LOCO: Abstracting Spectral Numerical Integrators for PDEs into Neural Operators

Lenz Pracher
Ludwig-Maximilians-Universität München
lenz.pracher@tum.de

Takashi Matsubara
Hokkaido University
matsubara@ist.hokudai.ac.jp

Abstract

Neural operators promise fast surrogates for PDE dynamics, but their architectures rarely reflect how numerical integrators compute. We argue that importing structure from spectral time stepping into the Fourier Neural Operator (FNO) yields a beneficial inductive bias for operator learning. As a *case study* in this abstraction, we introduce LOCO (Local Convolutional Operator), a slightly modified FNO that (i) applies nonlinear transforms in Fourier Space to emulate pseudo-differential operators and (ii) uses convolutions in Fourier Space to approximate local field couplings. A lightweight *Hybrid* (FNO + LOCO) variant combines both architectures. On the 1D Burgers and KdV equatios, LOCO improves single-step errors over a Fourier Neural Operator (FNO) baseline, and the Hybrid further reduces single step error on the incompressible 2D Navier-Stokes in direct comparison with FNO. However, we observe rollout instabilities in the incompressible 2D Navier-Stokes case; we do not claim LOCO is a SOTA (State Of The Art) architecture. Rather, our results support the broader position that principled abstractions of established numerical integrators provide a productive design space for neural operators.

1 Introduction

Physical systems governed by differential equations are pervasive across the sciences. Partial differential equations (PDEs) are the dominant modeling tool in these domains, yet high-fidelity numerical solvers require fine discretization in time and space and can be expensive to compute. The recent success of machine learning has therefore sparked strong interest in *neural surrogates* that are differentiable and efficient at prediction time and may learn underlying dynamics not accessible via PDEs. Among these, *neural operators* learn mappings between function spaces and have emerged as promising PDE surrogates [8, 7]. Notably, the *Fourier Neural Operator* (FNO) learns a mode-dependent, discretization invariant transform in Fourier Space interleaved with pointwise nonlinearities in physical space [8, 3]. Such operators have enabled applications including global weather forecasting (FourCastNet) using adaptive Fourier Neural Operators [11, 4].

While neural-operator architectures have advanced, relatively little attention has been paid to encoding the inductive biases of traditional numerical integrators. Much recent progress emphasizes physics-informed losses [12, 5], training strategies [10] or engineering tricks. Some advances relevant to discrete time-step PDE solving include: *(i)* spectral-inspired frequency-domain operators (SINO) [14] with spectral priors and filtering; *(ii)* neural operators with localized differential/integral kernels inspired by stencil discretizations [9] or *(iii)* energy-consistent neural operators (ENO) [13] that bias learning toward conservation/dissipation laws via energy-based penalties. We suppose that our approach offers a minimal modification of FNO while offering similar insights as in references [14, 9, 2].

However an explicit discrete time stepping operator can be written down by iterative expansion of spectral numerical integrators with possible bounds on prediction errors [15]. Our goal is therefore to find a neural operator, which is inspired by this insight and evolves an input function along the

1st Workshop on Differentiable Systems and Scientific Machine Learning @ EurIPS 2025.

PDE trajectory in discrete time steps Δt. Traditional numerical methods achieve this by leveraging integration, and *we argue that building blocks from numerical integration offer a natural blueprint for neural operator architectures*.

In pseudo-spectral time stepping, differential operators act as *pointwise operations* in Fourier Space, whereas local field interactions in physical space induce *spectral convolutions* that couple neighboring modes. Inspired by this, we propose **LOCO (Local Convolutional Operator)**, which combines pointwise transforms in Fourier Space with spectral convolutions to approximate local interactions. We also consider a **Hybrid** architecture that splits channels between a standard FNO branch and a LOCO branch.

On the Burgers, KdV, and 2D incompressible Navier-Stokes equation, we observe consistent one-step gains over an FNO baseline, with further improvements by employing the Hybrid Model. These benefits persist across different numerical integrators, suggesting that computational patterns from integration indeed provide a useful inductive bias. A limitation that remains is autoregressive rollouts on Navier-Stokes, which are unstable for LOCO/Hybrid models despite lower one-step error, pointing to an open design issue.

2 Method: LOCO

We consider a PDE of the form:

$$\frac{\partial u}{\partial t} = \mathcal{P}[u], \tag{1}$$

where $\mathcal{P}$ is an operator comprised of differential and k-linear operations. We examine one of the low order but common field interaction terms $u \cdot \frac{\partial u}{\partial x}$. Taking the discrete Fourier Transform $u(x) = \sum_k \hat{u}_k e^{ikx}$:

$$u\frac{\partial u}{\partial x} = u(x) \cdot \sum_k (ik)\hat{u}_k e^{ikx} = \sum_{j,k} \hat{u}_j \hat{u}_k (ik) e^{i(j+k)x} \tag{2}$$

Collecting terms with wavenumber $m = j + k$:

$$\widehat{u\frac{\partial u}{\partial x}}_m = \sum_k ik\hat{u}_k \hat{u}_{m-k} \tag{3}$$

We see that in Fourier Space, terms like this naturally decompose into nested convolutions and pointwise nonlinearities. These operations naturally form the whole computational graph, when following through with spectral numerical integration. Inspired by this we set the following design principles:

1. Differential operators become diagonal in Fourier Space → **Operate in Fourier Space**;
2. Temporal evolution depends only on the previous time step → **Skip connections**;
3. Decay of High Frequency Modes → **Mode truncation**;
4. Differentials become pointwise in Fourier Space → **Pointwise nonlinear operations**;
5. Field interactions manifest as spectral convolutions → **Spectral convolutions**.

We attribute FNO's success to these design choices. LOCO expands FNO in adding (5), the local field interactions. These operations nest recursively throughout the integration process. Remarkably, after just 10 iterations of a typical spectral solver, the accumulated computational graph contains over 100 spectral convolution operations. This is why we assert that insights need to be sufficiently abstracted before they are being used. We motivate our central hypothesis: *These design choices provide a natural basis for approximating the computational graph arising in numerical integration, even when vastly simplified.* Interestingly the Fourier Neural Operator is based on convolutions in real space. This is the opposite of the spectral method, proposed here, where convolutions mainly happen in frequency space.

Architecture. Figure 1 illustrates three architectures FNO, LOCO and Hybrid. All three models use linear lifting/projection layers and 2-layer Channel MLPs per block, but differ in spectral processing. We note that this differs from the neuraloperator library implementation which uses a 2 Layer ChannelMLP as a final projection layer [6]. We used a linear layer for better comparability as well as performance gains on FNO, Hybrid and LOCO.

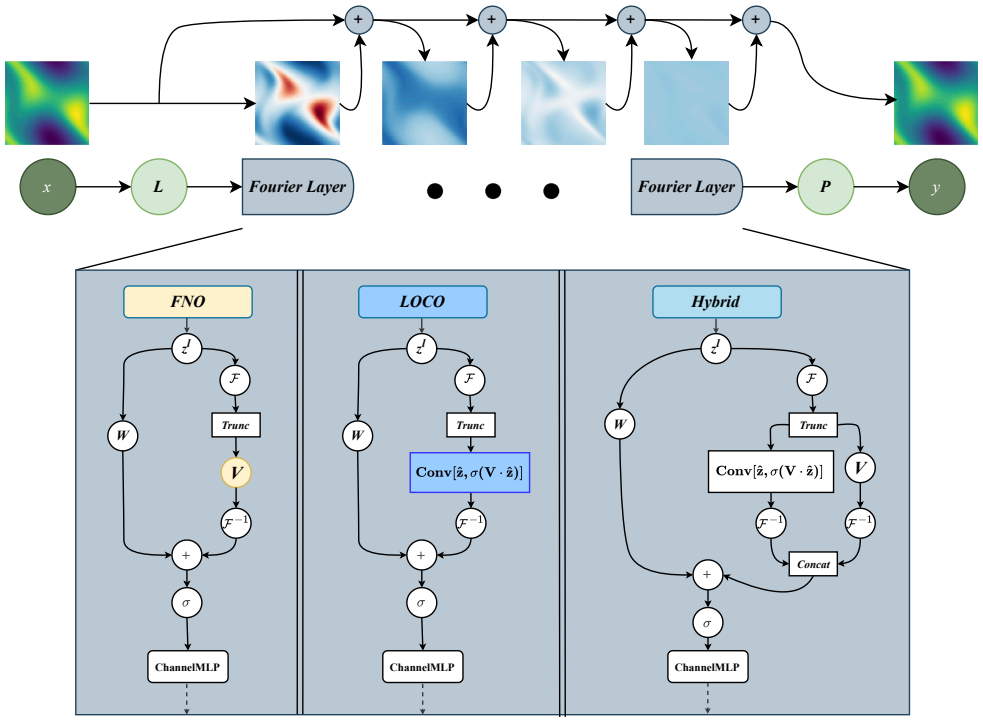
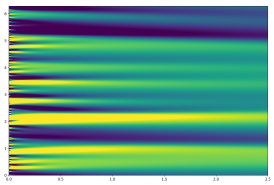
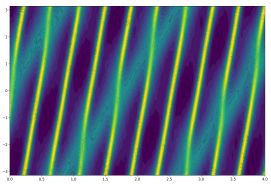
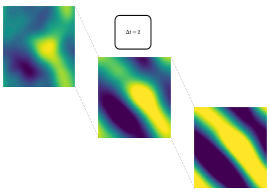


Figure 1: Architecture comparison. FNO: global spectral mixing via learnable V. LOCO: Local Spectral Convolutions with pointwise nonlinearity σ, learnable linear weight V. Hybrid: splits channels evenly between both blocks. We show the rollout of the model along a Navier-Stokes example using the linear projection layer to evaluate the contributions of each layer. One can see that contributions become smaller along the layer dimensions.

3 Experiments and Results

Experiments. We evaluate on three PDEs (Burgers 1D, KdV 1D, Navier-Stokes 2D), each using different numerical integration schemes for data generation (see Appendix A). Table 1 shows the equations and their spatiotemporal evolution. For Burgers and KdV we train on L2 Loss, while for Navier-Stokes we follow the FNO paper and use H1 Loss [8]. Please view Appendix B or our repository for implementation details (https://github.com/lenzpracher/eurips-loco).

Table 1: PDEs and spatiotemporal evolution patterns. Numerical integration schemes are described in Appendix A.

Burgers' Equation	**Korteweg-de Vries Equation**	**Navier-Stokes (2D)**
$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}$	$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} = 0$	$\frac{\partial \omega}{\partial t} + (\mathbf{u}\cdot\nabla)\omega = \nu\nabla^2\omega$
		

Results. Figure 2 shows training and validation loss curves. LOCO achieves significant uplift in convergence and validation loss for Burgers' equation. Improvements for KdV are more modest. In the Navier-Stokes equation, FNO demonstrates lower training loss, but higher validation loss than

LOCO. The Hybrid model beats both FNO and LOCO across all three PDEs. We note a higher instability in training for the LOCO model. Full metrics with parameter counts can be seen in Appendix B.
The Hybrid model outperforms FNO in autoregressive rollout in the Burgers and matching it in the KdV equation (see Appendix D).

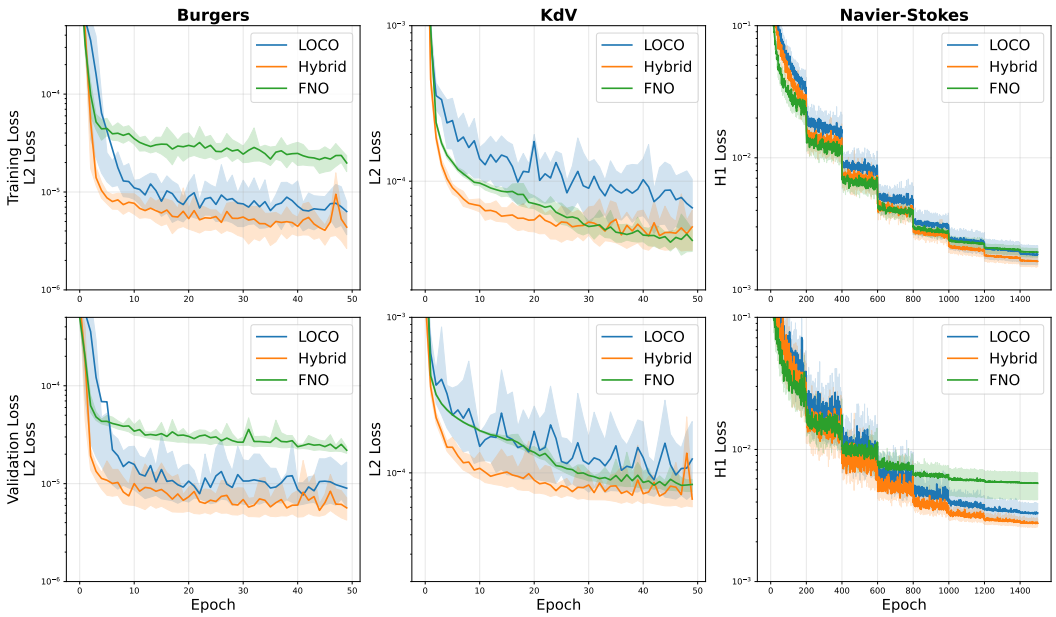


Figure 2: Training and validation losses for all three PDEs. Top row: KdV and Burgers (50 epochs, 20 runs); bottom row: 2D Navier–Stokes (1500 epochs, 5 runs). Solid curves show the mean across runs; shaded bands indicate the 10th–90th percentile across runs. LOCO and Hybrid demonstrate consistent improvements over the FNO baseline on single-step validation metrics.

Limitations. The Navier-Stokes experiments reveal a problematic limitation of the LOCO and Hybrid architecture: While the Hybrid model achieves lower initial validation loss, we note larger error accumulation during autoregressive rollouts than FNO (see Appendix D). We see the Hybrid and LOCO model seemingly converge to the same attractor state regardless of initial conditions, indicating architectural bias that steers predictions off-trajectory. We attempt to mitigate this using nonlinear projection layers, anti-aliasing, spatial convolutions, frequency dampening, learned mode gating—either, data augmentation via Gaussian noise or the pushforward trick [1]. While non linear projection layers and data augmentation using Gaussian noise yield similar rollout losses as FNO, the initial validation loss increases above the FNO threshold. The other methods fail to resolve the instability and degrade single-step performance to FNO baseline levels. We presume this is a solvable problem, which would yield a strong contender to the FNO baseline.

4 Conclusion

This work demonstrates how numerical integration principles can guide neural operator design from first principles. By analyzing computational patterns in spectral PDE solvers, we identified fundamental design choices that translate into neural architectures. The introduced architecture LOCO demonstrates performance gains against the FNO baseline equivalent in single step prediction. While autoregressive rollouts are a main concern for the applicability of the LOCO architecture, we argue that its simplicity along with its principled approach yields a building block for better prediction accuracy among domains in science and meteorology.

Acknowledgements

This study was partly supported by JST PRESTO (JPMJPR24TB), CREST (JPMJCR1914), and ASPIRE (JPMJAP2329).

References

[1] Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022.

[2] V. S. Fanaskov and I. V. Oseledets. Spectral neural operators. *Doklady Mathematics*, 108(S2):S226–S232, December 2023.

[3] Wenhan Gao, Ruichen Xu, Yuefan Deng, and Yi Liu. Discretization-invariance? on the discretization mismatch errors in neural operators. In *The Thirteenth International Conference on Learning Representations*, 2025.

[4] John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Efficient token mixing for transformers via adaptive fourier neural operators. In *International Conference on Learning Representations*, 2022.

[5] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, May 2021.

[6] Jean Kossaifi, Nikola Kovachki, Zongyi Li, David Pitt, Miguel Liu-Schiaffini, Valentin Duruisseaux, Robert Joseph George, Boris Bonev, Kamyar Azizzadenesheli, Julius Berner, and Anima Anandkumar. A library for learning neural operators. *arXiv preprint arXiv:2412.10354*, 2025.

[7] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[8] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

[9] Miguel Liu-Schiaffini, Julius Berner, Boris Bonev, Thorsten Kurth, Kamyar Azizzadenesheli, and Anima Anandkumar. Neural operators with localized integral and differential kernels. In *ICLR 2024 Workshop on AI4DifferentialEquations In Science*, 2024.

[10] Takashi Matsubara and Takaharu Yaguchi. Number theoretic accelerated learning of physics-informed neural networks. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence (AAAI-25)*, pages 595–603. Association for the Advancement of Artificial Intelligence, 2025.

[11] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators, 2022.

[12] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[13] Yusuke Tanaka, Takaharu Yaguchi, Tomoharu Iwata, and Naonori Ueda. Neural operators meet energy-based theory: Operator learning for hamiltonian and dissipative PDEs, 2024.

[14] Han Wan, Rui Zhang, and Hao Sun. Spectral-inspired neural operator for data-efficient PDE simulation in physics-agnostic regimes, 2025.

[15] Z Zlatev, R Berkowicz, and L.P Prahm. Stability restrictions on time-stepsize for numerical integration of first-order partial differential equations. *Journal of Computational Physics*, 51(1):1–27, 1983.

A Numerical Integration Methods

Please view our repository for implementation `LOCO`. For the Navier-Stokes implementation see reference [8].

Table 2: Overview of PDEs and Numerical Methods

Equation	**Method**	**Initial Conditions**
KdV: $u_t + 6uu_x + u_{xxx} = 0$	Strang Splitting + RK4	Two-soliton superposition with random amplitudes $c_i \in [0.5, 2.0]$
Burgers: $u_t + uu_x = \nu u_{xx}$	Semi-implicit Fourier	Random uniform noise $[-1, 1]$
2D Navier-Stokes [8]	Crank–Nicolson	Gaussian RF ($\alpha = 2.5$, $\tau = 7$)

B Experimental Setup

For the 2D Navier-Stokes experiments, we follow the experimental setup of [8]. For implementation please visit our repository `LOCO`

Table 3: Experimental Configuration

Parameter	**Burgers**	**KdV**	**Navier-Stokes**
Simulation Parameters			
Spatial Grid	$N = 512$	$N = 128$	256×256
Domain	$[0, 2\pi]$	$[0, 2\pi]$	$[0, 1]^2$
Time Step	$\Delta t = 10^{-4}$	$\Delta t = 10^{-4}$	$\Delta t = 10^{-4}$
Integration Time	$T = 20$	$T = 10$	$T = 100$
Save Interval	0.05 time units	0.01 time units	0.1 time units
Physical Parameters	$\nu = 0.01$	$a = 6$, $b = 1$	$\nu = 10^{-3}$
Training Setup			
Batch Size	32	32	8
Epochs	50	50	1500
Learning Rate	10^{-3}	10^{-3}	10^{-3}
Optimizer	AdamW	AdamW	AdamW
Weight Decay	10^{-4}	10^{-4}	10^{-4}
LR Schedule	-	-	StepLR ($\gamma = 0.5$, step=200)
Loss Function	MSE	MSE	H1 Loss
Independent Runs	20	20	5
Model Architecture			
Fourier Modes	16	16	12×12
Hidden Channels	24	26	32
Number of Layers	4	4	4
Input Channels	1	1	10
Output Channels	1	1	1
Data Configuration			
Training Resolution	512	128	64×64
Training Samples	1,000	1,000	1,000
Test Samples	200	200	200
Prediction Gap	10	10	1
Input Sequence Length	1	1	10

C Numerical Results

Table 4: Training / Validation Loss Results

Model	Burgers [10^{-5}]	KdV [10^{-5}]	Navier-Stokes [10^{-3}]
LOCO	0.632 / 0.899	6.74 / 12.3	1.86 / 3.30
Hybrid	**0.438 / 0.566**	5.08 / **6.78**	**1.65 / 2.76**
FNO	1.98 / 2.20	**4.16** / 8.42	1.93 / 5.55

D Spacetime Rollout

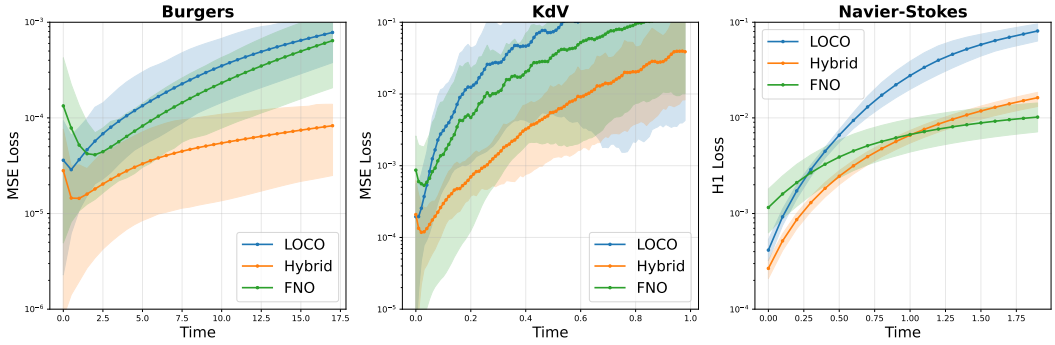


Figure 3: Autoregressive rollout analysis across all three PDEs. We plot MSE vs. 35 rollout steps for Burgers and 100 rollout steps KdV, and H1 loss vs. 20 steps for 2D Navier–Stokes. Solid curves show the mean across runs; shaded bands denote the 10th–90th percentile across runs. In Navier–Stokes, despite lower initial errors, the Hybrid model shows degraded long-term stability compared to FNO, with crossover at steps 4–5.

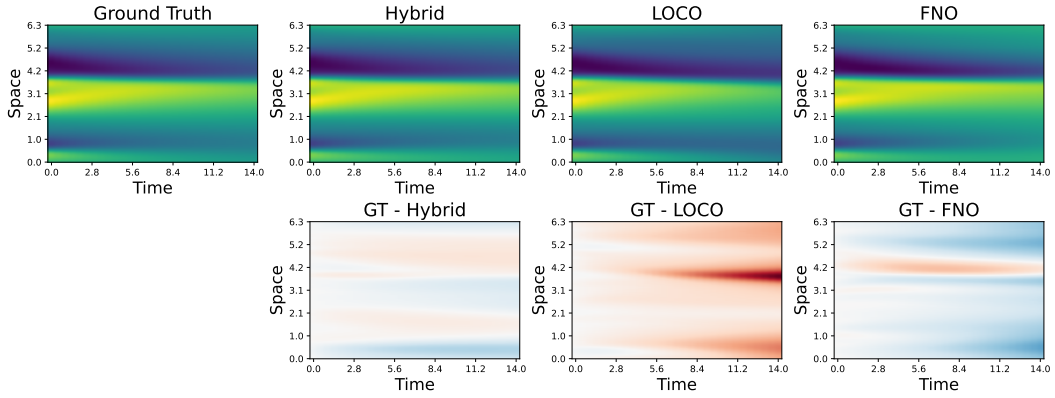


Figure 4: Spacetime rollout comparison for the 1D Burgers equation over 35 time steps. Top row: ground truth, Hybrid, LOCO, and FNO predictions. Bottom row: difference maps (GT − prediction) highlighting model-specific error patterns.

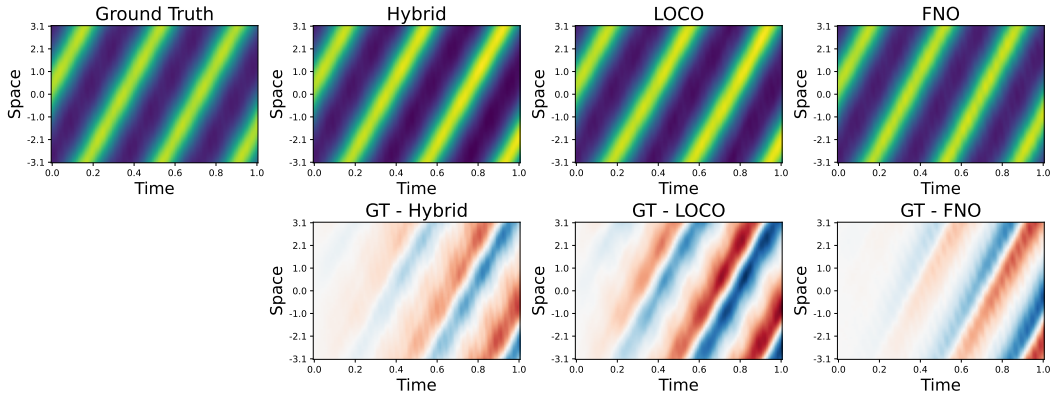


Figure 5: Spacetime rollout comparison for the Korteweg–de Vries (KdV) equation over 100 time steps. Top row: ground truth, Hybrid, LOCO, and FNO predictions. Bottom row: difference maps (GT − prediction) highlighting model-specific error patterns.