical role in plasma transport are the trapping effect of vortices and the long displacement caused by zonal flows [5, 10]. Accordingly, it is important to measure how well our model can capture these coherent, organised structures of the flow. Contrary to Fourier spectral analysis, Proper Orthogonal Decomposition (POD) is based on data-driven, empirical modes providing an optimal representation of the turbulence state in the energy norm [6]. We will decompose the electric potential into a set of deterministic spatial functions $u^{(l)}, v^{(l)}$ modulated by random time coefficients $w^{(l)}$

$$\phi(x, y, t) = \sum_{i=1}^{N} w_l(t)\phi^{(l)}(x, y), \tag{3}$$

$N = \min(t, N_x \times N_y)$.

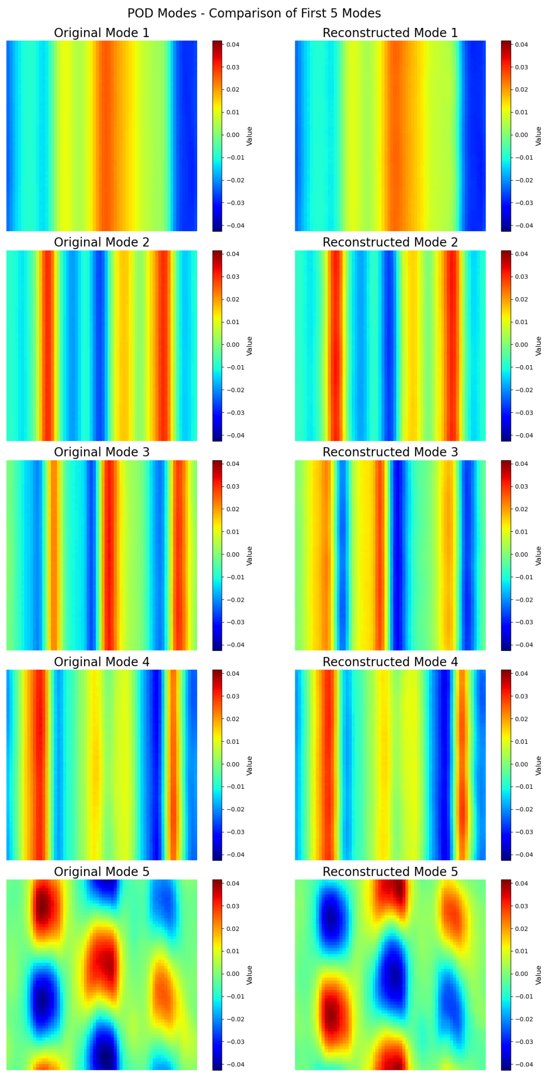

Figure 8: First 5 modes in Proper Orthogonal Decomposition for predictions on training data (1216 time slices)

As can be seen in Figure 8, our model succcessfully captures the primary tendencies of plasma turbulence on both the training and testing datasets. However, as modes increase in order, the model's accuracy declines, with higher-order modes showing more noise and mixed features (for modes 5 and above). We believe this arises from two factors. First, the model's complexity may simply be insufficient to fully capture all the intricate details of the higher-order modes, which are more sensitive to fine-scale fluctuations. Secondly, we note that the model currently bases its prediction of the electric field at the next time step solely on the present electric potential. However, according to the MuSHrooM equations, this provides insufficient physical information to accurately predict the subsequent time steps, why may result in the discrepancies we observe in higher-order modes. These limitations highlight the need for a more sophisticated model architecture and possibly additional input fields to improve the prediction of more complex, fine-scale dynamics. We discuss our plans for improving this in the Further Work section.

6 Alternative Approach: Laplace Neural Operator

CNNs are limited in terms of the resolution of input data which they are capable of handling, which motivates investigating if an alternative architecture could perhaps scale to modeling the full resolution of a MuSHrooM simulation. Laplace Neural Operators (LNOs) are a type of neural operator that leverages the Laplace transform to efficiently learn mappings between function spaces for solving partial differential equations, and in principle is not as strongly constrained by input data compared to CNNs [2]. The LNO passes its input through two separate networks that are designed to handle a transient and periodic response separately, which is well suited to our circumstance.

For the time being, we aim to accomplish these tasks in the same 64×64 data resolution we used for GAIT so that we can compare the performance, but no edits to the model are needed to use a higher resolution. There are two use cases for a Laplace Neural Operator in accomplishing our objective of constructing an accurate surrogate model for MuSHrooM.

- We can directly use an LNO to perform time evolution: we provide it will all five fields at a given time step, and have it predict all five fields at the next time step.
- We can use train LNO to understand the coupling between the five fields; that is, provide it with four of the fields and have it predict the fifth.

While neither of these approaches have been fully developed, preliminary results have been generated on both fronts.

Using an LNO as described in [2], we find that we are able to produce a visually similar time slice of the potential based on the data for all five fields at the previous time slice,as seen in Figure 9. Note that this is not the result of iterative evolution as with the GAIT model, but just a single prediction based on ground truth data. Further development is needed to adapt this model to predict all five fields and thus allow for iterative predictions.

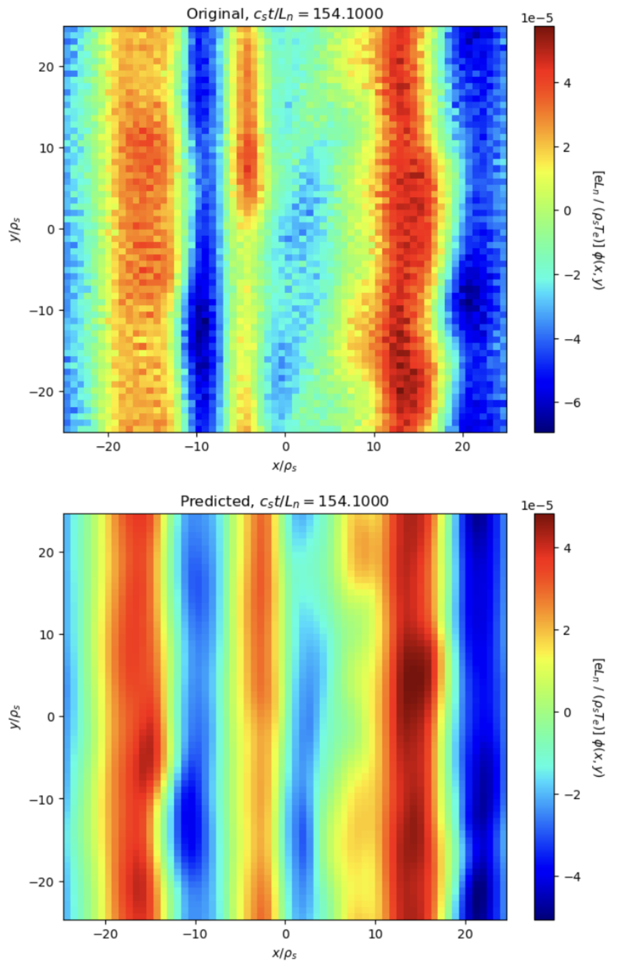

Figure 9: Prediction of potential at $\frac{c_s t}{L_n} = 154$ given all five fields at $\frac{c_s t}{L_n} = 153.9$ using LNO.

Additionally, we were able to train an LNO to predict the potential given the other four fields at a given time slice, and the result of this is shown in Figure 10. We see good visual agreement, indicating that this method has promise to characterizing the coupling between the fields, although further work is needed to assess the LNOs performance quantitatively.

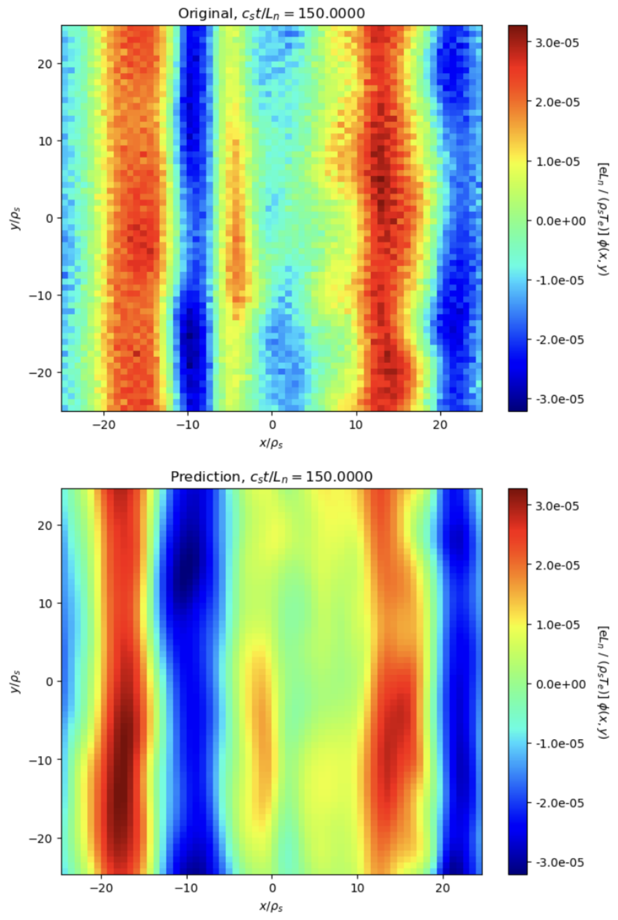

Figure 10: Prediction of potential at $\frac{c_s t}{L_n} = 150$ given the other four fields at $\frac{c_s t}{L_n} = 150$ using LNO.

7 Future Work

Our results provide a strong proof of concept for the viability of using scientific machine learning tools to simulate plasma gyrofluid simulations, particularly within the context of MuSHrooM. The GAIT model has proven effective at predicting general trends in plasma behaviour, producing visually accurate time-evolution predictions forcasted over more than 1200 time slices. this robustness highlights the potential of this approach significantly speeding up simulations.

However, despite its success in capturing the general dynamics, GAIT misses some of the finer details, as demonstrated by the Proper Orthogonal Decomposition (POD) test. This discrepancy suggests that while GAIT is capable of modeling large-scale flow features, it currently lacks the complexity needed to fully capture the finer, higher-order modes of the plasma dynamics.

Moving forward, we plan to extend our model to incorporate all five fields in which MuSHrooM evolves, recognizing that the current electric potential ϕ is insufficient to fully capture the dynamics of the system. To achieve this, we will utilize a DeepM&Net architecture [1] within the latent space to ensure self-consistency in the time evolution of the fields.

Additionally, we aim to fully develop the LNO time evolution model to predict the development of all fields, allowing it to be run iteratively and perform a quantitative analysis of its performance, as we did for GAIT. We also plan to train LNOs to predict the ion temperature, ion density, electron density, and electron temperature fields from the other four, as we did for the potential. These LNOs can then be used to guide training of a time evolution model to ensure self consistency.

Through the progress discussed in this paper and the planned improvements, we aim to significantly enhance the scalability and efficiency of plasma simulations, paving the way for faster, more accurate predictions of plasma dynamics.

References

[1] Shengze Cai, Zhicheng Wang, Lu Lu, Tamer A Zaki, and George Em Karniadakis. Deepm&mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *Journal of Computational Physics*, 436:110296, 2021.

[2] Qianying Cao, Somdatta Goswami, and George Em Karniadakis. Laplace neural operator for solving differential equations. *Nature Machine Intelligence*, 6(6):631–640, June 2024.

[3] M Cheng, F Fang, CC Pain, and IM Navon. An advanced hybrid deep adversarial autoencoder for parameterized nonlinear fluid flow modelling. *Computer Methods in Applied Mechanics and Engineering*, 372:113375, 2020.

[4] B. Clavier, D. Zarzoso, D. del Castilllo-Negrete, and E. Frenod. A generative machine learning surrogate model of plasma turbulence. *arXiv preprint arXiv:2405.13232*, 2024.

[5] Diego del Castillo-Negrete. Chaotic transport in zonal flows in analogous geophysical and plasma systems. *Physics of Plasmas*, 7(5):1702–1711, 2000.

[6] S. Futatani, S. Benkadda, and D. del Castillo-Negrete. Spatiotemporal multiscaling analysis of impurity transport in plasma turbulence using proper orthogonal decomposition. *Physics of Plasmas*, 16:042506, 2009.

[7] Akira Hasegawa and Masahiro Wakatani. Plasma edge turbulence. *Physical Review Letters*, 50(9):682, 1983.

[8] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[9] VS Mukhovatov and VD Shafranov. Plasma equilibrium in a tokamak. *Nuclear Fusion*, 11(6):605, 1971.

[10] Joel Sommeria. Experimental study of the two-dimensional inverse energy cascade in a square box. *Journal of fluid mechanics*, 170:139–168, 1986.

8 Appendix

8.1 MuSHrooM input file

Note that although dt is listed here as 0.001, MuSHrooM has adaptive timestepping, so this is only the initial time step. It should be noted that MuSHrooM is currently not public, although it is expected to become so after an initial publication which will hopefully occur in the spring.

```
&spaceIn
NkxG    = 340
NkyG    = 340
kxMin   = 0.1256
kyMin   = 0.1256
dt      = 0.001
endTime = 190
nFrames = 300
xyProcs = 32, 16
kMaxDyn = 0,0
/

&physIn
omSte       = 1.0
omde        = 1.818182
tau         = 1.0
muMass      = 60.59
deltae      = 2.0
deltaPerpe  = 1.0
eta_e       = 12.42
deltai      = 2.0
deltaPerpi  = 1.0
eta_i       = 12.42
delta0      = 0.0
lambdaD     = 0.0
alpha_e     = 0.035, 0.0
alpha_i     = 0.035, 0.0
nu_e        = 0.004, 0.0
nu_i        = 0.004, 0.0
/

&diffIn
hDiffOrder = 4
hDiffne    = 10.00
hDiffTe    = 10.00
hDiffni    = 10.00
hDiffTi    = 10.00
kDiffMin   = 0.0, 0.0
/

&icsIn
initialOp  =  1
initAuxX   = -3
initAuxY   = -3
initA      = 2.5e-2
```

```
noiseA     = 0.
/

&arkodeIn
kySplit       = 0
microTableExp = 2
microTableImp = 104
fastTableExp  = 2
fastTableImp  = 104
slowTable     = 201
dtAvgMin      = 1
dtAvgDur      = 3
dtAvgTol      = 0.0
dtMicro       = 0.0
dtFast        = 0.0
rtol          = 1.0e-4
atol          = 1.0e-8
ewtScaling    = 0
arkKxc        = 1340,  25
arkKyc        = 1340,  25
arkKxOrd      = 2.5, 1.5
arkKyOrd      = 2.5, 1.5
arkMux        = 0.0
arkMuy        = 0.0
arkSigmax     = 6.0
arkSigmay     = 6.0
outputStats   = 1
/
```

8.2 Github link

The Github with our code can be found here.