PDE-JEPA: PREDICTIVE REPRESENTATION LEARNING OF LATENT DYNAMICS MODELING FOR PARAMETRIC PDES

Anonymous authors
Paper under double-blind review

ABSTRACT

Physical trajectories contain more than snapshots of a system: they also reveal how its states evolve under governing conditions. However, representation learning for parametric partial differential equations (PDEs) has largely relied on reconstruction-based objectives that emphasize recovering observed physical fields. In this paper, we investigate predictive representation pretraining as an alternative to reconstruction-based learning. We find that predictive representations preserve rich physical information, yet this advantage alone does not ensure accurate field evolution. Based on these observations, we introduce PDE-JEPA for parametric PDE dynamics. Specifically, we first train an encoder using a masked-latent prediction to capture the underlying regularities of PDE dynamics. To explicitly adapt the pretrained representation toward a more dynamics-aligned state space, we then introduce a geometry projector that aligns latent trajectory geometry with the evolution geometry of physical fields. Finally, building on this geometry-aligned latent space, we further develop a physics-structured latent predictor that decomposes the dynamics into parameter-independent evolution and parameter-dependent response components. Extensive experiments on nine widely used PDE benchmarks demonstrate that our framework outperforms existing state-of-the-art methods by an average of 33.4% in-distribution, while achieving an average improvement of 51.4% when extrapolating to unseen governing parameters. Our implementation will be released.

1 INTRODUCTION

Real-world systems exhibit complex dynamics across diverse physical processes (Cross & Hohenberg, 1993), which are mainly governed by partial differential equations (PDEs) (Evans, 2022). Solving these equations typically relies on classical numerical methods (LeVeque, 2007), which can achieve high accuracy but often incur substantial computational cost. This computational bottleneck has motivated learning-based PDE solvers that approximate physical evolution directly from data (Li et al., 2020; Karniadakis et al., 2021). However, practical applications (Wang et al., 2024; Tan et al., 2026a) often involve variations in coefficients, forcing terms, and boundary conditions across physical environments. Parametric PDEs describe the resulting families of related dynamics, requiring models to capture temporal evolution while generalizing to unseen governing conditions.

Existing approaches to parametric PDEs have largely focused on improving models through parameter conditioning (Takamoto et al., 2023; Zhou et al., 2024; Bischof et al., 2026), adaptation across environments (Kassaï Koupaï et al., 2024; Yang & Ren, 2026), in-context learning (Kassaï Koupaï et al., 2026; Patel et al., 2026), or fine-tuning pretrained foundation models (Hao et al., 2024; Herde et al., 2024; McCabe et al., 2025). In latent dynamics models, the latent state representations are commonly learned through reconstruction-based objectives (Serrano et al., 2024b; Wang & Wang, 2024; Hagnberger et al., 2026), sometimes augmented with dynamics-aware losses (Wu et al., 2022; Li et al., 2025). However, reconstruction fidelity alone does not establish whether physically relevant information is readily accessible (Qu et al., 2026). Moreover, it does not guarantee that the learned latent space is well suited for accurate temporal evolution (Brettin et al., 2025), particularly under shifts in governing conditions. This raises a central question: *what makes a latent representation suitable for accurate prediction and extrapolation of parametric PDE dynamics?*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

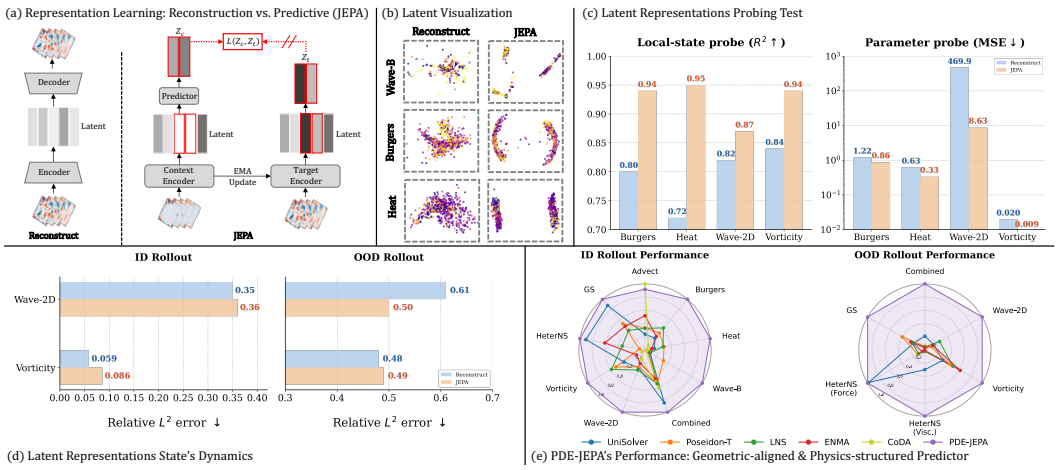


Figure 1: **Overview of our key observations.** Predictive representation (JEPA) learns physically informative representations, but informativeness alone does not ensure effective forecasting of parametric PDEs. Our approach, PDE-JEPA, improves PDE forecasting and extrapolation through geometry alignment and physics-structured latent prediction.

We explore this question through JEPA-based predictive representation learning. As illustrated in Figure 1(a), we predict masked latent targets rather than reconstructing observations (Assran et al., 2023). While such objectives have achieved considerable success in world modeling (Terver et al., 2025; Mur-Labadia et al., 2026; Klindt et al., 2026; Maes et al., 2026; Yan et al., 2026) and demonstrated early promise in parameter probing (Qu et al., 2026), their utility for parametric PDE dynamics remains largely unexplored. We therefore conduct an in-depth dissection of reconstruction-based and predictive representations through frozen-encoder probes and autoregressive rollout evaluation. Figure 1(b–d) shows the results, with additional analyses in Appendix D.1. Our analysis yields two key observations.

Observation 1: Predictive learning yields more informative representations of physical dynamics. We first examine how reconstruction-based and predictive learning organize physical information in latent space. Figure 1(b) suggests clearer parameter-dependent organization in JEPA features than in reconstruction-based features. For example, in the Burgers dataset, the JEPA representation exhibits two branches associated with different parameter ranges. To assess the physical relevance of these features, we evaluate two complementary probes on frozen representations, as shown in Figure 1(c). A local-state probe measures how accurately instantaneous physical fields can be recovered from the features. A parameter probe evaluates how well the governing conditions can be inferred from them. Across the evaluated PDEs, JEPA achieves stronger performance on both probing tasks. These results demonstrate that predicting dynamics in latent space encourages the encoder to retain both fine-grained state information and global governing factors that drive physical evolution.

Observation 2: Informative representations are not necessarily easy to evolve. Physical probing evaluates what can be recovered from representations of observed states. Autoregressive forecasting poses a different challenge: repeatedly evolving predicted states without access to future observations. Figure 1(d) reveals a mismatch between these two capabilities. Despite its stronger probing performance, the vanilla JEPA-based model produces higher ID rollout errors than the reconstruction-based baseline on both Wave-2D and Vorticity. Under parameter shifts, JEPA achieves lower OOD error on Wave-2D and comparable performance on Vorticity. These results suggest that predictive pretraining provides a physically informative starting point, but its representational advantages alone do not ensure accurate recursive evolution.

Taking these observations together, we define the central challenge as preserving the physical information captured during pretraining (**observation 1**) while adapting the latent organization to the demands of long-horizon evolution and parameter generalization (**observation 2**). In this paper, we introduce **PDE-JEPA**, a framework for learning evolvable state spaces for parametric PDEs.

Specifically, we first view predictive pretraining as a foundation to build upon rather than a complete solution to latent dynamics modeling. Based on pretrained representations, we then introduce the Physics-Aligned Latent Geometry (**PAG**) module, a lightweight residual geometry projector that aligns latent trajectory geometry with the evolution geometry of physical fields to enhance latent evolution while preserving the information encoded by the original representation via an anchor loss. Finally, to further improve generalization to unseen governing conditions, we develop the Physics-Structured Latent Predictor (**PSP**) that decomposes the dynamics into parameter-independent evolution and parameter-dependent responses. This structure reflects the common form of parametric PDEs, in which governing parameters modulate specific dynamical components, and provides an explicit inductive bias for extrapolation under unseen governing conditions.

We evaluate PDE-JEPA on 9 parametric PDE benchmarks spanning transport, diffusion, reaction–diffusion, wave propagation, and fluid dynamics. Our method achieves the lowest rollout error across most in-distribution benchmarks and all five benchmarks evaluated under out-of-distribution governing conditions. For example, as shown in Figure 1 (e), PDE-JEPA achieves the most improvement on Heat against Poseidon-T for the ID setting. Further analyses show that these gains are accompanied by more physically aligned latent trajectories and more consistent responses to changes in governing parameters. To sum up, our contributions are listed below:

- To our knowledge, we are the first to systematically study JEPA for parametric PDEs and show that informative representations alone do not ensure accurate rollout. We thus propose **PDE-JEPA**, combining **PAG** and **PSP** for accurate forecasting and OOD extrapolation.
- We propose **PAG** module, a lightweight geometry projector that aligns latent trajectory geometry with physical evolution to substantially improve rollout accuracy, while preserving the pretrained representation through an anchor loss.
- We further introduce the **PSP**, which incorporates PDE formulation inductive bias by separating parameter-independent evolution from parameter-dependent responses, improving OOD extrapolation.

2 Related Work

2.1 Parametric PDE Solvers

Learning-based PDE solvers broadly span physics-informed methods (Karniadakis et al., 2021; Toscano et al., 2025) and data-driven neural operators (Lu et al., 2021; Wu et al., 2024; Tan et al., 2026b). We focus on the latter for parametric forecasting, with the Fourier Neural Operator (FNO) (Li et al., 2020) providing a foundational framework. Parametric generalization has since been explored through explicit parameter conditioning (Brandstetter et al., 2022; Takamoto et al., 2023; Cho et al., 2024; Berman & Peherstorfer, 2024; Hagnberger et al., 2024), multi-environment adaptation (Yin et al., 2021; Kirchmeyer et al., 2022; Huang et al., 2022; Kassaï Koupaï et al., 2024) and in-context operator learning (Yang et al., 2023; Yang & Osher, 2024; Serrano et al., 2024a; Kassaï Koupaï et al., 2026; Patel et al., 2026). More recently, general-purpose PDE solvers leverage large-scale multi-physics pretraining (McCabe et al., 2024; Hao et al., 2024; Herde et al., 2024; Zhou et al., 2024; McCabe et al., 2025; Wang et al., 2026a; Wu et al., 2026). Complementary work further explores operator decomposition (Gopakumar et al., 2026) and joint parameter–boundary conditioning (Li et al., 2026). Parametric PDE solvers are not the scope of this paper, we approach parametric forecasting through latent-state dynamics, asking how the learned state space should be structured for evolution and extrapolation across governing conditions.

2.2 Latent Learning for PDE Dynamics

Latent PDE models represent physical states in learned latent spaces and model temporal evolution directly in representation space (Benner et al., 2015; Wiewel et al., 2019; Maulik et al., 2021; Han et al., 2022). A central design choice in this paradigm is how the latent state itself is learned. Many existing approaches obtain latent representations through reconstruction objectives, where the latent variables are optimized to recover the observed physical fields (Chen et al., 2022; Wu et al., 2022; Li et al., 2025). Furthermore, reconstruction objectives are often combined with additional losses that encourage latent dynamical predictability (Regazzoni et al., 2024). More recent approaches ex-

tend latent-space forecasting through autoregressive modeling over quantized or continuous latent representations (Serrano et al., 2024a; Kassaï Koupaï et al., 2026), but the learned state space is still reconstruction oriented. In contrast, our encoder is pretrained through masked latent prediction rather than physical-field reconstruction, allowing the state representation to be learned from predictable spatiotemporal structure in representation space. We then explicitly align the latent trajectory geometry for downstream dynamics modeling.

2.3 Predictive Learning for Physical Systems

Predictive representation learning provides an alternative to reconstruction-based self-supervision by learning features from predictable structure rather than directly recovering observations. Joint-embedding predictive architectures (JEPAs) instantiate this idea through latent prediction, beginning with I-JEPA (Assran et al., 2023) and later extending to video with V-JEPA (Bardes et al., 2024; Mur-Labadia et al., 2026). Related physical self-supervision has explored Lie-symmetry-based invariance (Mialon et al., 2023) and masked reconstruction (Zhou & Farimani, 2024). More recently, predictive representations have been shown to encode governing physical factors more effectively (Qu et al., 2026) and have been extended to 3D aerodynamic fields with AeroJEPA (Giral et al., 2026). Separately, latent trajectory geometry has been explicitly shaped to facilitate downstream dynamics (Wang et al., 2026b). Our work studies whether predictive representations can serve as informative latent states for parametric dynamics, and how their geometry should be adapted for long-horizon evolution and extrapolation.

3 Preliminaries and Problem Setup

3.1 Parametric PDE Dynamics

We consider time-dependent physical systems governed by a family of PDEs,

$$\frac{\partial \mathbf{u}}{\partial t} = \mathcal{F}\left(t, \mathbf{x}, \mathbf{u}, \nabla \mathbf{u}, \nabla^2 \mathbf{u}, \ldots; \boldsymbol{\xi}\right), \qquad \mathbf{x} \in \Omega,\ t \in (0, T], \tag{1}$$

subject to

$$\mathcal{B}_{\boldsymbol{\xi}}[\mathbf{u}](t, \mathbf{x}) = 0, \qquad \mathbf{x} \in \partial\Omega, \qquad \mathbf{u}(0, \mathbf{x}) = \mathbf{u}_0(\mathbf{x}). \tag{2}$$

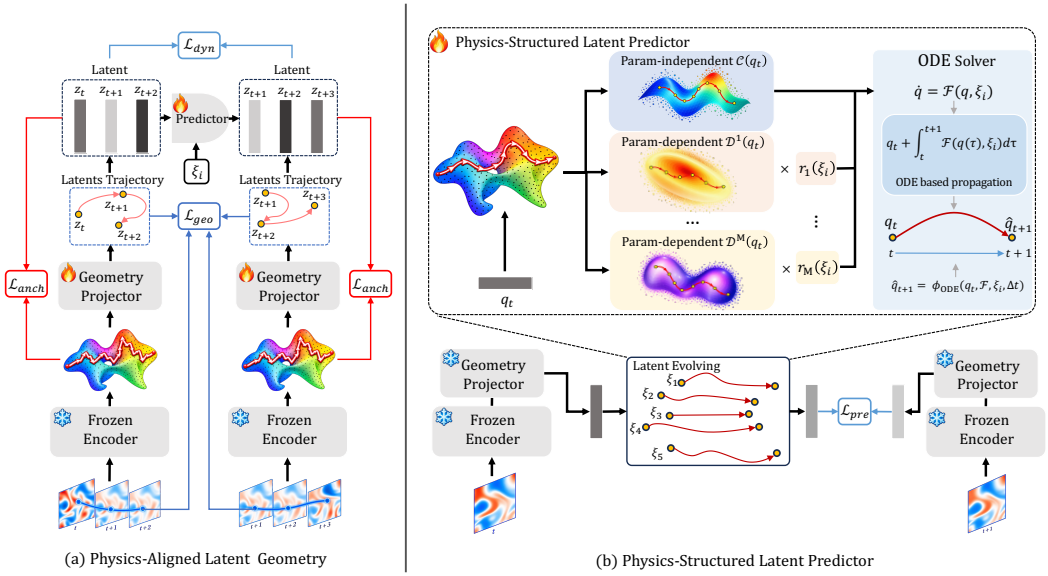
Here, $\mathbf{u}(t, \mathbf{x}) \in \mathbb{R}^C$ denotes the physical fields on the domain Ω with C channels and $\mathcal{B}$ denotes a boundary operator. $\boldsymbol{\xi}$ characterizes the governing environment, including PDE coefficients, forcing terms, and boundary conditions. A fixed $\boldsymbol{\xi}$ therefore defines a particular dynamical system, while varying $\boldsymbol{\xi}$ induces a family of related but distinct physical evolutions.

3.2 Joint-Embedding Predictive Learning

Joint-Embedding Predictive Architectures (JEPAs) learn representations by predicting missing content in a learned latent space rather than reconstructing the observation itself (LeCun et al., 2022; Assran et al., 2023; Bardes et al., 2024). Let $\mathcal{C}$ and $\mathcal{M}$ denote the index sets of visible context tokens and masked target tokens, respectively. Given the visible context $\mathbf{x}_{\mathcal{C}}$ and target mask tokens $\mathbf{m}$, the context encoder E_θ produces context representations, and the predictor P_ϕ predicts the target representations at positions $i \in \mathcal{M}$ provided by an EMA target encoder $\bar{E}_\theta$.

$$\mathcal{L}_{\text{JEPA}} = \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \left\| P_\phi(E_\theta(\mathbf{x}_{\mathcal{C}}), \mathbf{m})_i - \text{sg}\big(\bar{E}_\theta(\mathbf{x})_i\big) \right\|_1. \tag{3}$$

By predicting directly in representation space, JEPA encourages latent features to capture predictable spatiotemporal structure. V-JEPAs demonstrate strong motion understanding and temporally consistent representations (Mur-Labadia et al., 2026), while recent studies on physical systems show that JEPA features encode governing physical factors effectively (Qu et al., 2026). These results motivate us to investigate JEPA as a state representation for parametric physical dynamics.



(a) Physics-Aligned Latent Geometry (b) Physics-Structured Latent Predictor

Figure 2: **Overview of PDE-JEPA.** (a) We first adapt the frozen predictive representation initialized by JEPA with a light weight geometry projector that aligns latent trajectory geometry with physical evolution. (b) We then evolve the aligned latent states using a physics-structured latent predictor, which decomposes the dynamics into parameter-independent and parameter-dependent components and integrates their combined dynamics with an ODE solver.

4 Method

4.1 Predictive Physical States

We adopt the pretrained JEPA representation as the base latent space. Given a physical trajectory $\mathbf{u}_{0:T} = \{\mathbf{u}_t\}_{t=0}^{T}$, we encode the full state trajectory as

$$\mathbf{z}_{0:T} = E(\mathbf{u}_{0:T}), \qquad \mathbf{u}_{0:T} \in \mathbb{R}^{|\mathcal{X}|\times T\times C}, \quad \mathbf{z}_{0:T} \in \mathbb{R}^{N\times T\times D}, \tag{4}$$

where $|\mathcal{X}|$ denotes the grid number and C the number of physical-field channels. The encoder maps $\mathbf{u}_{0:t}$ to NT spatial latent tokens $\mathbf{z}_{0:T}$, each of dimension D. Although JEPA is pretrained on full trajectory, downstream states, such as those used for predictor training, are encoded frame-wise following (Mur-Labadia et al., 2026), ensuring that $\mathbf{z}_t$ contains no information from future observations.

Importantly, the parameter $\boldsymbol{\xi}$ is provided to neither the encoder nor the JEPA predictor during pretraining, encouraging the encoder to learn parameter-agnostic physical representations. We keep E frozen throughout the subsequent geometry-alignment and dynamics-learning stages, and use the resulting latent trajectory $\{\mathbf{z}_t\}_{t=0}^{T}$ as the input to the geometry projector.

4.2 Physics-Aligned Latent Geometry

Although predictive pretraining yields a physically informative state space, its geometry is not explicitly optimized for temporal evolution. We observe a clear mismatch between physical and latent trajectories: latent states exhibit larger turning angles. For example, the mean turning angle increases from 31.3° to 60.5° on Vorticity and from 18.2° to 59.4° on Burgers. Such excessive turning produces more zig-zag latent trajectories, potentially complicating rollout propagation. We therefore align latent trajectory geometry with physical evolution while preserving the information already encoded by JEPA. The overview is presented in Figure 2 (a).

Residual geometry projection. Given latents $\mathbf{z}_{0:T}$, we introduce a token-wise projector

$$\mathbf{q}_t = \mathbf{z}_t + G_\phi(\mathbf{z}_t), \qquad G_\phi(\mathbf{z}) = W_2\sigma(W_1\mathrm{LN}(\mathbf{z}))\,, \tag{5}$$

where σ is GELU, LN is layernorm, and W_2 is zero-initialized so that $\mathbf{q}_t = \mathbf{z}_t$ initially. It therefore acts as a lightweight coordinate correction rather than relearning the state representation.

Physical trajectory alignment. We align latent trajectory geometry with evolution measured in physical-field space. Define the normalized temporal directions

$$\mathbf{d}_t^u = \frac{\mathbf{u}_{t+1} - \mathbf{u}_t}{\|\mathbf{u}_{t+1} - \mathbf{u}_t\|_2 + \epsilon}, \qquad \mathbf{d}_t^q = \frac{\mathbf{q}_{t+1} - \mathbf{q}_t}{\|\mathbf{q}_{t+1} - \mathbf{q}_t\|_2 + \epsilon}. \tag{6}$$

For temporal lag ℓ, we measure trajectory turning by similarity between two directions

$$s_{t,\ell}^u = \langle \mathbf{d}_t^u, \mathbf{d}_{t+\ell}^u \rangle, \qquad s_{t,\ell}^q = \langle \mathbf{d}_t^q, \mathbf{d}_{t+\ell}^q \rangle, \tag{7}$$

and minimize

$$\mathcal{L}_{\text{geo}} = \sum_{\ell \in \{1,2,4\}} w_\ell \, \text{SmoothL1}\left(s_{t,\ell}^q, \text{sg}(s_{t,\ell}^u)\right). \tag{8}$$

The multi-lag objective captures local and longer-range directional consistency.

Preserving informativeness. To prevent geometric alignment from distorting the pretrained state information, we use an identity anchor

$$\mathcal{L}_{\text{anchor}} = \frac{\|\mathbf{q} - \mathbf{z}\|_2^2}{\|\mathbf{z}\|_2^2 + \epsilon}. \tag{9}$$

We further train an auxiliary conditional causal predictor to estimate $\Delta\mathbf{q}_t = \mathbf{q}_{t+1} - \mathbf{q}_t$,

$$\mathcal{L}_{dyn} = \frac{\|\widehat{\Delta\mathbf{q}}_t - \Delta\mathbf{q}_t\|_2^2}{\|\Delta\mathbf{q}_t\|_2^2 + \epsilon}, \tag{10}$$

so that the aligned coordinates remain dynamically predictable. The geometry stage optimizes

$$\mathcal{L}_{\text{align}} = \mathcal{L}_{dyn} + \lambda_{\text{geo}}\mathcal{L}_{\text{geo}} + \lambda_{\text{anchor}}\mathcal{L}_{\text{anchor}}. \tag{11}$$

The JEPA encoder remains frozen. After alignment, we discard the auxiliary predictor and freeze the projector before training the final physics-structured dynamics model.

4.3 Physics-Structured Latent Predictor

The geometry-aligned representation supports temporal evolution, but OOD generalization still requires extrapolating across governing parameters. A standard conditional predictor can inject the parameter directly to the networks, allowing arbitrary state–parameter interactions that may fit the training range well but extrapolate unreliably beyond it. We instead explicitly structure how the governing parameter enters the latent dynamics with formulation inductive bias.

Structured latent vector field. We model $\mathbf{q}(t)$ as a continuous-time parametric dynamical system,

$$\frac{d\mathbf{q}}{dt} = F_\theta(\mathbf{q}; \boldsymbol{\xi}), \tag{12}$$

rather than allowing $\boldsymbol{\xi}$ to interact arbitrarily with the latent state throughout the dynamics network, we separate a shared state evolution from a set of parameter-dependent responses:

$$\frac{d\mathbf{q}}{dt} = C_\theta(\mathbf{q}) + \sum_{j=1}^{M} r_j(\boldsymbol{\xi}) \, D_\theta^{(j)}(\mathbf{q}). \tag{13}$$

Here, C_θ captures shared evolution, while $D_\theta^{(j)}$ represents a state-dependent response associated with the j-th parameter. $r_j(\boldsymbol{\xi})$ are normalized physical parameters and M is the number of parameters. This decomposition reflects a common structure in parametric PDEs, where governing coefficients modulate specific components. For example, the Navier–Stokes equation in vorticity form is

$$\frac{\partial \omega}{\partial t} = -(\mathbf{u} \cdot \nabla)\omega + \nu \nabla^2 \omega, \tag{14}$$

where ν explicitly scales the viscous response. Motivated by this structure, the Eq. 13 becomes

$$\frac{d\mathbf{q}}{dt} = C_\theta(\mathbf{q}) + r(\nu) D_\theta(\mathbf{q}). \tag{15}$$

We do not require either response to recover the exact analytical PDE operators; the decomposition provides a physics-inspired inductive bias on how governing parameters enter the latent dynamics.

Table 1: **ID rollout performance across PDE benchmarks.** All results are reported in Relative L^2 error, lower is better. Best results are **bold** and second-best results are underlined.

Method	Advect	Burgers	Heat	Wave-B	Combined	Wave-2D	Vorticity	HeterNS	GS
Parametric Solvers									
FNO	0.0390	0.4972	0.3449	0.9819	0.0374	0.9913	0.1411	0.0210	0.0547
CAPE	0.0094	0.2230	0.2130	0.9780	0.0085	–	–	–	–
CoDA	**0.0068**	0.5460	0.7670	1.0200	0.0120	0.7770	0.6780	–	–
GEPS	0.0930	0.3989	0.6641	0.6317	0.0097	0.5138	0.0821	0.1032	0.0332
In-Context Solvers									
ViT-*in-context*	0.0902	0.4720	0.5820	0.4720	0.0885	0.3900	0.1730	–	0.0690
[CLS] ViT	0.1400	0.1360	0.1160	0.9710	0.0446	0.2710	0.9720	–	0.0480
Zebra	0.0079	0.1540	0.1150	0.2450	0.0096	0.2070	0.1190	–	0.0440
Foundation Models									
UniSolver	0.0284	0.1838	0.1933	0.4049	0.0087	0.4009	0.0954	0.0098	0.0323
MPP	0.0312	0.4547	0.5629	0.3856	0.0519	0.8815	0.4509	0.0347	0.2698
DPOT-S	0.0390	0.6480	0.4638	0.4063	0.0365	0.3152	0.1686	0.0896	0.4159
Poseidon-T	0.0203	0.1280	0.0933	0.1093	0.0137	0.4211	0.0679	0.1009	0.0544
Latent Solvers									
LE-PDE	0.0212	0.0869	0.1053	0.6927	0.0299	0.5140	0.5816	0.2380	0.1225
LNS	0.0207	0.0982	0.0997	0.4002	0.0136	0.3539	0.0592	0.0254	0.0734
MAE-PDE	0.1943	0.4467	0.3726	0.5285	0.0667	0.7792	0.1327	0.1798	0.0406
ENMA	0.0131	0.1607	0.2712	0.6295	0.0160	0.5285	0.2320	0.0144	0.0607
Ours	0.0074	**0.0428**	**0.0274**	**0.0350**	**0.0074**	**0.1140**	**0.0348**	**0.0089**	**0.0284**
Rel. Impr.	−8.8%	50.7%	70.6%	68.0%	12.9%	44.9%	41.2%	9.2%	12.1%

Continuous-time propagation. We propagate latent between observations using an ODE solver,

$$\widehat{\mathbf{q}}_{t+1} = \Phi_{\mathrm{ODE}}\left(\mathbf{q}_t, F_\theta, \boldsymbol{\xi}, \Delta t\right), \tag{16}$$

where Φ_{ODE} integrates the learned vector field over one observation interval. In practice, we use a fixed-step fourth-order Runge–Kutta (RK4) solver. Long-horizon predictions are obtained by recursively integrating the predicted states. We then supervise it with predictor objective

$$\mathcal{L}_{\mathrm{pre}} = \frac{\|\widehat{\mathbf{q}}_{t+1} - \mathbf{q}_{t+1}\|_2^2}{\|\Delta\mathbf{q}_t\|_2^2 + \epsilon}. \tag{17}$$

5 Experiments

5.1 Experiments Setup

Datasets. We evaluate on nine parametric PDE benchmarks spanning transport, diffusion, waves, reaction–diffusion, and fluid dynamics. Seven follow Zebra (Serrano et al., 2024a): Advection varies the transport speed, Burgers and Heat vary diffusion and forcing, Wave-B varies boundary conditions, Combined varies three differential coefficients, Wave-2D varies wave celerity and damping, and Vorticity varies viscosity. We further include HeterNS from UniSolver (Zhou et al., 2024) and Gray–Scott (GS) from ENMA (Kassaï Koupaï et al., 2026). Detailed equations, parameter ranges, and data splits and generation are provided in Appendix A.

Baselines. We compare against a broad set of PDE solvers covering four representative paradigms. *Parametric solvers* include FNO (Li et al., 2020), CAPE (Takamoto et al., 2023), CoDA (Kirchmeyer et al., 2022), and GEPS (Kassaï Koupaï et al., 2024); *in-context solvers* include ViT-*in-context*, [CLS] ViT (Peebles & Xie, 2023), and Zebra (Serrano et al., 2024a); *foundation models* include UniSolver (Zhou et al., 2024), MPP (McCabe et al., 2024), DPOT-S (Hao et al., 2024), and Poseidon-T (Herde et al., 2024); and *latent solvers* include LE-PDE (Wu et al., 2022), LNS (Li et al., 2025),

Table 2: **Module ablation on rollout.**

	Vorticity		Wave-2D	
Model	ID	OOD	ID	OOD
Vanilla	.086	.491	.363	.502
+ PAG	<u>.040</u>	<u>.397</u>	<u>.143</u>	<u>.321</u>
+ PSP	**.034**	**.288**	**.114**	**.157**

Table 3: **Representation geometry before and after alignment.**

Dataset	Angle MAE (°)	Sym. Acc. MAE	Lag-1 Cos. MAE
Vorticity	$29.1 \rightarrow \mathbf{6.3}$ (-78%)	$.46 \rightarrow \mathbf{.10}$ (-77%)	$.36 \rightarrow \mathbf{.05}$ (-84%)
Wave-2D	$35.5 \rightarrow \mathbf{10.1}$ (-71%)	$.46 \rightarrow \mathbf{.14}$ (-68%)	$.50 \rightarrow \mathbf{.14}$ (-74%)
Burgers	$41.4 \rightarrow \mathbf{20.8}$ (-49%)	$.73 \rightarrow \mathbf{.37}$ (-49%)	$.43 \rightarrow \mathbf{.17}$ (-59%)
GS	$28.8 \rightarrow \mathbf{20.4}$ (-29%)	$.39 \rightarrow \mathbf{.27}$ (-30%)	$.39 \rightarrow \mathbf{.26}$ (-32%)

MAE-PDE (Zhou & Farimani, 2024), and ENMA (Kassaï Koupaï et al., 2026). These baselines span direct operator learning, parameter-conditioned adaptation, in-context prediction, large-scale PDE pretraining, and latent-space dynamics modeling.

Metrics. We evaluate accuracy using the relative L^2 error over the full rollout trajectory,

$$\mathcal{E}_{\mathrm{rel}} = \frac{1}{N_{\mathrm{test}}} \sum_{j=1}^{N_{\mathrm{test}}} \frac{\|\hat{\mathbf{u}}_j^{1:T} - \mathbf{u}_j^{1:T}\|_2}{\|\mathbf{u}_j^{1:T}\|_2}, \tag{18}$$

where $\hat{\mathbf{u}}^{1:T}$ and $\mathbf{u}^{1:T}$ denote the predicted and ground-truth trajectories, respectively. Lower values indicate better long-horizon forecasting accuracy.

5.2 In-Distribution Generalization

Table 1 reports in-distribution rollout performance. Our method achieves the lowest relative L^2 on eight of nine benchmarks and ranks second on Advection. The gains are particularly large on Burgers, Heat, Wave-B, Wave-2D, and Vorticity. Although the strongest baseline varies across systems—from LE-PDE and Poseidon-T to Zebra and LNS—our method remains consistently strong, suggesting that its benefits extend across diverse dynamical regimes. Advection is the only case where our method is not best (0.0074 vs. 0.0068 for CoDA). We attribute this small gap partly to the translation-dominated dynamics, where structures mainly move across the domain rather than change their shape, thus reconstruction ability matters most. Indeed, several methods already achieve errors below 10^{-2}, indicating a near-saturated regime. In contrast, the larger gains on other non-linear systems suggest that our approach becomes more effective as the underlying dynamics grow more complex.

5.3 Out-of-Distribution Extrapolation

We next evaluate extrapolation to governing conditions outside the training distribution. Due to space constraints, Table 4 reports a representative subset of baselines, with the full comparison provided in the Appendix D.3.1. Our method achieves the lowest error on all five benchmarks. The gains are particularly large on Combined, Wave-2D, and GS, improving over the strongest baselines by 77.9%, 74.2%, and 59.7%, respectively. Moreover, the ID-to-OOD degradation remains limited on these systems, increasing only from 0.0074 to 0.0084, 0.1140 to 0.157, and 0.0284 to 0.0337, respectively, indicating strong extrapolation beyond the training regimes.

Table 4: **OOD rollout performance across PDE benchmarks.** Relative L^2 error is reported, lower is better. Best results are **bold** and second-best results are <u>underlined</u>.

Method	Combined	Wave-2D	Vorticity	HeterNS Visc./Force	GS
UniSolver	<u>0.038</u>	1.003	0.923	<u>0.037</u>/<u>0.105</u>	0.1636
Poseidon-T	0.146	1.511	0.665	0.560/0.821	<u>0.083</u>
LNS	0.1667	<u>0.610</u>	0.481	0.610/0.932	0.146
ENMA	0.243	1.151	0.467	1.501/2.271	0.134
Zebra	–	0.680	<u>0.320</u>	–	–
Ours	**0.008**	**0.157**	**0.288**	**0.011/0.103**	**0.033**
Rel. Impr.	77.9%	74.2%	9.7%	69.5%/1.5%	59.7%

5.4 Effect of PAG

We first examine whether PAG facilitates long-horizon evolution. As shown in Table 2, adding the PAG reduces Vorticity error from 0.086 to 0.040 on ID data and from 0.491 to 0.397 on OOD

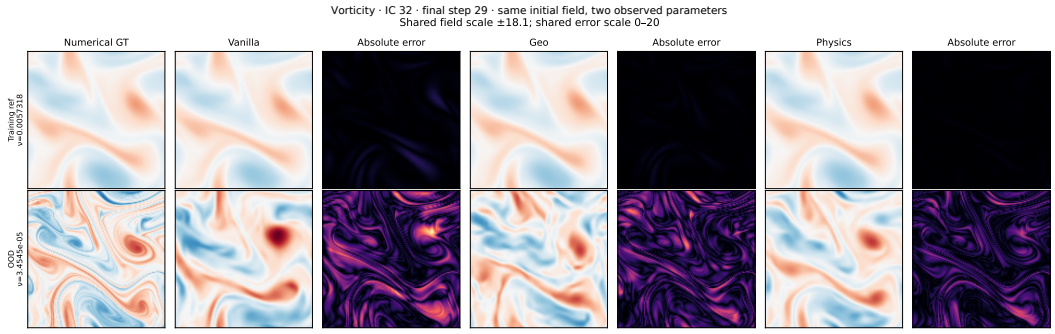


Figure 3: **Visualization on Vorticity under matched initial conditions on final timestep.** Prediction errors are shown for the vanilla, geometry-aligned and physics-structured models. More visualization can be found in Appendix D.4.

data, corresponding to 53.4% and 19.1% improvements. To characterize this geometric change, we compare the original representation $\mathbf{z}$ and aligned representation $\mathbf{q}$ with the physical trajectory using *Angle MAE* for local turning angles, *Sym. Acc. MAE* for second-order temporal variation, and *Lag-1 Cos. MAE* for directional consistency between consecutive increments. As shown in Table 3, alignment reduces all three errors, with particularly large reductions on Vorticity ($78\%/77\%/84\%$) and Wave-2D ($71\%/68\%/74\%$), while Burgers and GS show smaller but consistent improvements. Since these metrics are computed directly on $\mathbf{z}$ and $\mathbf{q}$ without a dynamics predictor, they confirm that the projector itself aligns latent trajectory geometry more closely with physical-field evolution. More analyses are in Appendix D.2

5.5 Parameter-Response Mechanism for PSP

We then isolate the effect of the physics-structured predictor on top of the geometry-aligned representation. As shown in Table 2, adding the PSP further reduces OOD rollout error from .397 to .288 on Vorticity and from .321 to .157 on Wave-2D. To understand this gain, Table 5 evaluates parameter-conditioned responses under matched initial conditions (ICs). The structured predictor consistently improves parameter-conditioned responses in both field and latent spaces. Here, Cos. measures the directional agreement between the predicted and reference parameter-induced changes, while Amp. measures whether the response magnitude is correct, with values closer to 1 indicating better agreement. Figure 3 provides complementary qualitative evidence: when only the governing parameter is changed, the structured predictor more faithfully follows the corresponding numerical solution, particularly under the OOD condition. More analyses are in Appendix D.2.

Table 5: **Parameter-response consistency under matched ICs.**

		Field		Latent	
Dataset	Model	Cos. ↑	Amp. → 1	Cos. ↑	Amp. → 1
Vorticity	PAG	.626	.846	.913	1.142
	+ PSP	**.708**	**.987**	**.924**	**1.043**
Wave-2D	PAG	.958	.979	.939	.990
	+ PSP	**.986**	**.996**	**.947**	**1.002**

6 Conclusion

In this paper, we study how learned state-space structure affects parametric PDE dynamics. Predictive JEPA pretraining yields physically informative representations, while explicit trajectory alignment further improves long-horizon evolution. Building on this, PDE-JEPA combines physics-aligned latent geometry with a structured predictor that separates shared evolution from parameter-dependent responses. Across nine PDE benchmarks, it achieves strong ID performance and substantially improves extrapolation to unseen governing conditions. Further analyses show better physical trajectory alignment and more consistent parameter-conditioned responses. Overall, generalizable physical forecasting depends not only on the evolution model, but also on learning a state-space geometry suited for evolution and extrapolation.

AI USE STATEMENT

Generative AI was employed only as an auxiliary tool during the development of this work. Its use was limited to tasks such as improving written presentation, reorganizing portions of the manuscript, assisting with draft preparation, and supporting code implementation and troubleshooting. The scientific content of the paper, including the derivation and presentation of equations, the analysis of experimental results, and the interpretation of the findings, was produced and determined by the authors. No generative AI system was used to create experimental observations, determine reported results, or make scientific conclusions on behalf of the authors. Any code produced with AI assistance was manually inspected, executed, and validated against the intended implementation and experimental behavior. Similarly, AI-assisted prose was subsequently edited by the authors, and all numerical values reported in the manuscript were verified against the corresponding experimental records. The authors remain fully responsible for the accuracy, validity, and integrity of all material presented in this work.

REFERENCES

Akio Arakawa. Computational design for long-term numerical integration of the equations of fluid motion: Two-dimensional incompressible flow. part i. *Journal of computational physics*, 135(2): 103–114, 1997.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629. IEEE, 2023.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review*, 57(4):483–531, 2015.

Jules Berman and Benjamin Peherstorfer. Colora: Continuous low-rank adaptation for reduced implicit neural modeling of parameterized partial differential equations. *arXiv preprint arXiv:2402.14646*, 2024.

Rafael Bischof, Michal Piovarci, Michael Kraus, Siddhartha Mishra, and Bernd Bickel. Hypino: Multi-physics neural operators via hyperpinns and the method of manufactured solutions. *Advances in Neural Information Processing Systems*, 38:144798–144831, 2026.

Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*, 2022.

Andrew E Brettin, Laure Zanna, and Elizabeth A Barnes. Learning propagators for sea surface height forecasts using koopman autoencoders. *Geophysical Research Letters*, 52(4): e2024GL112835, 2025.

John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

Peter Yichen Chen, Jinxu Xiang, Dong Heon Cho, Yue Chang, GA Pershing, Henrique Teles Maia, Maurizio M Chiaramonte, Kevin Carlberg, and Eitan Grinspun. Crom: Continuous reduced-order modeling of pdes using implicit neural representations. *arXiv preprint arXiv:2206.02607*, 2022.

Woojin Cho, Minju Jo, Haksoo Lim, Kookjin Lee, Dongeun Lee, Sanghyun Hong, and Noseong Park. Parameterized physics-informed neural networks for parameterized pdes. *arXiv preprint arXiv:2408.09446*, 2024.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

Mark C Cross and Pierre C Hohenberg. Pattern formation outside of equilibrium. *Reviews of modern physics*, 65(3):851, 1993.

Aaron Defazio and Konstantin Mishchenko. Learning-rate-free learning by d-adaptation. In *International conference on machine learning*, pp. 7449–7479. PMLR, 2023.

John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

Lawrence C Evans. *Partial differential equations*, volume 19. American mathematical society, 2022.

Francisco Giral, Abhijeet Vishwasrao, Andrea Arroyo Ramo, Mahmoud Golestanian, Federica Tonti, Adrian Lozano-Duran, Steven L Brunton, Sergio Hoyas, Hector Gomez, Soledad Le Clainche, et al. Aerojepa: Learning semantic latent representations for scalable 3d aerodynamic field modeling. *arXiv preprint arXiv:2605.05586*, 2026.

S Godounov. A difference method for numerical calculation of discontinuous solutions of the equation of hydrodynamics. *Matematicheskii Sbornik*, 47(89-3):271–306, 1959.

Vignesh Gopakumar, Ander Gray, Dan Giles, Lorenzo Zanisi, Matt J Kusner, Timo Betcke, Stanislas Pamela, and Marc Peter Deisenroth. Learning physical operators using neural operators. *arXiv preprint arXiv:2602.23113*, 2026.

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*, 2021.

Jan Hagnberger, Marimuthu Kalimuthu, Daniel Musekamp, and Mathias Niepert. Vectorized conditional neural fields: A framework for solving time-dependent parametric partial differential equations. *arXiv preprint arXiv:2406.03919*, 2024.

Jan Hagnberger, Daniel Musekamp, and Mathias Niepert. Calm-pde: Continuous and adaptive convolutions for latent space modeling of time-dependent pdes. *Advances in Neural Information Processing Systems*, 38:160431–160489, 2026.

Xu Han, Han Gao, Tobias Pfaff, Jian-Xun Wang, and Li-Ping Liu. Predicting physics in mesh-reduced space with temporal attention. *arXiv preprint arXiv:2201.09113*, 2022.

Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. Dpot: Auto-regressive denoising operator transformer for large-scale pde pre-training. *arXiv preprint arXiv:2403.03542*, 2024.

Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel De Bezenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for pdes. *Advances in Neural Information Processing Systems*, 37:72525–72624, 2024.

Xiang Huang, Zhanhong Ye, Hongsheng Liu, Shi Ji, Zidong Wang, Kang Yang, Yang Li, Min Wang, Haotian Chu, Fan Yu, et al. Meta-auto-decoder for solving parametric partial differential equations. *Advances in Neural Information Processing Systems*, 35:23426–23438, 2022.

Guang-Shan Jiang and Chi-Wang Shu. Efficient implementation of weighted eno schemes. *Journal of computational physics*, 126(1):202–228, 1996.

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, 2021.

Armand Kassaï Koupaï, Jorge Mifsut Benet, Yuan Yin, Jean-Noël Vittaut, and Patrick Gallinari. Boosting generalization in parametric pde neural solvers through adaptive conditioning. *Advances in Neural Information Processing Systems*, 37:70659–70692, 2024.

Armand Kassaï Koupaï, Lise Le Boudec, Louis Serrano, and Patrick Gallinari. Enma: Tokenwise autoregression for continuous neural pde operators. *Advances in Neural Information Processing Systems*, 38:127341–127409, 2026.

Matthieu Kirchmeyer, Yuan Yin, Jérémie Donà, Nicolas Baskiotis, Alain Rakotomamonjy, and Patrick Gallinari. Generalizing to new physical systems via context-informed dynamics model. In *International conference on machine learning*, pp. 11283–11301. PMLR, 2022.

David Klindt, Yann LeCun, and Randall Balestriero. When does lejepa learn a world model? *arXiv preprint arXiv:2605.26379*, 2026.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMlR, 2019.

Yann LeCun et al. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. SIAM, 2007.

Ruoyan Li, Yizhou Sun, and Wei Wang. Generalized neural operator for parametric and boundary-value problems. *arXiv preprint arXiv:2607.21932*, 2026.

Zijie Li, Saurabh Patil, Francis Ogoke, Dule Shu, Wilson Zhen, Michael Schneier, John R Buchanan Jr, and Amir Barati Farimani. Latent neural pde solver: A reduced-order modeling framework for partial differential equations. *Journal of Computational Physics*, 524:113705, 2025.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11999–12009. IEEE, 2022.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Romit Maulik, Bethany Lusch, and Prasanna Balaprakash. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *Physics of Fluids*, 33(3), 2021.

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. *Advances in Neural Information Processing Systems*, 37:119301–119335, 2024.

Michael McCabe, Payel Mukhopadhyay, Tanya Marwah, Bruno Regaldo-Saint Blancard, Francois Rozet, Cristiana Diaconu, Lucas Meyer, Kaze WK Wong, Hadi Sotoudeh, Alberto Bietti, et al. Walrus: A cross-domain foundation model for continuum dynamics. *arXiv preprint arXiv:2511.15684*, 2025.

Grégoire Mialon, Quentin Garrido, Hannah Lawrence, Danyal Rehman, Yann LeCun, and Bobak Kiani. Self-supervised learning with lie symmetries for partial differential equations. *Advances in Neural Information Processing Systems*, 36:28973–29004, 2023.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022.

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-jepa 2.1: Unlocking dense features in video self-supervised learning. *arXiv preprint arXiv:2603.14482*, 2026.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Yash Patel, Abhiti Mishra, and Ambuj Tewari. Continuum transformers perform in-context learning by operator gradient descent. In *International Conference on Learning Representations*, volume 2026, pp. 15968–15998, 2026.

Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Helen Qu, Rudy Morel, Michael McCabe, Alberto Bietti, François Lanusse, Shirley Ho, and Yann LeCun. Representation learning for spatiotemporal physical systems. *arXiv preprint arXiv:2603.13227*, 2026.

Francesco Regazzoni, Stefano Pagani, Matteo Salvador, Luca Dede', and Alfio Quarteroni. Learning the intrinsic dynamics of spatio-temporal processes through latent dynamics networks. *Nature Communications*, 15(1):1834, 2024.

Louis Serrano, Armand Kassaï Koupaï, Thomas X Wang, Pierre Erbacher, and Patrick Gallinari. Zebra: In-context generative pretraining for solving parametric pdes. *arXiv preprint arXiv:2410.03437*, 2024a.

Louis Serrano, Thomas X Wang, Etienne Le Naour, Jean-Noël Vittaut, and Patrick Gallinari. Aroma: Preserving spatial structure for latent pde modeling with local neural fields. *Advances in Neural Information Processing Systems*, 37:13489–13521, 2024b.

Makoto Takamoto, Francesco Alesiani, and Mathias Niepert. Learning neural pde solvers with parameter-guided channel attention. In *International Conference on Machine Learning*, pp. 33448–33467. PMLR, 2023.

Zhentao Tan, Yuze Hao, Boyi Zou, Mingsheng Long, Yi Yang, and Gang Bao. Harnessing ai for inverse partial differential equation problems: Past, present, and prospects. *arXiv preprint arXiv:2605.16966*, 2026a.

Zhentao Tan, Ruijie Quan, and Yi Yang. From points to edges: Edge-conditioned spectral operators for physics-sensitive pde learning. *arXiv preprint arXiv:2608.06894*, 2026b.

Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models? *arXiv preprint arXiv:2512.24497*, 2025.

Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022.

Juan Diego Toscano, Vivek Oommen, Alan John Varghese, Zongren Zou, Nazanin Ahmadi Daryakenari, Chenxi Wu, and George Em Karniadakis. From pinns to pikans: Recent advances in physics-informed machine learning. *Machine Learning for Computational Science and Engineering*, 1(1): 15, 2025.

Lloyd N Trefethen. *Spectral methods in MATLAB*. SIAM, 2000.

Haixin Wang, Yadi Cao, Zijie Huang, Yuxuan Liu, Peiyan Hu, Xiao Luo, Zezheng Song, Wanjia Zhao, Jilin Liu, Jinan Sun, et al. Recent advances on machine learning for computational fluid dynamics: A survey. *arXiv preprint arXiv:2408.12171*, 2024.

Hong Wang, Haiyang Xin, Jie Wang, Xuanze Yang, Fei Zha, Yan Jiang, et al. Mixture-of-experts operator transformer for large-scale pde pre-training. *Advances in Neural Information Processing Systems*, 38:31498–31527, 2026a.

Tian Wang and Chuang Wang. Latent neural operator for solving forward and inverse pde problems. *Advances in Neural Information Processing Systems*, 37:33085–33107, 2024.

Ying Wang, Oumayma Bounou, Gaoyue Zhou, Randall Balestriero, Tim GJ Rudner, Yann LeCun, and Mengye Ren. Temporal straightening for latent planning. *arXiv preprint arXiv:2603.12231*, 2026b.

Gerhard Wanner and Ernst Hairer. *Solving ordinary differential equations II*, volume 375. Springer Berlin Heidelberg New York, 1996.

Steffen Wiewel, Moritz Becher, and Nils Thuerey. Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer graphics forum*, volume 38, pp. 71–82. Wiley Online Library, 2019.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.

Haixu Wu, Minghao Guo, Zongyi Li, Zhiyang Dou, Mingsheng Long, Kaiming He, and Wojciech Matusik. Geopt: Scaling physics simulation via lifted geometric pre-training. *arXiv preprint arXiv:2602.20399*, 2026.

Tailin Wu, Takashi Maruyama, and Jure Leskovec. Learning to accelerate partial differential equations via latent global evolution. *Advances in Neural Information Processing Systems*, 35:2240–2253, 2022.

Haodong Yan, Jiaguan Zhu, Mingyuan Jia, Ruiqing Yin, Junjie He, Zhide Zhong, Junfeng Li, Jinxuan Lu, Hengtao Li, Tianran Zhang, et al. Is forward prediction enough? physical state grounding for jepa world models. *arXiv preprint arXiv:2608.06799*, 2026.

Hao-Ran Yang and Chuan-Xian Ren. A physics-preserved transfer learning method for differential equations. *Advances in Neural Information Processing Systems*, 38:11829–11856, 2026.

Liu Yang and Stanley J Osher. Pde generalization of in-context operator networks: A study on 1d scalar nonlinear conservation laws. *Journal of Computational Physics*, 519:113379, 2024.

Liu Yang, Siting Liu, Tingwei Meng, and Stanley J Osher. In-context operator learning with data prompts for differential equation problems. *Proceedings of the National Academy of Sciences*, 120(39):e2310142120, 2023.

Yuan Yin, Ibrahim Ayed, Emmanuel de Bézenac, Nicolas Baskiotis, and Patrick Gallinari. Leads: Learning dynamical systems that generalize across environments. *Advances in Neural Information Processing Systems*, 34:7561–7573, 2021.

Anthony Zhou and Amir Barati Farimani. Masked autoencoders are pde learners. *arXiv preprint arXiv:2403.17728*, 2024.

Hang Zhou, Yuezhou Ma, Haixu Wu, Haowen Wang, and Mingsheng Long. Unisolver: Pde-conditional transformers towards universal neural pde solvers. *arXiv preprint arXiv:2405.17527*, 2024.

APPENDIX CONTENTS

A Dataset details **17**

A.1 Advection . . . 17

A.2 Burgers equation . . . 18

A.3 Heat equation . . . 19

A.4 Wave-B: one-dimensional waves with varying boundaries . . . 20

A.5 Combined Equation . . . 20

A.6 Vorticity: two-dimensional incompressible flow . . . 21

A.7 HeterNS . . . 22

A.8 Wave-2D: damped two-dimensional waves . . . 23

A.9 Gray–Scott Equation . . . 24

B Architecture Details **24**

B.1 Pretraining Stage . . . 25

B.2 Geometry Alignment Stage . . . 25

B.3 Predictor Learning Stage . . . 27

B.4 Decoder Learning Stage . . . 28

C Implementation Details **30**

C.1 PDE-JEPA Implementation . . . 30

C.1.1 Physics-Structured Latent Predictor Implementations . . . 30

C.1.2 Latent time integration. . . . 31

C.2 Baseline Implementations . . . 33

D Additional experiments **39**

D.1 Representation Analyses . . . 39

D.1.1 Models and evaluation protocol . . . 39

D.1.2 Parameter Information Analyses . . . 40

D.1.3 Variance concentration and latent organization . . . 41

D.1.4 Temporal and spatial organization . . . 42

D.1.5 Extension to other PDEs . . . 43

D.1.6 Direct Reconstruction versus Autoregressive Rollout . . . 46

D.2 Mechanism Analyses . . . 46

D.2.1 Evaluation protocol and dynamics diagnostics . . . 47

D.2.2 Parameter-dependent dynamics . . . 47

D.2.3 Latent trajectory geometry . . . 49

D.2.4 Field and latent rollout accuracy . . . 50

D.2.5 Physics Supervision Reduces Error Propagation . . . 54

D.2.6 Extension to 1D PDEs . . . 55

D.3 Extended Experiments and Ablation Studies 56
D.3.1 Full OOD results 56
D.3.2 Geometry Alignment Module 56
D.3.3 Predictor Module 58
D.4 Trajectory Visualizations 60
D.4.1 Advection 60
D.4.2 Burgers 60
D.4.3 Heat 60
D.4.4 Wave-B 60
D.4.5 Combined Equation 61
D.4.6 Wave-2D 61
D.4.7 Vorticity 61
D.4.8 Gray–Scott 62
D.4.9 HeterNS 62

A DATASET DETAILS

We consider nine PDE datasets: Advection, Burgers, Heat, Wave-B, Combined Equation, Vorticity, HeterNS, Wave-2D, and Gray–Scott. Advection, Wave-B, Combined Equation, Wave-2D, and Vorticity are followed from Zebra (Serrano et al., 2024a). Burgers and Heat are generated following MP-PDE (Brandstetter et al., 2022), with an additional forcing coefficient, while all other settings are kept consistent with Zebra. Gray–Scott is adopted from ENMA (Kassaï Koupaï et al., 2026), and HeterNS from Unisolver (Zhou et al., 2024). A summary of the dataset configurations is provided in Table 6 and 7, with further details given below.

Table 6: **Datasets overview.** Counts denote trajectory number. A dash means that no separate OOD split is specified for the audited dataset version. One-dimensional shapes use (T, X); two-dimensional shapes use (C, X, Y, T).

Dataset	Shape per trajectory	Train	Val.	Test	OOD
Advection	140×256	12,000	120	120	–
Burgers	250×256	12,000	120	120	–
Heat	250×256	12,000	120	120	–
Wave-B	250×256	12,000	120	120	–
Combined Equation	140×256	12,000	480	480	120
Vorticity	$1 \times 128 \times 128 \times 30$	12,000	1,200	1,200	120
Wave-2D	$2 \times 64 \times 64 \times 30$	12,000	1,200	1,200	120
Gray–Scott	$2 \times 32 \times 32 \times 20$	12,000	1,200	1,200	120
HeterNS	$1 \times 64 \times 64 \times 20$	15,000	1,500	1,500	1,500

Table 7: **In-distribution (In-D) and out-of-distribution (Out-D) parameter settings for each dataset.**

Dataset	Parameter	In-D	Out-D
Combined	α	$\mathcal{U}([0, 1])$	$[1.0682, 1.7581]$
	β	$\mathcal{U}([0, 0.4])$	same as In-D
	γ	$\mathcal{U}([0, 1])$	same as In-D
Vorticity	ν	$[10^{-3}, 10^{-2}]$	$[10^{-5}, 10^{-4}]$
HeterNS	ν	$\{10^{-5}, 5\times10^{-5}, 10^{-4}, 5\times10^{-4}, 10^{-3}\}$	Interp.: $(\{2, 3, 4, 6, 7, 8, 9\}\times10^{-5})$ $\cup$ $(\{2, 3, 4, 6, 7, 8, 9\}\times10^{-4})$ Extra.: $\{2, 3, 4, 5, 6, 7, 8, 9\}\times10^{-3}$
	m	$\{1, 2, 3\}$	$\{1, 2, 3\}$ (viscosity OOD) $\{0.5, 1.5, 2.5, 3.5\}$ (forcing OOD)
Wave-2D	c	$[100,\ 500]$	$[500,\ 550]$
	k	$[0,\ 50]$	$[50,\ 60]$
Gray–Scott	F	$\mathcal{U}([0.023, 0.045])$	$\mathcal{U}([0.045, 0.0467])$
	k	$\mathcal{U}([0.0590, 0.0640])$	$\mathcal{U}([0.0570, 0.0590])$

A.1 ADVECTION

Equation and environments. We solve the constant-speed transport equation

$$\partial_t u + \beta\, \partial_x u = 0, \qquad x \in [0, L), \qquad L = 128, \tag{19}$$

with periodic boundaries. The sole environment parameter is the speed $\beta \sim \mathrm{Unif}[0, 4]$. We draw 1,200 training environments and 12 environments for each of validation and testing, with ten independently seeded initial conditions per environment.

Initial conditions. Each trajectory starts from a random superposition of three cosine modes,

$$u_0(x) = \sum_{j=1}^{3} A_j \cos\left(\frac{2\pi \ell_j x}{L} + \phi_j\right), \tag{20}$$
$$A_j \sim \mathrm{Unif}[-0.5{,}0.5], \quad \phi_j \sim \mathrm{Unif}[0, 2\pi], \quad \ell_j \sim \mathrm{Unif}\{1, 2, 3, 4, 5\}.$$

The mode coefficients are sampled independently for each initial condition.

Numerical generation. We use the adapter provided by ENMA to generate trajectories analytically via periodic translation,

$$u(x,t) = u_0\left((x - \beta t) \bmod L\right),$$

on a 256-point spatial grid. The solution is evaluated at 250 uniformly spaced times over $t \in [0, 100]$, with the last 140 frames retained, corresponding approximately to $t \in [44.18, 100]$. The visualization is shown in Figure 4.

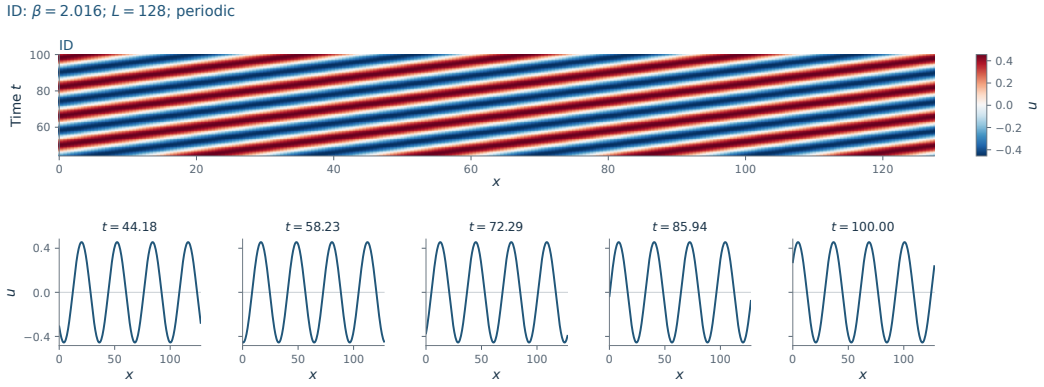


Figure 4: **Advection.** Transport speed $\beta = 2.01597$, within the training support $[0, 4]$, with periodic boundaries on a domain of length 128. The stored segment spans $t \simeq 44.18$ to 100.

A.2 Burgers equation

Equation and environments. On a periodic interval of length $L = 16$, we solve

$$\partial_t u + \partial_x\left(\tfrac{1}{2}u^2 - \beta\,\partial_x u\right) = F(t,x), \qquad \beta \sim \mathrm{LogUniform}[10^{-3}, 5], \tag{21}$$

where the forcing is

$$F(t,x) = \sum_{j=1}^{5} A_j \sin\left(\omega_j t + \frac{2\pi \ell_j x}{L} + \phi_j\right), \tag{22}$$

$$A_j \sim \mathrm{Unif}[-0.5, 0.5], \quad \omega_j \sim \mathrm{Unif}[-0.4, 0.4], \quad \ell_j \sim \mathrm{Unif}\{1,2,3\}, \quad \phi_j \sim \mathrm{Unif}[0, 2\pi].$$

Here log-uniform sampling means that $\log \beta$ is uniform between the logarithms of the two endpoints. Diffusivity and all 20 forcing coefficients are fixed within an environment and vary between environments.

Initial conditions. For each trajectory, we independently draw

$$u_0(x) = \sum_{j=1}^{5} \widetilde{A}_j \sin\left(\frac{2\pi \widetilde{\ell}_j x}{L} + \widetilde{\phi}_j\right), \tag{23}$$

using the amplitude, integer mode, and phase distributions in Eq. equation 22. These initial-condition coefficients are independent of the environment's forcing coefficients. The split contains 1,200/12/12 environments for training/validation/testing, with ten trajectories per environment. All three splits use the same parameter support.

Numerical generation. The local MP-PDE implementation uses WENO reconstruction (Jiang & Shu, 1996) with Godunov flux splitting (Godounov, 1959) for the nonlinear term, finite differences for diffusion, and an adaptive Dormand–Prince 4/5 integrator (Dormand & Prince, 1980) with tolerances of 10^{-5}. Trajectories are generated on 256 spatial points over $t \in [0, 4]$, with 250 snapshots evaluated and every tenth frame retained to form 25-frame sequences. We follow the implementation-specific spatial discretization and forcing range $[-0.4, 0.4]$ used by the released generator. The visualization is shown in Figure 5.

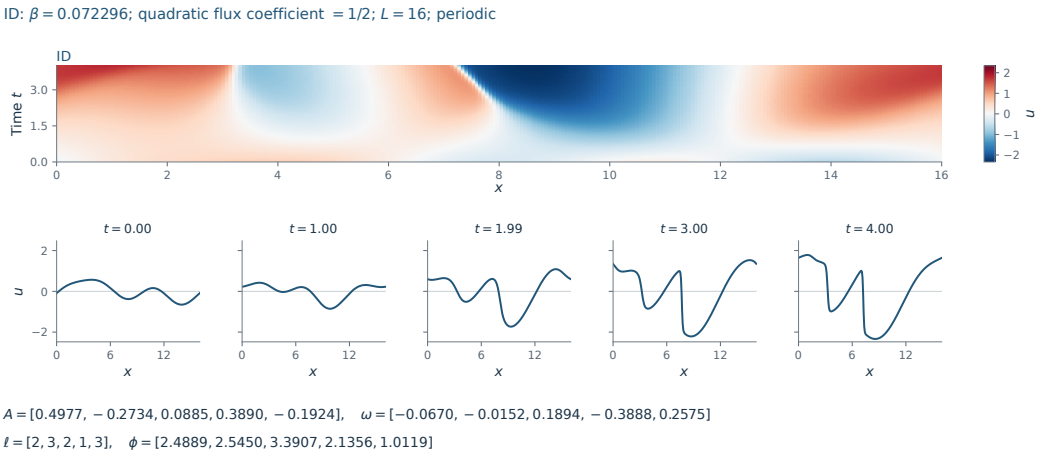


Figure 5: **Burgers.** Viscosity $\beta = 0.072296$, within the training support $[10^{-3}, 5]$, with periodic boundaries on a domain of length 16. The forcing is $F(t,x) = \sum_{j=1}^{5} A_j \sin(\omega_j t + 2\pi \ell_j x/16 + \phi_j)$; its coefficients A_j, ω_j, ℓ_j, and ϕ_j are listed beneath.

A.3 Heat equation

Equation and environments. We solve

$$\partial_t u = \beta\, \partial_{xx} u + F(t,x), \qquad x \in [0,16], \qquad \beta \sim \text{LogUniform}[10^{-3}, 5], \tag{24}$$

with periodic differentiation. The forcing is the five-mode process in Eq. equation 22;

Initial conditions. Initial conditions are independently sampled from Eq. equation 23. Following Burgers, one environment fixes the diffusivity and the complete forcing realization, while its ten trajectories have independent initial conditions.

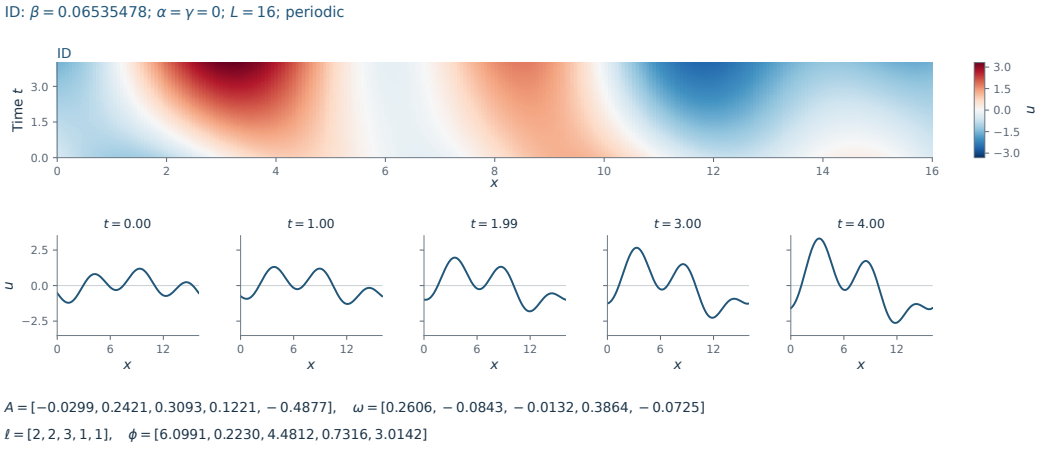


Figure 6: **Heat.** Diffusivity $\beta = 0.0653548$, within the training support $[10^{-3}, 5]$. The trajectory solves $\partial_t u = \beta \partial_{xx} u + F(t,x)$ on a periodic interval of length 16, with the five-mode forcing coefficients shown beneath.

Numerical generation. We generate the heat trajectories using the local MP-PDE implementation with the nonlinear and dispersive terms disabled, leaving a finite-difference diffusion operator and an adaptive Dormand–Prince 4/5 integrator (Dormand & Prince, 1980) with tolerances of 10^{-5}. Each trajectory is evaluated on 256 spatial points at 250 uniformly spaced times over $t \in [0,4]$. Every tenth frame is retained to form 25-frame sequences; we use 1,200/12/12 train/validation/test environments with ten trajectories per environment. The visualization is shown in Figure 6.

A.4 WAVE-B: ONE-DIMENSIONAL WAVES WITH VARYING BOUNDARIES

Equation and environments. We consider

$$\partial_{tt} u - c^2 \partial_{xx} u = 0, \qquad x \in [-8, 8], \quad c = 2. \tag{25}$$

Each endpoint independently uses either homogeneous Dirichlet conditions ($u = 0$, denoted D) or homogeneous Neumann conditions ($\partial_x u = 0$, denoted N). This defines four environments: DD, DN, ND, and NN. Wave speed is fixed and is not an environment variable.

Initial conditions. For each trajectory, a pulse center $s \sim \mathrm{Unif}[-4, 4]$ determines both initial displacement and initial velocity:

$$u(x, 0) = \exp[-(x - s)^2], \qquad \partial_t u(x, 0) = -2c(x - s) \exp[-(x - s)^2]. \tag{26}$$

The four environments each contain 3,000 training trajectories, 30 validation trajectories, and 30 test trajectories. All four boundary combinations occur in every split: this split tests generalization to new initial conditions, not unseen boundary types.

Numerical generation. We generate the trajectories using the MP-PDE Chebyshev differentiation operator (Trefethen, 2000) on a 256-point nonuniform grid, with an implicit Radau integrator (Wanner & Hairer, 1996) over $t \in [0, 100]$ and tolerances of 10^{-3}. Each trajectory is evaluated at 250 output times. For model input, the sequence is reordered into forward physical time and every tenth frame is retained, yielding 25-frame trajectories. The visualization is shown in Figure 7.

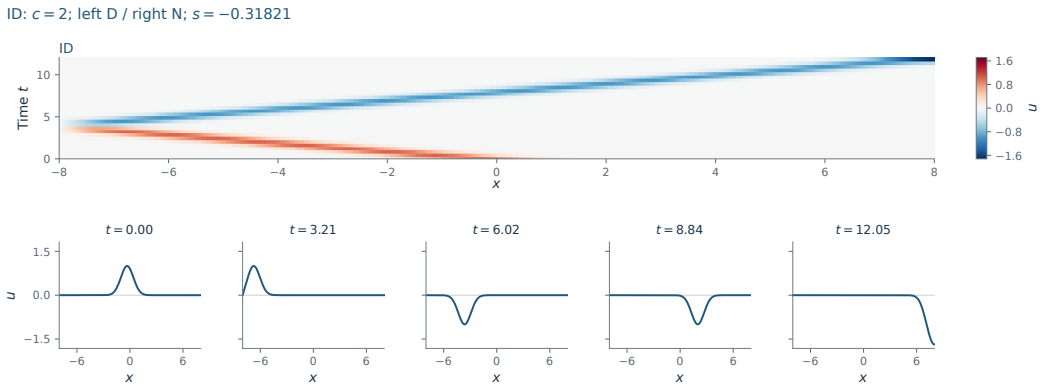


Figure 7: **Wave-B.** Wave speed $c = 2$, a left Dirichlet and a right Neumann boundary, and initial pulse center $s = -0.31821$. The initial displacement and velocity are $u_0(x) = e^{-(x-s)^2}$ and $v_0(x) = -2c(x - s)u_0(x)$. The plot shows $t \simeq 0$–12.05 after restoring forward time from the reversed storage order, retaining the original nonuniform Chebyshev grid.

A.5 COMBINED EQUATION

Equation and environments. The Combined Equation follows the setting of Zebra, which adopts the benchmark introduced by MP-PDE et al. (Brandstetter et al., 2022) without the forcing term. The dynamics are governed by

$$\partial_t u + \partial_x \left(\alpha u^2 - \beta \partial_x u + \gamma \partial_{xx} u \right) = 0, \tag{27}$$

with initial conditions given by random finite sums of sinusoidal modes,

$$u_0(x) = \sum_{j=1}^{J} A_j \sin\left(\frac{2\pi \ell_j x}{L} + \phi_j \right). \tag{28}$$

For training, 1,200 parameter triplets are sampled uniformly from $\alpha \in [0, 1]$, $\beta \in [0, 0.4]$, and $\gamma \in [0, 1]$, with ten trajectories generated for each parameter setting, yielding 12,000 training trajectories. An additional 120 trajectories are used for testing. Following Zebra, solutions are generated using

the MP-PDE solver (Brandstetter et al., 2022) on 256 spatial points over $t \in [0, 10]$, with 140 temporal snapshots. Temporal downsampling by a factor of ten gives trajectories of shape 256×14.

OOD data. The OOD split contains 120 trajectories with $\alpha \in [1.0682, 1.7581]$, extending beyond the ID range $\alpha \in [0, 1]$, while β and γ remain within their respective ID ranges. Separate OOD training and validation splits contain ten trajectories each and are excluded from the ID data. The visualization is shown in Figure 8.

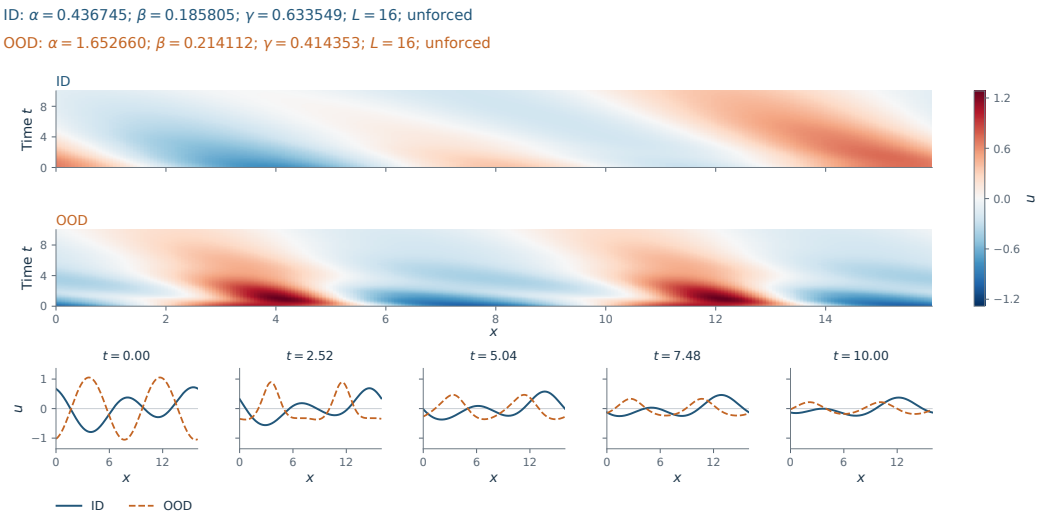


Figure 8: **Combined Equation.** ID: $(\alpha, \beta, \gamma) = (0.436745, 0.185805, 0.633549)$; OOD: $(\alpha, \beta, \gamma) = (1.652660, 0.214112, 0.414353)$. The OOD nonlinear-transport coefficient exceeds the ID interval $[0, 1]$, while β and γ remain within their ID ranges $[0, 0.4]$ and $[0, 1]$. Both trajectories use periodic boundaries, no external forcing, and the same saved coordinates on $x \in [0, 16)$ and $t \in [0, 10]$. Their initial fields and all three coefficients differ, so every trajectories have different initial conditions.

A.6 Vorticity: two-dimensional incompressible flow

Equation and environments. The dataset contains unforced two-dimensional incompressible flow in vorticity form on a periodic square:

$$\partial_t \omega + (\mathbf{u} \cdot \nabla)\omega = \nu \Delta \omega, \qquad \nabla \cdot \mathbf{u} = 0, \tag{29}$$

where velocity is recovered from a streamfunction through a Poisson solve. Viscosity ν is the environment parameter. The ID support is $[10^{-3}, 10^{-2}]$, with 1,200 training and 120 validation/test environments per split, each containing ten trajectories. The OOD release contains 12 viscosities on a linear grid spanning $[10^{-5}, 10^{-4}]$, again with ten trajectories per viscosity.

Initial conditions. The initial vorticity fields are generated from the prescribed energy spectrum

$$E(k) = \frac{4}{3}\sqrt{\pi}\left(\frac{k}{k_0}\right)^4 \frac{1}{k_0} \exp\left[-\left(\frac{k}{k_0}\right)^2\right], \tag{30}$$

with the corresponding vorticity spectrum

$$\omega(k) = \sqrt{\frac{E(k)}{\pi k}}. \tag{31}$$

Numerical generation. We follow the numerical solver used in Zebra (Serrano et al., 2024a), combining a five-point finite-difference Laplacian, the Arakawa Jacobian (Arakawa, 1997), an FFT-based Poisson solver (Cooley & Tukey, 1965), and fourth-order Runge–Kutta integration (Butcher, 2016). The simulation is performed on a 512×512 grid over $t \in [0, 2]$, and the resulting trajectories

are spatially and temporally subsampled to 128×128 resolution with 30 frames. The visualization is shown in Figure 9.

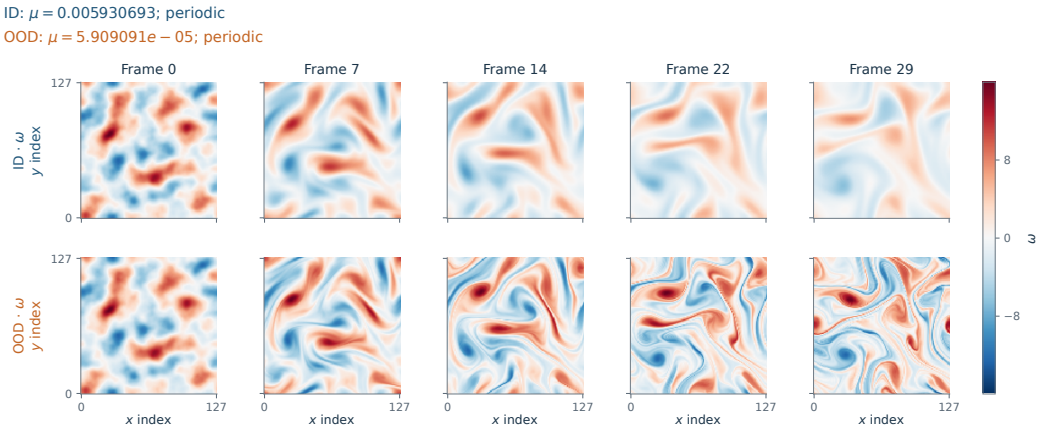


Figure 9: **Vorticity.** ID: $\nu = 0.00593069$; OOD: $\nu = 5.90909 \times 10^{-5}$, below the training support $[10^{-3}, 10^{-2}]$. ID and OOD both use initial-condition, and their first saved vorticity fields are exactly equal. Rows compare the same five saved-frame positions, with periodic boundaries and a shared symmetric color scale.

A.7 HeterNS

Equation and environments. HeterNS considers the two-dimensional incompressible Navier–Stokes equation in vorticity form on the unit torus. All settings are identical to those used in Uni-Solver (Zhou et al., 2024); we include this description only for completeness.

$$\begin{aligned} \partial_t \omega + \mathbf{u} \cdot \nabla \omega &= \nu \Delta \omega + f(\mathbf{x}), \\ \nabla \cdot \mathbf{u} &= 0, \end{aligned} \tag{32}$$

where ν is the viscosity coefficient and the forcing is

$$f(\mathbf{x}) = 0.1 \left[\sin\left(\omega_f \pi (x_1 + x_2)\right) + \cos\left(\omega_f \pi (x_1 + x_2)\right) \right]. \tag{33}$$

The PDE environment is therefore determined by the viscosity ν and forcing frequency ω_f. The training environments use $\nu \in \left\{10^{-5}, 5 \times 10^{-5}, 10^{-4}, 5 \times 10^{-4}, 10^{-3}\right\}, \omega_f \in \{1, 2, 3\}$, giving 15 distinct PDE configurations. Each configuration contains 1,000 trajectories, yielding 15,000 training trajectories in total. The ID test set keeps the same PDE configurations while using unseen initial conditions. The visualization is shown in Figure 10.

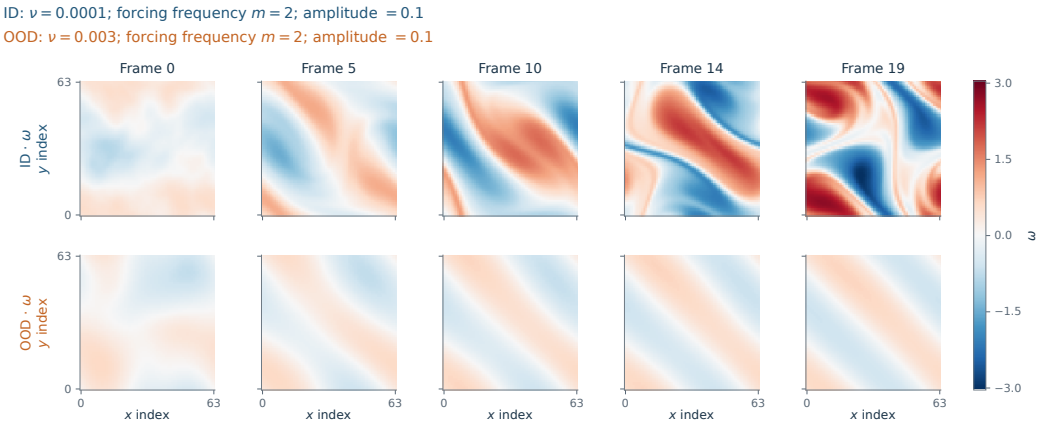


Figure 10: **HeterNS.** ID: $\nu \simeq 10^{-4}$; OOD: $\nu \simeq 3 \times 10^{-3}$, above the training range $[10^{-5}, 10^{-3}]$. Both use $F(x, y) = 0.1[\sin(2\pi(x + y)) + \cos(2\pi(x + y))]$.

OOD data. We consider three OOD settings. For viscosity interpolation, we use $\nu \in \left(\{2,3,4,6,7,8,9\} \times 10^{-5}\right) \cup \left(\{2,3,4,6,7,8,9\} \times 10^{-4}\right)$, with $m \in \{1,2,3\}$. For viscosity extrapolation, we use $\nu \in \{2,3,4,5,6,7,8,9\} \times 10^{-3}$, again with $m \in \{1,2,3\}$. For forcing OOD, we fix $\nu = 10^{-5}$ and use unseen forcing frequencies $m \in \{0.5, 1.5, 2.5, 3.5\}$. We sample 1,500 OOD trajectories in total, consisting of 900 viscosity-interpolation trajectories, 510 viscosity-extrapolation trajectories, and 90 forcing-OOD trajectories.

A.8 WAVE-2D: DAMPED TWO-DIMENSIONAL WAVES

Equation and environments. The released data represent

$$\partial_{tt} u = c^2 \Delta u - k \partial_t u, \tag{34}$$

with two stored channels, $(u, \partial_t u)$. Direct inspection establishes $c \in [100, 500]$ and $k \in [0, 50]$ for the ID data. Training uses 1,200 distinct parameter pairs selected from a 100×100 Cartesian parameter grid; validation and test each contain 120 pairs. Each pair has 10 initial conditions.

Initial conditions. The initial condition is constructed as a sum of five Gaussian functions,

$$\omega_0(x, y) = \sum_{i=1}^{5} \exp\left(-\frac{(x - x_i)^2 + (y - y_i)^2}{2\sigma_i^2}\right), \tag{35}$$

where $x_i, y_i \sim \mathcal{U}([0, 1])$ and $\sigma_i \sim \mathcal{U}([0.025, 0.1])$. Each Gaussian has unit amplitude.

Numerical generation. Following Zebra (Serrano et al., 2024a), the spatial domain is discretized on a 64×64 grid using a 5×5 discrete Laplacian operator with dirichlet boundary conditions. The dynamics are integrated using a fourth-order Runge–Kutta scheme with time step $\Delta t = 6.25 \times 10^{-6}$ over $t \in [0, 5 \times 10^{-3}]$. The visualization is shown in Figure 11.

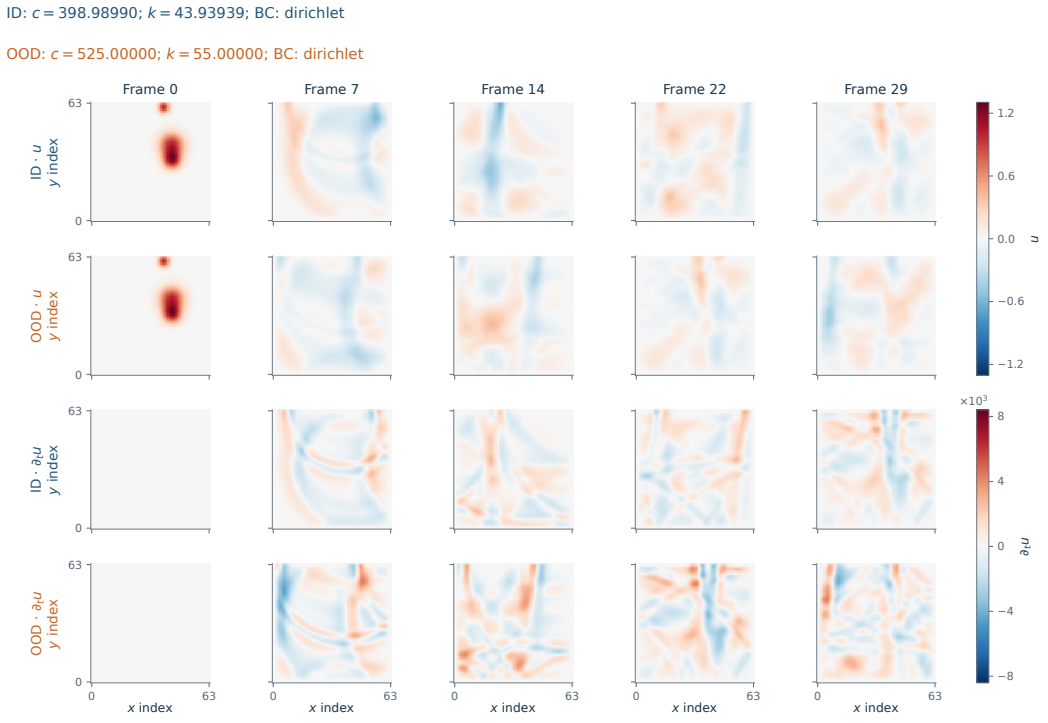


Figure 11: **Wave-2D:.** ID: $(c, k) = (398.98990, 43.93939)$; OOD: $(c, k) = (525, 55)$, outside the training ranges $c \in [100, 500]$ and $k \in [0, 50]$.

OOD data. The designated OOD release has 12 parameter pairs from the 5×5 grid $c \in [500, 550]$, $k \in [50, 60]$, with ten trajectories per pair. Eleven pairs are outside the ID support, while $(c, k) =$

$(500, 50)$ lies on its boundary. Thus this file contains 110 strict parameter-extrapolation trajectories and ten boundary-support trajectories.

A.9 Gray–Scott Equation

Equation and environments. The Gray–Scott dataset describes a two-dimensional reaction–diffusion system governed by

$$\begin{aligned} \frac{\partial u}{\partial t} &= D_u \Delta u - uv^2 + F(1-u), \\ \frac{\partial v}{\partial t} &= D_v \Delta v - uv^2 - (F+k)v, \end{aligned} \tag{36}$$

where periodic boundary conditions are imposed and the diffusion coefficients are fixed to $D_u = 0.102$ and $D_v = 0.204$. For ID trajectories, the reaction parameters are sampled as $F \sim \mathcal{U}([0.023, 0.045])$ and $k \sim \mathcal{U}([0.0590, 0.0640])$. For OOD evaluation, following ENMA (Kassaï Koupaï et al., 2026), we use $F \sim \mathcal{U}([0.045, 0.0467])$ and $k \sim \mathcal{U}([0.0570, 0.0590])$. The spatial domain is discretized on a 32×32 grid with spatial resolution $\Delta s = 2$. The visualization is shown in Figure 12.

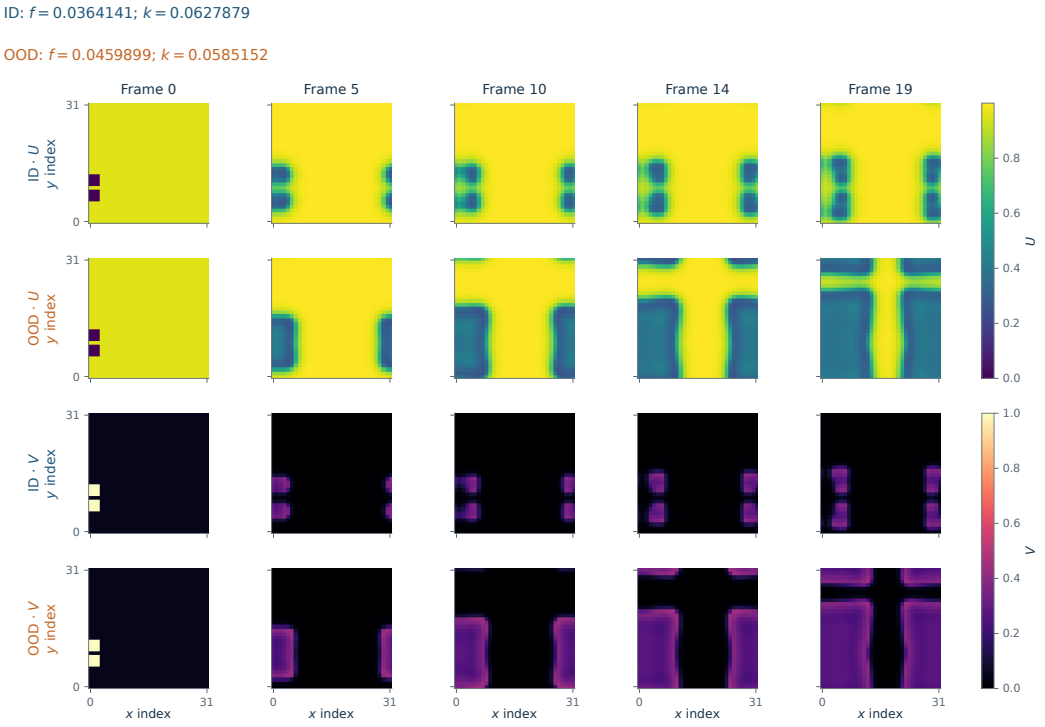


Figure 12: **Gray–Scott.** ID: $(f, k) = (0.0364141, 0.0627879)$; OOD: $(f, k) = (0.0459899, 0.0585152)$, outside the training ranges $f \in [0.023, 0.045]$ and $k \in [0.059, 0.064]$.

B Architecture Details

Our framework is trained in four stages: predictive representation pretraining, geometry alignment, latent predictor learning, and decoder training. We describe the architecture and tensor flow of each stage below. Let an input trajectory be

$$\mathbf{X} \in \mathbb{R}^{B \times T \times C \times H \times W},$$

where B, T, C, H, and W denote the batch size, temporal length, number of physical channels, and spatial resolution, respectively.

B.1 PRETRAINING STAGE

Encoder. The pretraining stage learns physical representations using a joint-embedding predictive objective (JEPA). Given a trajectory

$$\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_T], \qquad \mathbf{x}_t \in \mathbb{R}^{C \times H \times W},$$

each frame is partitioned into non-overlapping spatial patches and projected into D-dimensional tokens. With a temporal tubelet size of one, this patch embedding does not mix information across adjacent frames, yielding N spatial tokens per time step.

The resulting spatiotemporal tokens are then processed jointly by a Vision Transformer. Specifically, the $T \times N$ tokens are arranged as a single sequence and augmented with spatial and temporal positional information before being passed through a stack of multi-head self-attention and feed-forward blocks. The encoder therefore produces

$$\mathbf{Z} = f_\theta(\mathbf{X}) \in \mathbb{R}^{B \times T \times N \times D},$$

where each temporal slice

$$\mathbf{z}_t = [\mathbf{Z}]_t \in \mathbb{R}^{N \times D}$$

is obtained in the context of the full trajectory rather than by independently encoding $\mathbf{x}_t$.

Masked latent prediction. During pretraining, a subset of latent tokens is masked from the context encoder. A predictor receives the visible context representations together with embeddings corresponding to the masked target positions and predicts their latent representations. The prediction target is produced by a momentum-updated target encoder.

Let

$$\mathbf{Z}^c \in \mathbb{R}^{B \times N_c \times D}$$

denote the visible context tokens and

$$\mathbf{Z}^t \in \mathbb{R}^{B \times N_t \times D}$$

the latent representations of the target tokens, where N_c and N_t denote the number of context and target tokens, respectively. The latent predictor maps the context representation to

$$\widehat{\mathbf{Z}}^t = p_\psi(\mathbf{Z}^c) \in \mathbb{R}^{B \times N_t \times D}.$$

The pretraining objective aligns the predicted target representations with those generated by the target encoder,

$$\mathcal{L}_{\text{pre}} = \mathcal{D}\left(\widehat{\mathbf{Z}}^t, \mathbf{Z}^t\right),$$

where $\mathcal{D}$ denotes the latent-space prediction loss. After pretraining, the encoder f_θ is retained as the representation model for the following stages.

B.2 GEOMETRY ALIGNMENT STAGE

The pretrained representation preserves predictive information about the physical trajectory, but its latent coordinates are not explicitly constrained to reflect the evolution geometry of the underlying physical states. We therefore introduce a lightweight geometry projector to adapt the latent state space before learning the dynamics model.

Geometry projector. The pretrained encoder is frozen during this stage. For each latent token

$$\mathbf{z}_{t,n} \in \mathbb{R}^D,$$

the geometry projector applies a token-wise residual transformation,

$$\mathbf{q}_{t,n} = \mathbf{z}_{t,n} + g_\phi(\mathbf{z}_{t,n}),$$

where

$$g_\phi : \mathbb{R}^D \to \mathbb{R}^D$$

is a multilayer perceptron consisting of normalization, channel expansion to an intermediate dimension D_g, nonlinear activation, and projection back to D.

The projector therefore preserves the complete latent tensor shape,

$$\mathbf{Q} = \mathbf{Z} + g_\phi(\mathbf{Z}) \in \mathbb{R}^{B\times T\times N\times D}.$$

The transformation is independently applied to every token and time step. In particular, the projector does not perform temporal or spatial-token mixing, and only adjusts the latent channel coordinates.

Temporal geometry alignment. To characterize the temporal geometry of the physical trajectory, each physical state is flattened as

$$\widetilde{\mathbf{x}}_t \in \mathbb{R}^{CHW},$$

while each projected latent state is flattened across its token and channel dimensions,

$$\widetilde{\mathbf{q}}_t \in \mathbb{R}^{ND}.$$

Temporal displacement vectors are then computed independently in the two state spaces,

$$\Delta\mathbf{x}_t = \widetilde{\mathbf{x}}_{t+1} - \widetilde{\mathbf{x}}_t,$$

and

$$\Delta\mathbf{q}_t = \widetilde{\mathbf{q}}_{t+1} - \widetilde{\mathbf{q}}_t.$$

Although the physical and latent states have different feature dimensions, i.e.,

$$CHW \neq ND,$$

their trajectory geometry can be compared through dimension-independent directional statistics. For a temporal offset ℓ, we compute

$$s^x_{t,\ell} = \frac{\Delta\mathbf{x}_t^\top \Delta\mathbf{x}_{t+\ell}}{\|\Delta\mathbf{x}_t\|_2 \|\Delta\mathbf{x}_{t+\ell}\|_2},$$

and

$$s^q_{t,\ell} = \frac{\Delta\mathbf{q}_t^\top \Delta\mathbf{q}_{t+\ell}}{\|\Delta\mathbf{q}_t\|_2 \|\Delta\mathbf{q}_{t+\ell}\|_2}.$$

Both quantities are scalar directional similarities, allowing the trajectory geometry of the two spaces to be directly aligned. The geometry loss aggregates the discrepancy across a set of temporal offsets $\mathcal{S}$,

$$\mathcal{L}_{\text{geo}} = \sum_{\ell \in \mathcal{S}} w_\ell \mathcal{D}_{\text{geo}} \left(s^q_{t,\ell}, s^x_{t,\ell} \right),$$

where w_ℓ controls the contribution of each temporal scale.

To prevent the projector from unnecessarily altering the predictive representation, we additionally constrain the projected states to remain close to the pretrained latent states,

$$\mathcal{L}_{\text{anchor}} = \mathcal{D}_{\text{anchor}}(\mathbf{Q}, \mathbf{Z}).$$

The geometry-alignment stage optimizes only the projector parameters while keeping the pretrained encoder fixed.

Auxiliary conditional causal predictor. In addition to geometric alignment, we introduce an auxiliary causal predictor to encourage the projected space to remain compatible with parameter-dependent dynamics. Given the projected latent history and the physical parameter $\boldsymbol{\xi}$, the predictor estimates the next latent displacement,

$$\widehat{\Delta \mathbf{q}}_t = P_\psi \left(\mathbf{q}, \boldsymbol{\xi} \right),$$

where $\boldsymbol{\xi}$ is embedded as a conditioning token and supplied to the causal predictor. Importantly, the geometry projector itself is parameter-independent and takes only the pretrained latent representation as input; parameter information enters exclusively through the auxiliary dynamics predictor.

The corresponding prediction objective is

$$\mathcal{L}_{dyn} = \mathcal{D}_{\text{pred}} \left(\widehat{\Delta \mathbf{q}}_t, \Delta \mathbf{q}_t \right),$$

where

$$\Delta \mathbf{q}_t = \mathbf{q}_{t+1} - \mathbf{q}_t.$$

Together with the geometry and anchor objectives, the alignment-stage loss is

$$\mathcal{L}_{\text{align}} = \mathcal{L}_{dyn} + \lambda_{\text{geo}} \mathcal{L}_{\text{geo}} + \lambda_{\text{anchor}} \mathcal{L}_{\text{anchor}}.$$

The pretrained encoder remains frozen, while the geometry projector and auxiliary predictor are jointly optimized during this stage.

B.3 Predictor Learning Stage

After geometry alignment, physical evolution is modeled directly in the aligned latent state space. Given a projected latent state

$$\mathbf{q} \in \mathbb{R}^{N \times D}$$

Physics-structured latent predictor. The latent transition is decomposed into a parameter-independent evolution component and a parameter-dependent response component. we model its evolution through a parameter-structured latent vector field,

$$\dot{q}_t = \mathcal{F}_\theta(q_t, \xi) = \mathcal{F}_{\text{evo}}(q_t) + \mathcal{F}_{\text{par}}(q_t, \xi),$$

The evolution branch

$$\mathcal{F}_{\text{evo}} : \mathbb{R}^{N\times D} \to \mathbb{R}^{N\times D}$$

models dynamics shared across different governing conditions. The parameter-dependent branch

$$\mathcal{F}_{\text{par}} : \mathbb{R}^{N\times D} \times \mathbb{R}^{D_\xi} \to \mathbb{R}^{N\times D}$$

captures the change in latent evolution induced by the governing parameters.

Both branches preserve the spatial-token structure of the representation, so that their outputs can be directly combined with the current latent state. For continuous-time predictors, the next latent state is obtained by numerically integrating the vector field over one observation interval,

$$\hat{q}_{t+1} = \Phi_{\text{ODE}}(q_t, \mathcal{F}_\theta, \xi, \Delta t).$$

Autoregressive rollout. For long-horizon prediction, the predicted latent state is recursively used as the input to the next transition,

$$\widehat{\mathbf{q}}_{t+k} = \mathcal{P}\left(\widehat{\mathbf{q}}_{t+k-1}, \boldsymbol{\xi}\right),$$

where $\mathcal{P}$ denotes the complete latent transition operator.

The dynamics are therefore evolved entirely in the learned latent space. Physical predictions are recovered only when needed through a reconstruction head

$$\mathcal{D}_\omega : \mathbb{R}^{N\times D} \to \mathbb{R}^{C\times H\times W}.$$

This avoids repeatedly mapping autoregressive predictions between the physical and latent domains during rollout.

B.4 Decoder Learning Stage

The decoder is trained after the geometry projector and latent predictor have been learned. During this stage, the encoder, geometry projector, and latent predictor are all frozen, and only the decoder parameters are optimized.

Rollout-based latent generation. Given a physical trajectory

$$\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{T-1}] \in \mathbb{R}^{B\times T\times C\times H\times W},$$

only the initial physical state $\mathbf{x}_0$ is used to construct the latent input. It is first mapped by the frozen encoder and geometry projector as

$$\mathbf{z}_0 = f_\theta(\mathbf{x}_0) \in \mathbb{R}^{B\times N\times D},$$

$$\mathbf{q}_0 = \mathbf{z}_0 + g_\phi(\mathbf{z}_0) \in \mathbb{R}^{B\times N\times D}.$$

Starting from q_0, the frozen physics-structured latent predictor recursively generates the remaining latent states by integrating the learned latent vector field:

$$\hat{q}_{t+1} = \Phi_{\text{ODE}}(\hat{q}_t, F_\theta, \xi, \Delta t), \qquad \hat{q}_0 = q_0.$$

In our implementation, Φ_{ODE} is instantiated using fourth-order Runge–Kutta integration with four substeps.

This produces the complete latent rollout

$$\widehat{\mathbf{Q}} = [\mathbf{q}_0, \widehat{\mathbf{q}}_1, \ldots, \widehat{\mathbf{q}}_{T-1}] \in \mathbb{R}^{B\times T\times N\times D}.$$

Thus, future ground-truth states $\mathbf{x}_1, \ldots, \mathbf{x}_{T-1}$ are never encoded to construct the decoder input. They are used only as reconstruction targets. This exposes the decoder during training to the same autoregressive latent distribution encountered at inference time, including deviations accumulated during latent rollout.

Decoder. Before decoding, the temporal and spatial-token dimensions of the rollout are combined,

$$\widehat{\mathbf{Q}} \in \mathbb{R}^{B\times T\times N\times D} \to \mathbb{R}^{B\times TN\times D}.$$

The latent tokens are first normalized and projected from the latent dimension D to an initial convolutional feature dimension D_0. Since each latent state contains a two-dimensional token grid with

$$N = N_h N_w,$$

the token sequence is rearranged into frame-wise spatial feature maps,

$$\mathbb{R}^{B\times TN\times D_0} \to \mathbb{R}^{(BT)\times D_0\times N_h\times N_w}.$$

The decoder then progressively reconstructs the physical spatial resolution. Starting from the coarse latent token grid, residual convolutional blocks refine the features, followed by a sequence of spatial upsampling stages,

$$\widehat{\mathbf{Q}}^{(s+1)} = \mathcal{R}^{(s)}\left(\mathrm{Up}_2\left(\widehat{\mathbf{Q}}^{(s)}\right)\right),$$

where $\mathrm{Up}_2(\cdot)$ denotes a factor-of-two spatial interpolation and $\mathcal{R}^{(s)}$ denotes the residual convolutional refinement at stage s. After S stages, the feature resolution is restored from

$$N_h \times N_w$$

to

$$H \times W.$$

A final convolutional head maps the reconstructed features to the C physical channels. The frame dimension is then restored, yielding

$$\widehat{\mathbf{X}} = \mathcal{D}_\omega\left(\widehat{\mathbf{Q}}\right) \in \mathbb{R}^{B\times T\times C\times H\times W}.$$

Although the full latent rollout is provided jointly as the decoder input tensor, the spatial reconstruction is performed frame-wise: the temporal dimension is folded into the batch dimension before the convolutional decoding blocks. Consequently, the decoder itself does not model temporal evolution.

Training objective. The complete ground-truth physical trajectory serves as the reconstruction target,

$$\mathcal{L}_{\mathrm{dec}} = \mathcal{D}_{\mathrm{phy}}\left(\widehat{\mathbf{X}}, \mathbf{X}\right).$$

During this stage $f_\theta, g_\phi, \mathcal{P}_\psi$remain frozen, and gradients are propagated only through $\mathcal{D}_\omega$.

C Implementation Details

All codes are written in Pytorch (Paszke et al., 2019). All experiments are conducted on 4 RTX PRO 6000 Blackwell 96G with approximately total of 9000 GPU hours.

C.1 PDE-JEPA Implementation

The hyperparameter settings for PDE-JEPA's architecture across all datasets are summarized in Table 8. All stages use AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$, except structured predictors, which use $(0.9, 0.95)$.

C.1.1 Physics-Structured Latent Predictor Implementations

Following the general formulation in Section 4.3, the predictor first maps the current latent state through a shared backbone,

$$\mathbf{H}_\theta(\mathbf{q}) = \mathcal{H}_\theta(\mathbf{q}),$$

and applies independent component heads to the shared features,

$$C_\theta(\mathbf{q}) = h_{C,\theta}(\mathbf{H}_\theta(\mathbf{q}))\,, \qquad D_\theta^{(j)}(\mathbf{q}) = h_{j,\theta}(\mathbf{H}_\theta(\mathbf{q}))\,, \quad j = 1, \dots, M.$$

Each component has the same spatial and channel dimensions as the latent state, i.e.,

$$C_\theta(\mathbf{q}),\, D_\theta^{(j)}(\mathbf{q}) \in \mathbb{R}^{N \times D}.$$

The latent vector field is then parameterized as

$$\dot{\mathbf{q}} = C_\theta(\mathbf{q}) + \sum_{j=1}^{M} r_j(\boldsymbol{\xi})\, D_\theta^{(j)}(\mathbf{q}),$$

where $r_j(\boldsymbol{\xi})$ denotes the normalized coefficient associated with the j-th parameter-dependent component. We instantiate this general form according to the governing structure of each PDE.

For a training set of N_{tr} trajectories, we denote

$$\langle g(a) \rangle_{\mathrm{tr}} = \frac{1}{N_{\mathrm{tr}}} \sum_{i=1}^{N_{\mathrm{tr}}} g(a_i).$$

Vorticity. For Vorticity, the latent vector field is

$$\dot{\mathbf{q}} = \mathbf{C}_\theta(\mathbf{q}) + r_\nu \mathbf{D}_\theta^\nu(\mathbf{q}), \qquad r_\nu = \frac{\nu - \nu_0}{s_\nu},$$

where

$$\nu_0 = \frac{\nu_{\min}^{\mathrm{tr}} + \nu_{\max}^{\mathrm{tr}}}{2}, \qquad s_\nu = \frac{\nu_{\max}^{\mathrm{tr}} - \nu_{\min}^{\mathrm{tr}}}{2}.$$

Here $\nu_{\min}^{\mathrm{tr}}$ and $\nu_{\max}^{\mathrm{tr}}$ denote the minimum and maximum viscosities in the ID training split.

Wave-2D. For Wave-2D, we use

$$\dot{\mathbf{q}} = \mathbf{C}_\theta(\mathbf{q}) + r_{c^2} \mathbf{D}_\theta^c(\mathbf{q}) - r_k \mathbf{D}_\theta^k(\mathbf{q}),$$

where

$$r_{c^2} = \frac{c^2}{s_{c^2}}, \qquad r_k = \frac{k}{s_k^{\mathrm{Wave}}},$$

with

$$s_{c^2} = \sqrt{\langle c^4 \rangle_{\mathrm{tr}}}, \qquad s_k^{\mathrm{Wave}} = \sqrt{\langle k^2 \rangle_{\mathrm{tr}}}.$$

Gray–Scott. For Gray–Scott, the predictor is

$$\dot{\mathbf{q}} = \mathbf{C}_\theta(\mathbf{q}) + \frac{F}{s_F}\mathbf{D}_\theta^F(\mathbf{q}) + \frac{k}{s_k^{\mathrm{GS}}}\mathbf{D}_\theta^k(\mathbf{q}),$$

where

$$s_F = \langle |F| \rangle_{\mathrm{tr}}\,, \qquad s_k^{\mathrm{GS}} = \langle |k| \rangle_{\mathrm{tr}}\,.$$

Combined Equation. For the combined equation, we associate separate latent vector fields with the transport, diffusion, and dispersion coefficients:

$$\dot{\mathbf{q}} = -\frac{\alpha}{s_\alpha}\mathbf{D}_\theta^\alpha(\mathbf{q}) + \frac{\beta}{s_\beta}\mathbf{D}_\theta^\beta(\mathbf{q}) - \frac{\gamma}{s_\gamma}\mathbf{D}_\theta^\gamma(\mathbf{q}),$$

where the coefficient scales are

$$s_p = \sqrt{\langle p^2 \rangle_{\mathrm{tr}}}, \qquad p \in \{\alpha, \beta, \gamma\}.$$

HeterNS. For HeterNS, viscosity and the spatial forcing field are modeled separately:

$$\dot{\mathbf{q}} = \mathbf{C}_\theta(\mathbf{q}) + r_\nu \mathbf{D}_\theta^\nu(\mathbf{q}) + m\mathbf{D}_\theta^m(\mathbf{q}),$$

where

$$r_\nu = \frac{\nu - \nu_0}{s_\nu},$$

and

$$\nu_0 = \frac{\nu_{\min}^{\mathrm{tr}} + \nu_{\max}^{\mathrm{tr}}}{2}, \qquad s_\nu = \frac{\nu_{\max}^{\mathrm{tr}} - \nu_{\min}^{\mathrm{tr}}}{2}.$$

C.1.2 Latent time integration.

For predictors formulated as continuous latent vector fields, we obtain the next latent state by numerically integrating

$$\dot{\mathbf{q}} = \mathcal{F}_\theta(\mathbf{q}, \boldsymbol{\xi}),$$

where $\mathcal{F}_\theta$ denotes the corresponding physics-structured vector field defined above. Given the current latent state $\mathbf{q}_t$ and the physical interval Δt between two consecutive frames, we use the classical fourth-order Runge–Kutta (RK4) scheme with M substeps. Let $h = \Delta t/M$ and $\mathbf{q}_t^{(0)} = \mathbf{q}_t$. For each substep $m = 0, \ldots, M-1$,

$$\begin{aligned}
\mathbf{k}_1 &= \mathcal{F}_\theta\left(\mathbf{q}_t^{(m)}, \boldsymbol{\xi}\right), \\
\mathbf{k}_2 &= \mathcal{F}_\theta\left(\mathbf{q}_t^{(m)} + \frac{h}{2}\mathbf{k}_1, \boldsymbol{\xi}\right), \\
\mathbf{k}_3 &= \mathcal{F}_\theta\left(\mathbf{q}_t^{(m)} + \frac{h}{2}\mathbf{k}_2, \boldsymbol{\xi}\right), \\
\mathbf{k}_4 &= \mathcal{F}_\theta\left(\mathbf{q}_t^{(m)} + h\mathbf{k}_3, \boldsymbol{\xi}\right),
\end{aligned}$$

and the latent state is updated as

$$\mathbf{q}_t^{(m+1)} = \mathbf{q}_t^{(m)} + \frac{h}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right).$$

After M substeps, the prediction for the next frame is

$$\widehat{\mathbf{q}}_{t+1} = \mathbf{q}_t^{(M)}.$$

We use $M = 4$ RK4 substeps for each latent transition.

Table 8: **Implementation and training hyperparameters of PDE-JEPA.**

Hyperparameter	Combined	Advection	Burgers	Heat	Wave-B	GS	Wave2D	Vorticity	HeterNS
Model input grid	256	256	256	256	256	32×32	64×64	128×128	64×64
Sampled frames / stride	14 / 10	14 / 5	25 / 10	25 / 10	25 / 10	20 / 1	30 / 1	30 / 1	20 / 1
Channels / patch size	1/4	1/4	1/4	1/4	1/4	2/4×4	2/4×4	1/8×8	1/4×4
Latent dimension	192	192	192	192	192	192	192	192	192
Stage 1: Pretraining									
Encoder depth	12	12	12	12	12	24	12	12	12
Predictor depth	12	12	12	12	12	24	12	12	12
Predictor hidden size	384	384	384	384	384	384	384	384	384
Encoder attention heads	3	3	3	3	3	3	3	3	3
Predictor attention heads	12	12	12	12	12	12	12	12	12
MLP ratio	4	4	4	4	4	4	4	4	4
Dropout	0	0	0	0	0	0	0	0	0
Normalization type	LayerNorm	LayerNorm	LayerNorm	LayerNorm	LayerNorm	LayerNorm	LayerNorm	LayerNorm	LayerNorm
Activation	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU
Positional encoding	RoPE	RoPE	RoPE	RoPE	RoPE	RoPE	RoPE	RoPE	RoPE
Epoch	15 + 20	15 + 30	20 + 50	20 + 50	20 + 50	20 + 50	20 + 50	20 + 50	20 + 50
Global batch	32	32	32	32	32	32	32	32	32
Peak LR	3.5×10^{-4}	3.5×10^{-4}	3.5×10^{-4}	3.5×10^{-4}	3.5×10^{-4}	3.5×10^{-4}	3.5×10^{-4}	3.5×10^{-4}	3.5×10^{-4}
Final LR	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
Weight decay	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
Warmup epochs	2 / 0	2 / 0	2 / 0	2 / 0	2 / 0	2 / 0	2 / 0	2 / 0	2 / 0
EMA momentum	0.99925	0.99925	0.99925	0.99925	0.99925	0.99925	0.99925	0.99925	0.99925
Spatial mask scales	.15 / .70	.12 / .60	.15 / .70	.15 / .70	.15 / .70	.15 / .70	.12 / .60	.15 / .70	.15 / .70
Stage 2: Geometry Alignment									
Projector MLP layers	2	2	2	2	2	2	2	2	2
Projector hidden width	384	384	384	384	384	384	384	384	384
Projector residual	True	True	True	True	True	True	True	True	True
Zero output initialization	True	True	True	True	True	True	True	True	True
Causal predictor depth	24	24	24	24	24	24	24	24	24
Causal predictor width	384	384	384	384	384	384	384	384	384
Causal predictor head	12	12	12	12	12	12	12	12	12
Normalization	LN	LN	LN	LN	LN	LN	LN	LN	LN
Activation	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU
Epoch	50	50	50	50	50	50	50	50	50
Global batch	16	16	16	16	16	16	16	16	16
Peak LR	5×10^{-5}	5×10^{-5}	5×10^{-5}	5×10^{-5}	5×10^{-5}	5×10^{-5}	5×10^{-5}	5×10^{-5}	5×10^{-5}
Final LR	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
Weight decay	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
Warmup epochs	5	5	5	5	5	5	5	5	5
Loss weights $(\lambda_{\text{geom}}, \lambda_{\text{anchor}})$	.1 / .01	.1 / .01	.1 / .01	.1 / .01	.1 / .01	.1 / .01	.1 / .01	.1 / .01	.1 / .01
Stage 3: Predictor Learning									
Dynamics / integrator	RK4 ×4	RK4 ×4	RK4 ×4	RK4 ×4	RK4 ×4	RK4 ×4	RK4 ×4	RK4 ×4	RK4 ×4
Predictor depth	24	24	24	24	24	8	24	24	24
Hidden size	384	384	384	384	384	384	384	384	384
Attention heads	12	12	12	12	12	12	12	12	12
MLP ratio	4	4	4	4	3.90104	4	3.85938	4	4
Feed-forward width	1536	1536	1536	1536	1498	1536	1482	1536	1536
Normalization	LN	LN	LN	LN	LN	LN	LN	LN	LN
Activation	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU
Dropout	0	0	0	0	0	0	0	0	0
Positional encoding	RoPE	RoPE	RoPE	RoPE	RoPE	sin/cos	RoPE	sin/cos	RoPE
Dynamics state dimension	192	192	192	192	192	192	192	192	192
Condition dimension	3	1	7	7	4	2	2	1	2
Epoch	50	50	50	50	50	50	50	50	100
Global batch	16	16	16	16	64	64	64	64	64
Peak LR	5×10^{-5}	5×10^{-5}	5×10^{-5}	5×10^{-5}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}
Final LR	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
Weight decay	0.04	0.04	0.04	0.04	0.05	0.05	0.05	0.05	0.05
Warmup	5	5	5	5	5	5	5	5	5
Stage 4: Decoder									
Latent input	Rollout 204/148	Rollout 128/96	Rollout 192/128	Rollout 192/128	Rollout 248/168	Rollout 192/128	Rollout 192/128	Rollout 384/256	Rollout 424/288
Convolutional channel widths	103/54	64/32	96	96	112	64	96/64	192/128	112
Residual blocks	1	1	1	1	2	1	1	1	2
Normalization type	LN + GN(8)	LN + GN(8)	LN + GN(8)	LN + GN(8)	LN + GN(8)	LN + GN(8)	LN + GN(8)	LN + GN(8)	LN + GN(8)
Activation	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU
Spatial scale factors	2/2	2/2	2/2	2/2	2/2	2/2	1/2/2	2/2/2	2/2
Spatial padding	Circular	Circular	Circular	Circular	Zeros	Circular	Reflect	Circular	Circular
Reconstructed channels	1	1	1	1	1	2	2	1	1
Epoch	2000	2000	200	200	200	200	200	200	1000

Continued on next page

Table 8 – continued from previous page

Hyperparameter	Combined	Advection	Burgers	Heat	Wave-B	GS	Wave2D	Vorticity	HeterNS
Global batch	96	64	16	16	32	16	32	16	32
Peak LR	2×10^{-4}	10^{-4}	10^{-4}	10^{-4}	2×10^{-4}	10^{-4}	10^{-4}	10^{-4}	3×10^{-4}
Final LR	10^{-5}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	0
Weight decay	10^{-4}	0.04	0.04	0.04	10^{-4}	0.04	0.04	0.04	0.01
Warmup epochs	50	50	10	10	10	10	10	10	50

C.2 Baseline Implementations

We provide the implementation details of the baselines used in our experiments. We compare our method against FNO (Li et al., 2020), CAPE (Takamoto et al., 2023), CoDA (Kirchmeyer et al., 2022), GEPS (Kassaï Koupaï et al., 2024), ViT-*in-context*, [CLS] ViT (Peebles & Xie, 2023), Zebra (Serrano et al., 2024a), UniSolver (Zhou et al., 2024), MPP (McCabe et al., 2024), DPOT-S (Hao et al., 2024), Poseidon-T (Herde et al., 2024), LE-PDE (Wu et al., 2022), LNS (Li et al., 2025), MAE-PDE (Zhou & Farimani, 2024), and ENMA (Kassaï Koupaï et al., 2026). For CAPE, CoDA, Zebra, [CLS] ViT, and ViT-*in-context*, we directly adopt the results reported in Zebra (Serrano et al., 2024a). All remaining baselines are implemented and evaluated under our experimental setting, and their implementation details are described below. All reimplemented baselines use the same evaluation trajectories / metric / rollout protocol.

FNO. We concatenated the temporal history along the channel dimension and appended spatial coordinates to the input. We stacked four Fourier layers, each combining a spectral convolution with a pointwise convolution and GELU activation. The 1D models used 32 Fourier modes and a hidden width of 195, whereas the 2D models used 12 modes per spatial axis and a width of 66. We used a two-frame history for the all datasets and generated trajectories autoregressively. The configuration is in Table 9.

Table 9: **FNO implementation hyperparameters.**

Hyperparameter	Combined	Advection	Burgers	Heat	Wave-B	GS	Wave2D	Vorticity	HeterNS
Model input grid	256	256	256	256	256	32×32	64×64	128×128	64×64
State channels	1	1	1	1	1	2	2	1	1
Fourier layers	4	4	4	4	4	4	4	4	4
Hidden width	195	195	195	195	195	66	66	66	66
Modes per spatial axis	32	32	32	32	32	12	12	12	12
Input history (frames)	2	2	2	2	2	2	2	2	2
Layer normalization	False	False	False	False	False	False	False	False	False
Input instance normalization	True	True	True	True	True	True	True	True	True
Activation	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU
Coordinate features	Grid	Grid	Grid	Grid	Grid	Grid	Grid	Grid	Grid
Temporal stride	10	10	10	10	10	1	1	1	1
Training epochs (budget)	500	500	500	500	500	500	500	500	500
Batch size / GPU	32	32	32	32	32	32	32	16	32
Training rollout steps	1	1	1	1	1	1	2	1	1
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Peak learning rate	0.001	0.001	0.001	0.001	0.001	0.001	10^{-4}	0.001	0.001
Weight decay	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
LR schedule	OneCycle	OneCycle	OneCycle	OneCycle	OneCycle	OneCycle	OneCycle	OneCycle	OneCycle
Warmup epochs	40	40	40	40	40	40	20	40	40
Gradient clipping	10000	10000	10000	10000	10000	10000	1	10000	10000

Adam uses $(\beta_1, \beta_2) = (0.9, 0.9)$; both OneCycle division factors are 10^4.

GEPS. We used an FNO backbone whose weights and activations were modulated by a learned four-dimensional environment code. The 1D models contained eight Fourier layers with width 64 and 16 modes; the 2D models contained four Fourier layers with width 16 and 12 modes per spatial axis. The local convolution branches used kernel sizes of 1 in 1D and 7×7 in 2D. We included spatial coordinates and used code-conditioned Swish activations. At evaluation, the environment code was adapted using a support trajectory while the shared network parameters remained frozen. The configuration is in Table 10.

Table 10: **GEPS implementation hyperparameters.**

Hyperparameter	Combined	Advection	Burgers	Heat	Wave-B	GS	Wave2D	Vorticity	HeterNS
Model input grid	128	128	128	128	128	32×32	64×64	64×64	64×64
State channels	1	1	1	1	1	2	2	1	1
Fourier layers	8	8	8	8	8	4	4	4	4
Hidden width	64	64	64	64	64	16	16	16	16
Modes per spatial axis	16	16	16	16	16	12	12	12	12
Local convolution kernel	1	1	1	1	1	7	7	7	7
Environment code size	4	4	4	4	4	4	4	4	4
Adaptation factor	1	1	1	1	1	1	1	1	1
Activation	Swish	Swish	Swish	Swish	Swish	Swish	Swish	Swish	Swish
Coordinate features	Grid	Grid	Grid	Grid	Grid	Grid	Grid	Grid	Grid
Model input frames	1	1	1	1	1	1	1	1	1
Training epochs	20000	20000	20000	20000	20000	20000	20000	20000	20000
Batch size	64	64	64	64	64	64	64	64	256
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Shared-weight LR	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
Environment-code LR	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Min. learning rate	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}
LR schedule	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine
Rollout curriculum (steps)	1/2/4	1/2/4	1/2/4	1/2/4	1/2/4	1/2/4	1/2/4	1/2/4	1/2/4
Gradient clipping	1	1	1	1	1	1	1	1	1
Adaptation epochs	500	500	500	500	500	500	500	500	500
Adaptation batch size	32	32	12	12	32	32	32	120	64
Adaptation learning rate	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Adaptation rollout steps	4	4	4	4	4	4	4	4	4

Adam uses $(\beta_1, \beta_2) = (0.9, 0.9)$.

Unisolver. We used a PDE-conditioned transformer with ten blocks, a hidden size of 256, eight attention heads, and a feed-forward width of 1008. Domain-level PDE parameters and pointwise geometric features modulated the transformer through separate AdaLN-Zero conditioning branches. Geometry was represented by distances to a reference grid. We used patches of 16 points for 1D inputs, 2×2 patches for Gray–Scott, and 4×4 patches for Wave2D and Vorticity. Given the current field, the model predicted a residual update that was added to the input state for autoregressive forecasting. The configuration is in Table 11.

Table 11: **UniSolver implementation hyperparameters.**

Hyperparameter	Combined	Advection	Burgers	Heat	Wave-B	GS	Wave2D	Vorticity
Model input grid	128	128	128	128	128	32×32	64×64	64×64
Spatial patch size	16	16	16	16	16	2×2	4×4	4×4
State channels	1	1	1	1	1	2	2	1
Transformer depth	10	10	10	10	10	10	10	10
Hidden size	256	256	256	256	256	256	256	256
MLP ratio	3.9375	3.9375	3.9375	3.9375	3.9375	3.9375	3.9375	3.9375
Attention heads	8	8	8	8	8	8	8	8
Dropout	0	0	0	0	0	0	0	0
Block normalization	AdaLN	AdaLN	AdaLN	AdaLN	AdaLN	AdaLN	AdaLN	AdaLN
MLP activation	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU
Position / geometry	Ref. dist.	Ref. dist.	Ref. dist.	Ref. dist.	Ref. dist.	Ref. dist.	Ref. dist.	Ref. dist.
Reference grid	16	16	16	16	16	8×8	4×4	4×4
Input history (frames)	1	1	1	1	1	1	1	1
Residual prediction	True	True	True	True	True	True	True	True
Training epochs	100	100	100	100	100	100	100	100
Batch size / GPU	64	64	64	64	64	64	64	64
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
Peak learning rate	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}
Weight decay	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}
LR schedule	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine
Warmup epochs	5	5	5	5	5	5	5	5
Gradient clipping	1	1	1	1	1	1	1	1

MPP. We used the AViT-B (Müller et al., 2022) architecture with twelve blocks that alternated temporal attention and axial spatial attention. The model used a hidden size of 768, twelve attention heads, an MLP ratio of 4, and 16×16 spatial patches. A learned projection embedded the physical

state variables, and a patch decoder reconstructed the next field from up to sixteen history frames. For 1D inputs, we repeated the field along a synthetic axis of length 16 to preserve the 2D patch architecture. We used relative positional biases, LayerNorm on queries and keys, and a maximum drop-path rate of 0.1. All reported runs were trained from scratch. We use DAdaptAdan, a family of Adan (Defazio & Mishchenko, 2023) as optimizer. The configuration is in Table 12.

Table 12: MPP implementation hyperparameters.

Hyperparameter	Combined	Advection	Burgers	Heat	Wave-B	GS	Wave2D	Vorticity	HeterNS
Spatial patch size	16	16	16	16	16	16×16	16×16	16×16	16×16
Space–time blocks	12	12	12	12	12	12	12	12	12
Hidden size	768	768	768	768	768	768	768	768	768
MLP ratio	4	4	4	4	4	4	4	4	4
Attention heads	12	12	12	12	12	12	12	12	12
Attention dropout	0	0	0	0	0	0	0	0	0
Max. drop-path rate	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
QK normalization	LN	LN	LN	LN	LN	LN	LN	LN	LN
Spatial norm denominator	std-div.	std-div.	std-div.	std-div.	std-div.	RMS	std-div.	std-div.	std-div.
Temporal normalization	IN	IN	IN	IN	IN	IN	IN	IN	IN
Activation	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU
Positional bias	Relative	Relative	Relative	Relative	Relative	Relative	Relative	Relative	Relative
Max. input history	16	16	16	16	16	16	16	16	16
State channels	1	1	1	1	1	2	2	1	1
Initialization	Scratch	Scratch	Scratch	Scratch	Scratch	Scratch	Scratch	Scratch	Scratch
Training epochs	500	500	500	500	500	500	500	500	500
Batch size / process	16	16	16	16	16	16	16	8	16
Global batch size	16	16	16	16	16	16	16	16	16
Steps per epoch	125	125	125	125	125	125	125	125	125
Gradient accumulation	1	1	1	1	1	1	1	1	1
Optimizer	DAdaptAdan	DAdaptAdan	DAdaptAdan	DAdaptAdan	DAdaptAdan	DAdaptAdan	DAdaptAdan	DAdaptAdan	DAdaptAdan
LR multiplier	1	1	1	1	1	1	1	1	1
Weight decay	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
LR schedule	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine
Applied warmup steps	50	50	50	50	50	50	50	50	50

LN denotes LayerNorm and IN denotes InstanceNorm.

DPOT-S. We fine-tuned the pretrained DPOT-S backbone on the target PDE datasets. We retained six adaptive Fourier neural operator blocks with a hidden size of 1024, eight Fourier channel groups, and an MLP ratio of 1. Inputs were represented on a 128×128 grid and embedded using 8×8 spatial patches. Learned positional embeddings and temporal aggregation combined the ten-frame input buffer before Fourier processing. The model used GroupNorm and GELU, followed by a convolutional upsampling head, and was fine-tuned using the target solution trajectories. The configuration is in Table 13.

Table 13: **DPOT implementation hyperparameters.**

Hyperparameter	Combined	Advection	Burgers	Heat	Wave-B	GS	Wave2D	Vorticity	HeterNS
Model input grid	128×128	128×128	128×128	128×128	128×128	128×128	128×128	128×128	128×128
Spatial patch size	8×8	8×8	8×8	8×8	8×8	8×8	8×8	8×8	8×8
AFNO depth	6	6	6	6	6	6	6	6	6
Hidden size	1024	1024	1024	1024	1024	1024	1024	1024	1024
MLP ratio	1	1	1	1	1	1	1	1	1
Fourier blocks	8	8	8	8	8	8	8	8	8
Fourier modes (configured)	32	32	32	32	32	32	32	32	32
Padded state channels	4	4	4	4	4	4	4	4	4
History buffer (frames)	10	10	10	10	10	10	10	10	10
Initial true frames	1	1	1	1	1	1	1	1	1
Block normalization	GN	GN	GN	GN	GN	GN	GN	GN	GN
Normalization groups	8	8	8	8	8	8	8	8	8
Activation	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU
Positional embedding	Learned	Learned	Learned	Learned	Learned	Learned	Learned	Learned	Learned
Initialization	DPOT-S	DPOT-S	DPOT-S	DPOT-S	DPOT-S	DPOT-S	DPOT-S	DPOT-S	DPOT-S
Training epochs	200	200	200	200	200	200	200	200	200
Batch size / GPU	16	16	16	16	16	16	16	16	16
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Peak learning rate	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
Weight decay	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
LR schedule	OneCycle	OneCycle	OneCycle	OneCycle	OneCycle	OneCycle	OneCycle	OneCycle	OneCycle

Continued on next page

Table 13 – continued from previous page

Hyperparameter	Combined	Advection	Burgers	Heat	Wave-B	GS	Wave2D	Vorticity	HeterNS
Warmup epochs	40	40	40	40	40	40	40	40	40
Gradient clipping	10000	10000	10000	10000	10000	10000	10000	10000	10000

DPOT-S is initialized from the saved pretrained checkpoint `model_S.pth`. All tasks are adapted to a 128×128 input with four state slots; 1D signals are lifted to the 2D input representation. The ten-frame network buffer is initialized by repeating the single available initial frame. The eight Fourier blocks are AFNO (Guibas et al., 2021) channel groups, not attention heads. The nominal mode count is 32. Adam uses $(0.9, 0.9)$, and both OneCycle division factors are 10^4.

Poseidon-T. We initialized the ScOT encoder–decoder from the pretrained Poseidon-T checkpoint and fine-tuned it on the target PDE datasets. The model used four resolution stages in each branch, with four transformer blocks per stage and skip connections between corresponding stages. The stage widths H was $48, 96, 192, 384$, with $3, 6, 12, 24$ attention heads, respectively. We retained 4×4 patches, an MLP ratio of 4, shifted-window attention with a configured window size of 16, and time-conditioned LayerNorm. Input and output projections were adapted to the required channels, and the transformer backbone remained trainable during fine-tuning. The configuration is in Table 14.

Table 14: **Poseidon implementation hyperparameters.**

Hyperparameter	Combined	Advection	Burgers	Heat	Wave-B	GS	Wave2D	Vorticity	HeterNS
Model input grid	64×64	64×64	64×64	64×64	64×64	64×64	64×64	128×128	64×64
Spatial patch size	4×4	4×4	4×4	4×4	4×4	4×4	4×4	4×4	4×4
Encoder / decoder stages	4 / 4	4 / 4	4 / 4	4 / 4	4 / 4	4 / 4	4 / 4	4 / 4	4 / 4
Depths per branch	4/4/4/4	4/4/4/4	4/4/4/4	4/4/4/4	4/4/4/4	4/4/4/4	4/4/4/4	4/4/4/4	4/4/4/4
Stage hidden sizes	H	H	H	H	H	H	H	H	H
Stage attention heads	3/6/12/24	3/6/12/24	3/6/12/24	3/6/12/24	3/6/12/24	3/6/12/24	3/6/12/24	3/6/12/24	3/6/12/24
MLP ratio	4	4	4	4	4	4	4	4	4
Window size (configured)	16	16	16	16	16	16	16	16	16
Input channels (incl. PDE)	4	2	3	3	6	4	4	2	3
Output channels	1	1	1	1	1	2	2	1	1
Attention / hidden dropout	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
Drop-path rate	0	0	0	0	0	0	0	0	0
QK normalization	L2 cosine	L2 cosine	L2 cosine	L2 cosine	L2 cosine	L2 cosine	L2 cosine	L2 cosine	L2 cosine
Block normalization	Cond. LN	Cond. LN	Cond. LN	Cond. LN	Cond. LN	Cond. LN	Cond. LN	Cond. LN	Cond. LN
Activation	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU
Positional bias	Cont. rel.	Cont. rel.	Cont. rel.	Cont. rel.	Cont. rel.	Cont. rel.	Cont. rel.	Cont. rel.	Cont. rel.
Initialization	Poseidon-T	Poseidon-T	Poseidon-T	Poseidon-T	Poseidon-T	Poseidon-T	Poseidon-T	Poseidon-T	Poseidon-T
Training epochs	100	100	100	100	100	100	100	100	100
Batch size	64	64	64	64	64	64	64	64	64
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
Backbone / head LR	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
Weight decay	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
LR schedule	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine
Warmup fraction	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
Gradient clipping	5	5	5	5	5	5	5	5	5

All models are initialized from the pretrained Poseidon-T checkpoint and fine-tuned on the corresponding target PDE dataset. L2 cosine denotes query/key unit normalization in SwinV2 attention (Liu et al., 2022). Cont. rel. denotes continuous relative positional bias, with absolute positional embeddings disabled. Cond. LN denotes time-conditioned LayerNorm. The same learning rate is used for the backbone and task-specific input/output projections, and the transformer backbone remains trainable during fine-tuning. The 256-point 1D signals are arranged on a 16×16 serpentine grid and resized to 64×64 before being passed to the model.

LE-PDE. We used a convolutional autoencoder with four downsampling blocks, GroupNorm as GN, and ELU activations (Clevert et al., 2015). The dynamic encoder maps the input history to a 256-dimensional latent, while a context encoder extracts a 64-dimensional static code from another trajectory in the same environment. Their concatenation is evolved by a five-layer residual MLP with hidden width 256, and the decoder reconstructs the physical field. The configuration is given in Table 15.

Table 15: **LE-PDE implementation hyperparameters.**

Hyperparameter	Combined	Advection	Burgers	Heat	Wave-B	GS	Wave2D	Vorticity	HeterNS
Model input grid	256	256	256	256	256	32×32	64×64	64×64	64×64
State channels	1	1	1	1	1	2	2	1	1
Input history (frames)	1	1	1	1	1	1	1	1	10
Context trajectory frames	10	10	10	10	15	7	10	10	20
Encoder downsampling blocks	4	4	4	4	4	4	4	4	4
Base convolution width	16	16	16	16	16	16	16	16	16
Dynamic latent size	256	256	256	256	256	256	256	256	256
Static latent size	64	64	64	64	64	64	64	64	64
Evolution MLP linear layers	5	5	5	5	5	5	5	5	5
Evolution hidden size	256	256	256	256	256	256	256	256	256
Activation	ELU	ELU	ELU	ELU	ELU	ELU	ELU	ELU	ELU
AE normalization	GN	GN	GN	GN	GN	GN	GN	GN	GN
Latent noise amplitude	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}
One-shot context	True	True	True	True	True	True	True	True	True
AE pretraining epochs	20	20	20	20	20	20	20	20	20
Subsequent training epochs	200	200	200	200	200	200	200	200	200
Initially frozen AE epochs	20	20	20	20	20	20	20	20	20
Batch size	64	64	64	64	64	20	20	20	12
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Dynamics learning rate	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
AE fine-tuning LR	10^{-5}	0.001	0.001	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}
Weight decay	0	0	0	0	0	0	0	0	0
LR schedule	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine
Gradient clipping	1	1	1	1	1	1	1	1	1
Prediction loss weights	1/.1/.1/.1	1/.1/.1/.1	1/.1/.1/.1	1/.1/.1/.1	1/.1/.1/.1	1/.1/.1/.1	1/.1/.1/.1	1/.1/.1/.1	1/.1/.1/.1
Latent loss weights	1/1/1/1	1/1/1/1	1/1/1/1	1/1/1/1	1/1/1/1	1/1/1/1	1/1/1/1	1/1/1/1	1/1/1/1

The encoder uses four spatial downsampling blocks, GroupNorm as GN, and ELU. The latent evolution network has three Linear–ELU pairs followed by two additional linear layers. A different trajectory from the same environment supplies the one-shot context. The prediction and latent losses use four rollout steps with the listed weights; reconstruction and consistency coefficients are both 1.

LNS. We used a convolutional autoencoder with a spatially structured latent space and a parameter-conditioned convolutional propagator. The 1D encoder maps 256 points to 16 latent sites with four channels, while the 2D encoder downsamples each axis by 8 with 64 latent channels. A three-block conditional residual propagator evolves the latent state using Fourier-embedded PDE parameters, with the pretrained autoencoder frozen during dynamics training. The configuration is given in Table 16.

Table 16: **LNS implementation hyperparameters.**

Hyperparameter	Combined	Advection	Burgers	Heat	Wave-B	GS	Wave2D	Vorticity	HeterNS
Model input grid	256	256	256	256	256	32×32	64×64	128×128	64×64
State channels	1	1	1	1	1	2	2	1	1
Latent spatial grid	8	8	8	8	8	4×4	8×8	8×8	8×8
Latent channels	16	16	16	16	16	64	64	64	64
Encoder channel schedule	E1	E1	E1	E1	E1	E2	E2	E2	E2
Decoder channel schedule	D1	D1	D1	D1	D1	D2	D2	D2	D2
AE residual blocks / level	1	1	1	1	1	1	1	1	1
AE attention heads	8	8	8	8	8	8	8	8	8
AE attention head size	64	64	64	64	64	64	64	64	64
Dynamics blocks	3	3	3	3	3	3	3	3	3
Dynamics hidden size	128	128	128	128	128	128	128	128	128
Condition input size	3	1	3	21	2	2	2	1	2
Condition embedding size	96	64	84	84	64	64	64	64	64
Dynamics dilation	2	2	2	2	2	2	2	2	2
Periodic geometry	True	True	True	True	False	True	False	True	True
AE / dynamics epochs	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
AE batch / GPU	8	8	8	8	8	8	8	8	8
AE frames / trajectory	14	14	25	25	25	20	30	30	20
Dynamics batch / GPU	100	100	100	100	100	100	100	100	100
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam
AE learning rate	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}

Continued on next page

Table 16 – continued from previous page

Hyperparameter	Combined	Advection	Burgers-F	Heat	Wave-B	GS	Wave2D	Vorticity	HeterNS
Dynamics learning rate	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
Minimum learning rate	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
Weight decay	0	0	0	0	0	0	0	0	0
LR schedule	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine
Gradient clipping	1	1	1	1	1	1	1	1	1

E1=$[64, 64, 64, 128, 128, 128]$; D1=$[128, 128, 128, 64, 64]$; E2=$[64, 64, 64, 128, 128]$; D2=$[128, 128, 64, 64]$. AE Adam betas are $(0.5, 0.9)$; dynamics use Adam defaults $(0.9, 0.999)$. Evaluation is recursive from one true initial frame.

MAE-PDE. we pretrained a masked autoencoder on five-frame solution windows. Its ViT encoder used four layers, hidden size 192, six attention heads, an MLP ratio of 2, and learned positional embeddings. The time–space patches were 1×8 in 1D, $1 \times 4 \times 4$ for Gray–Scott, and $1 \times 8 \times 8$ for the other 2D datasets. We masked 75% of tokens in 1D and 90% in 2D and used a two-layer reconstruction decoder with width 64 and four heads. For forecasting, we froze the encoder and projected its representation to a 64-dimensional condition for an FNO with four layers and width 48. The forecaster used 24 modes in 1D, eight per axis for Gray–Scott, and twelve per axis for the other 2D datasets, predicting residual updates from the five-frame history. The configuration is given in Table 17.

Table 17: **MAE-PDE implementation hyperparameters.**

Hyperparameter	Combined	Advection	Burgers	Heat	Wave-B	GS	Wave2D	Vorticity	HeterNS
Model input grid	256	256	256	256	256	32×32	64×64	64×64	64×64
State channels	1	1	1	1	1	2	2	1	1
Patch size (time, space)	1×8	1×8	1×8	1×8	1×8	$1 \times 4 \times 4$	$1 \times 8 \times 8$	$1 \times 8 \times 8$	$1 \times 8 \times 8$
Input history (frames)	5	5	5	5	5	5	5	5	5
Transformer depth	4	4	4	4	4	4	4	4	4
Hidden size	192	192	192	192	192	192	192	192	192
MLP ratio	2	2	2	2	2	2	2	2	2
Attention heads	6	6	6	6	6	6	6	6	6
Attention head size	32	32	32	32	32	32	32	32	32
Dropout	0	0	0	0	0	0	0	0	0
Block normalization	LayerNorm	LayerNorm	LayerNorm	LayerNorm	LayerNorm	LayerNorm	LayerNorm	LayerNorm	LayerNorm
Activation	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU
Positional embedding	Learned	Learned	Learned	Learned	Learned	Learned	Learned	Learned	Learned
MAE decoder depth / width	2 / 64	2 / 64	2 / 64	2 / 64	2 / 64	2 / 64	2 / 64	2 / 64	2 / 64
MAE decoder heads	4	4	4	4	4	4	4	4	4
Masking ratio	0.75	0.75	0.75	0.75	0.75	0.9	0.9	0.9	0.9
FNO layers / width	4 / 48	4 / 48	4 / 48	4 / 48	4 / 48	4 / 48	4 / 48	4 / 48	4 / 48
FNO modes per axis	24	24	24	24	24	8	12	12	12
FNO condition embedding	64	64	64	64	64	64	64	64	64
Pretrain / forecast epochs	20 / 10	20 / 10	20 / 10	20 / 10	20 / 10	20 / 10	20 / 10	20 / 10	20 / 10
Pretrain / forecast batch	96	128	128	128	128	12	12	12	12
Pretraining LR	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
Forecasting LR	8×10^{-4}	8×10^{-4}	8×10^{-4}	8×10^{-4}	8×10^{-4}	8×10^{-4}	8×10^{-4}	8×10^{-4}	8×10^{-4}
Weight decay	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}
Optimizer / schedule	AdamW / cos.	AdamW / cos.	AdamW / cos.	AdamW / cos.	AdamW / cos.	AdamW / cos.	AdamW / cos.	AdamW / cos.	AdamW / cos.
Warmup fraction	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Gradient clipping	1	1	1	1	1	1	1	1	1

AdamW uses $(\beta_1, \beta_2) = (0.9, 0.98)$ during pretraining and $(\beta_1, \beta_2) = (0.9, 0.999)$ during forecasting.

ENMA. We encoded solution trajectories with a convolutional variational autoencoder using 32 latent channels. A temporal transformer provided context to a spatial transformer that predicted masked latent tokens; each stack contained six layers with a hidden size of 512, eight attention heads, and a feed-forward width of 2048. Each token covered one latent spatial site. Both stacks used RMSNorm, query/key normalization, SwiGLU, sinusoidal positional embeddings, and rotary attention. A conditional flow-matching head with three residual blocks and width 512 generated latent values, which were mapped back to physical fields by the VAE decoder. The configuration is given in Table 18.

Table 18: **ENMA implementation hyperparameters.**

Hyperparameter	Combined	Advection	Burgers	Heat	Wave-B	GS	Wave2D	Vorticity	HeterNS
Model input grid	128	128	128	128	128	32×32	64×64	64×64	64×64
VAE spatial reduction	8	8	8	8	8	4×4	4×4	4×4	4×4
Latent spatial grid	16	16	16	16	16	64	256	256	256
Latent channels	32	32	32	32	32	32	32	32	32
Latent spatial patch size	1	1	1	1	1	1×1	1×1	1×1	1×1
Transformer depth	6	6	6	6	6	6	6	6	6
Hidden size	512	512	512	512	512	512	512	512	512
FFN hidden size	2048	2048	2048	2048	2048	2048	2048	2048	2048
MLP ratio	4	4	4	4	4	4	4	4	4
Attention heads	8	8	8	8	8	8	8	8	8
Dropout	0	0	0	0	0	0	0	0	0
Block normalization	RMSNorm	RMSNorm	RMSNorm	RMSNorm	RMSNorm	RMSNorm	RMSNorm	RMSNorm	RMSNorm
Activation	SwiGLU	SwiGLU	SwiGLU	SwiGLU	SwiGLU	SwiGLU	SwiGLU	SwiGLU	SwiGLU
Positional encoding	Sin. + RoPE	Sin. + RoPE	Sin. + RoPE	Sin. + RoPE	Sin. + RoPE	Sin. + RoPE	Sin. + RoPE	Sin. + RoPE	Sin. + RoPE
Flow head depth / width	3 / 512	3 / 512	3 / 512	3 / 512	3 / 512	3 / 512	3 / 512	3 / 512	3 / 512
Sampling steps	10	10	10	10	10	10	10	10	10
Latent training frames	14	14	14	14	14	10	10	10	10
VAE / dynamics epochs	200 / 300	200 / 300	200 / 300	200 / 300	200 / 300	200 / 300	200 / 300	500 / 1000	200 / 300
VAE / dynamics batch / GPU	32 / 32	32 / 32	32 / 32	32 / 32	32 / 32	32 / 64	32 / 64	32 / 64	32 / 64
VAE / dynamics LR	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
VAE weight decay	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
Dynamics weight decay	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}
VAE KL weight	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
Optimizer / schedule	AdamW / cos.	AdamW / cos.	AdamW / cos.	AdamW / cos.	AdamW / cos.	AdamW / cos.	AdamW / cos.	AdamW / cos.	AdamW / cos.
Gradient clipping	1	1	1	1	1	1	1	1	1

AdamW uses $(\beta_1, \beta_2) = (0.9, 0.999)$ for the VAE and $(\beta_1, \beta_2) = (0.9, 0.95)$ for the latent dynamics model.

D Additional experiments

Terminology. For brevity, throughout the appendix we refer to the JEPA baseline as **Vanilla**, the model with Physics-Aligned Latent Geometry (PAG) as **Geo**, and the full model with both PAG and the Physics-Structured Latent Predictor (PSP) as **Physics**. Thus, Vanilla→Geo isolates the effect of PAG, whereas Geo→Physics isolates the additional effect of PSP.

D.1 Representation Analyses

We compare the representations learned by JEPA with LNS (Li et al., 2025), which learns an autoencoder-based latent state, and VideoMAE (Tong et al., 2022), which learns representations through masked pixel reconstruction. We organize our analysis mainly on Vorticity around three questions. **Q1: What physical information is encoded?** We first examine whether the representations preserve governing parameters and local physical states through parameter and local-state probing in subsection D.1.2 and D.1.5. **Q2: How is this information organized?** We then characterize the global and spatial organization of the latent space through variance concentration, cross-model similarity, and local spatial structure in subsection D.1.3. **Q3: Is the representation easy to evolve?** Finally, we study temporal trajectory geometry and explicitly separate state reconstruction from autoregressive rollout accuracy in subsection D.1.4 and D.1.6. Together, these analyses distinguish physical informativeness from latent organization and temporal evolvability.

D.1.1 Models and evaluation protocol

Let $z = E(x)$ denote a frozen encoder representation of the model's native observation x. The primary VideoMAE baseline is the 10 M-parameter Tiny encoder pretrained for 100 epochs.

We analyze the learned representations from three complementary analyses on Vorticity. First, we evaluate parameter readout using attentive probes over the representations of the first 16 observed frames, using 10%, 50%, and 100% of the labeled training data. Second, we evaluate local physical-state accessibility using resolution- and dimension-controlled linear probes across multiple PDEs.

Third, we analyze the latent organization of 2,400 training, 120 external-ID, and 120 far-OOD trajectories across LNS, VideoMAE, and JEPA. For each 30-frame trajectory, we construct 15 representations corresponding to adjacent frame pairs, centered at times $0.5, 2.5, \ldots, 28.5$. For LNS and JEPA, each representation is obtained by averaging the features of the two adjacent frames. For VideoMAE, we use the corresponding tubelet features extracted from two overlapping 16-frame clips (frames 0–15 and 14–29). Thus, the VideoMAE context changes between the pair centers at 14.5 and 16.5. All representations are extracted from observed trajectories and are used only for representation analysis, rather than causal forecasting.

For the Vorticity latent-organization analysis, we process each model independently. We first standardize each latent channel using statistics computed from the training trajectories and then fit a model-specific 16-dimensional PCA basis (PCA16 (Pearson, 1901)) on the standardized training representations. The retained PCA coordinates are further normalized by a single global scale determined by their total retained variance, without whitening individual principal components. We use this PCA16 space to characterize variance concentration and temporal geometry, and additionally perform an auxiliary linear viscosity readout. The linear probe is fitted on 1,800 training trajectories and evaluated on 600 internal holdout trajectories, while the external-ID and far-OOD trajectories are kept as separate test sets. Because PCA is fitted independently for each model, neither the PCA axes nor the original latent dimensions are assumed to be shared across JEPA, LNS, and VideoMAE.

D.1.2 Parameter Information Analyses

We mainly discuss in Vorticity. The 10%, 50%, and 100% label budgets are nested subsets of complete viscosity environments, containing 1,200, 6,000, and 12,000 training trajectories. The regression target is train-standardized log viscosity. Probe optimization runs for 100 epochs with a fixed seed, and checkpoint selection uses ID validation data before evaluation on the ID test set.

JEPA achieves normalized test MSEs of 0.042597, 0.012902, and 0.009310 across the three supervision budgets (Table 19). Its advantage is modest at 10%, but widens with more labeled data: at 100%, JEPA reduces error by 55.3% relative to LNS and 50.9% relative to VideoMAE. Compared with VideoMAE, the reduction increases from 7.9% to 47.5% to 50.9% as supervision grows, indicating that the strongest separation emerges in the higher-label regime rather than under limited supervision.

Table 19: **Primary comparison.** Vorticity viscosity readout from frozen JEPA, LNS, and VideoMAE.

Representation	MSE, 10% ↓	MSE, 50% ↓	MSE, 100% ↓	R^2, 100% ↑	Rel. MAE (%) ↓
JEPA, raw z	0.042597	0.012902	0.009310	0.9897	4.12
LNS AE	0.043677	0.028622	0.020830	0.9769	6.00
VideoMAE-Tiny	0.046264	0.024560	0.018950	0.9790	5.88

Native-token attentive probes use the first 16 observed frames. MSE uses train-standardized log viscosity. VideoMAE denotes the 10M-parameter Tiny encoder pretrained for 100 epochs.

VideoMAE nevertheless provides a strong physical representation: its full-budget R^2 is 0.9790 and mean relative viscosity error is 5.88%, compared with 0.9769 and 6.00% for LNS and 0.9897 and 4.12% for JEPA. VideoMAE outperforms LNS at the two larger budgets, while LNS has slightly lower error at 10%.

Table 20: **Viscosity information in the aligned pooled representations.**

Representation	Holdout R^2 ↑	External ID R^2 ↑	kNN-5 R^2 ↑	Far-OOD R^2 ↑	Far-OOD RMSE ↓
JEPA, raw z	0.9555	0.9659	0.7820	-11.2557	4.0933
LNS	0.9484	0.1422	0.6780	-23.3780	5.7730
VideoMAE-Tiny	0.9182	0.9194	0.7586	-15.9398	4.8123

Linear probes use PCA16 features at the final two-frame center, 28.5, with 1,800 fitting trajectories, 600 internal holdout trajectories, and separate 120-trajectory ID and far-OOD sets. The kNN-5 column uses all 15 centers in the internal holdout. Far-OOD RMSE is standardized by the training target scale.

The standardized PCA-based comparison reaches a consistent conclusion using a much smaller linear readout (Table 20; Figure 13). At the final two-frame center, JEPA has internal-holdout/external-ID R^2 of 0.9555/0.9659, compared with 0.9484/0.1422 for LNS and 0.9182/0.9194 for VideoMAE. LNS's external-ID decline coexists with a Spearman correlation of 0.9605, and the prediction scatter shows a systematic offset despite preserved ordering. However, strong ID readout does not directly translate to parameter extrapolation. On far-OOD viscosities, all three linear probes yield negative R^2, with standardized RMSEs of 4.0933, 5.7730, and 4.8123 for JEPA, LNS, and VideoMAE, respectively. JEPA therefore retains the smallest extrapolation error, but it does not provide reliable out-of-range prediction.

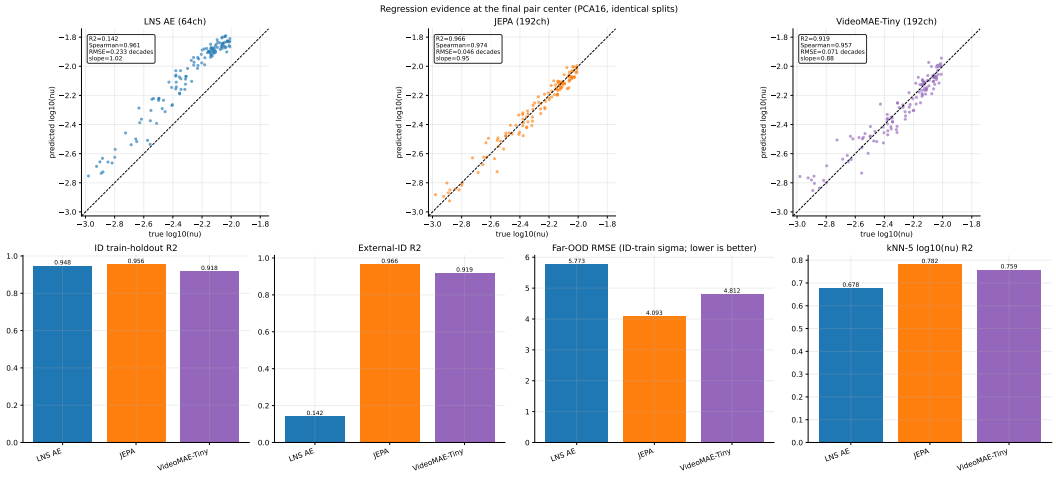


Figure 13: **Viscosity readout across evaluation regimes.** Upper panels compare external-ID viscosity predictions with numerical truth at the final two-frame center. Lower panels report internal holdout readout, external-ID readout, far-OOD standardized RMSE, and five-neighbor readout. LNS retains strong rank ordering despite a calibration shift on external ID.

A five-nearest-neighbor probe (kNN-5) over all aligned pair centers further separates physical information from linear probing. Viscosity R^2 is 0.7820 for JEPA, 0.6780 for LNS, and 0.7586 for VideoMAE. VideoMAE therefore retains substantial local parameter organization even though its dominant PCA structure differs visibly from the other two models.

D.1.3 Variance concentration and latent organization

The three representations exhibit markedly different variance concentrations (Figure 14; Table 21). After standardizing the training channels, LNS is highly concentrated in a few dominant directions: its first two principal components explain 97.11% of the variance, and only two components are required to reach 95% explained variance. JEPA is less concentrated, with 86.47% captured by the first two components and six components required for 95% variance. VideoMAE exhibits the broadest spectrum, with only 45.02% captured by the first two components and eight components required to reach 95%. The corresponding entropy effective ranks are 1.31, 2.77, and 8.92 for LNS, JEPA, and VideoMAE, respectively. These results show that LNS compresses variation into a small number of dominant directions, whereas JEPA and particularly VideoMAE distribute variation across a broader latent subspace.

Neither a compact spectrum nor a higher effective rank directly ranks physical usefulness. LNS retains strong viscosity ordering in a concentrated representation, JEPA combines a moderately distributed spectrum with stronger readout, and VideoMAE's larger effective rank partly reflects temporal-position structure. The pairwise linear centered kernel alignment (CKA) (Kornblith et al., 2019) matrix further characterizes the differences between their representations (Table 22). On 10,000 aligned pooled samples, linear CKA is 0.8018 between JEPA and LNS, compared with 0.1361 between JEPA and VideoMAE and 0.1964 between LNS and VideoMAE. Despite this low cross-model similarity, VideoMAE achieves an external-ID viscosity R^2 of 0.9194 (Table 20). Thus, different representation structures can support useful viscosity inference.

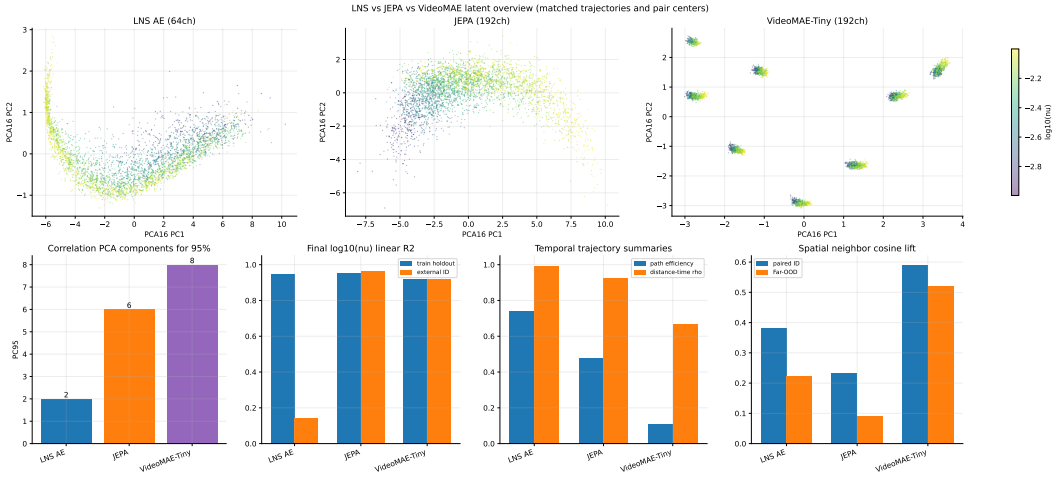


Figure 14: **Primary baseline families on matched Vorticity trajectories and two-frame centers.** The upper panels show model-specific pooled PCA coordinates colored by viscosity. Lower panels summarize variance concentration, held-out linear readout, training-trajectory geometry, and native-grid spatial similarity.

Table 21: **Variance concentration and temporal organization of pooled features.**

Representation	PC95	Eff. rank	Top-2 var. (%)	ID path efficiency ↑	ID distance–time ρ ↑
JEPA	6	2.77	86.47	0.4901	0.9451
LNS	2	1.31	97.11	0.7581	0.9925
VideoMAE-Tiny	8	8.92	45.02	0.1070	0.6695

Spectral statistics use training-channel-standardized pooled features. Temporal statistics use external ID trajectories in each model's PCA16 space. Higher path efficiency means a straighter path; neither it nor effective rank is a physical-accuracy score. These are pooled-channel spectra, not token PCA spectra.

D.1.4 Temporal and spatial organization

We next examine whether physically informative representations also induce latent trajectories that are easy to evolve. Path efficiency, defined as the endpoint displacement divided by the accumulated step length, measures how directly a trajectory progresses through latent space. On external ID data, LNS achieves a path efficiency of 0.7581, compared with 0.4901 for JEPA and 0.1070 for VideoMAE (Table 21). The corresponding correlations between elapsed time and distance from the initial state are 0.9925, 0.9451, and 0.6695, respectively. Thus, LNS follows a straighter and more monotonically separating trajectory than JEPA, despite JEPA exhibiting stronger physical readout in the preceding analyses. This distinction suggests that physical structure and evolvability are complementary properties: a representation may encode governing factors well without automatically yielding the simplest geometry for rollout. The same trend is observed on the training trajectories (Figure 15), motivating the explicit geometry adaptation introduced later.

VideoMAE requires an additional caveat because its pooled geometry is strongly influenced by clip-relative position. Relative tubelet position is nearly perfectly linearly readable ($R^2 = 0.9995$) and explains 94.5% of the pooled PCA16 variance (Figure 16). Although its stitched trajectory is sensitive to the clip-context boundary, temporal progression remains well organized within each individual view ($\rho \approx 0.95$). We therefore treat its pooled temporal geometry as context dependent rather than as a context-invariant instantaneous state.

All three representations also preserve local spatial organization (Table 23; Figure 17). On their shared 16×16 token grids, LNS and JEPA obtain ID neighbor–distant cosine lifts of 0.3809 and 0.2335, respectively, with corresponding far-OOD values of 0.2218 and 0.0902. VideoMAE also exhibits persistent local structure, but its 8×8 token grid has a different physical spacing. Its spatial statistics are therefore included as a descriptive reference rather than a resolution-controlled com-

Table 22: Pairwise linear CKA on 10,000 aligned pooled samples.

Model	JEPA	LNS	VideoMAE
JEPA	1.0000	0.8018	0.1361
LNS	0.8018	1.0000	0.1964
VideoMAE	0.1361	0.1964	1.0000

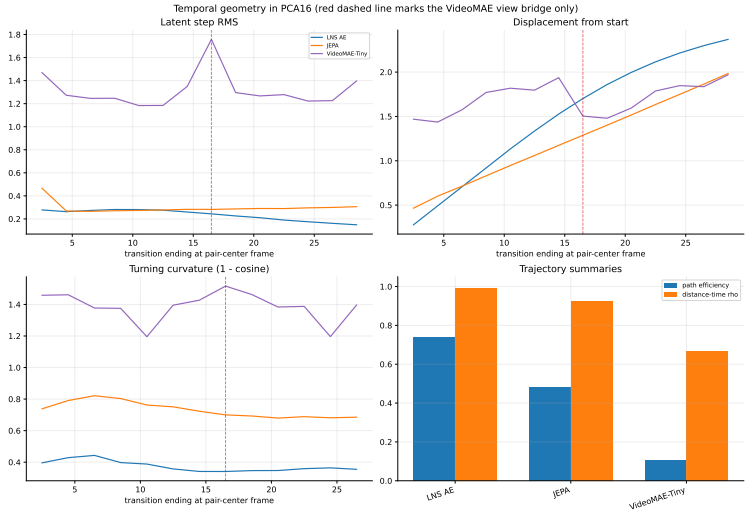


Figure 15: **Temporal organization in model-specific pooled PCA16 coordinates.** *UpperLeft:* Latent step magnitude measures the size of consecutive temporal transitions; LNS contracts, JEPA stays stable, and VideoMAE fluctuates sharply. *UpperRight:* Distance from the initial state measures global temporal progression; LNS separates most steadily; JEPA remains stable; VideoMAE is disrupted. *LowerLeft:* Temporal turning, quantified by one minus the cosine similarity between consecutive latent increments, measures changes in trajectory direction; LNS turns least, JEPA moderately, and VideoMAE most. *LowerRight:* Path efficiency and distance–time correlation summary; LNS is most smooth; JEPA remains ordered; VideoMAE is most context dependent.

parison. Overall, these results further separate physical informativeness from geometric simplicity: JEPA provides a physically informative representation, while additional geometric adaptation is needed to make its latent evolution more favorable for rollout.

Table 23: **Spatial organization of the primary representation families.**

Representation	Native grid	ID neighbor cos. ↑	ID distant cos. ↑	ID lift ↑	Far-OOD lift ↑
JEPA	16×16	0.4792	0.2457	0.2335	0.0902
LNS	16×16	0.4501	0.0692	0.3809	0.2218
VideoMAE-Tiny	8×8	0.6210	0.0296	0.5914	0.5192

Lift denotes neighboring-token minus distant-token cosine similarity, averaged across the 15 two-frame centers. A larger lift indicates stronger local spatial organization.

D.1.5 Extension to other PDEs

The attentive comparison further extends JEPA and LNS to Wave2D, Burgers, and Heat (Table 24; Figure 18), where JEPA achieves lower test MSE across all label budgets. At full supervision, the error reductions are 21.5%, 59.1%, and 13.8%, respectively. The largest gains appear in Wave2D wave-speed readout (R^2: 0.4783→0.5893) and Burgers log-diffusivity (0.2020→0.6734), while Heat remains challenging for both models.

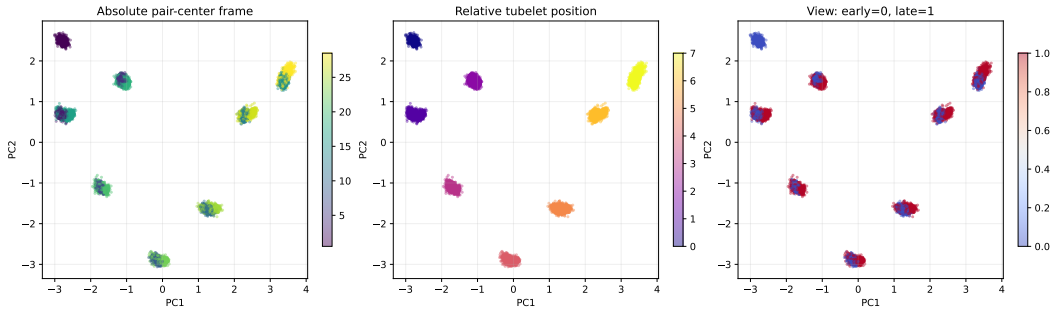


Figure 16: **VideoMAE pooled features colored by absolute time, relative tubelet position, and clip identity.** *Left:* Absolute-time coloring tests whether the clusters reflect physical time. *Middle:* Relative-position coloring shows that the clusters are primarily organized by tubelet position within the input clip. *Right:* Early/late view coloring shows substantial overlap within clusters, indicating that clip identity alone does not explain the separation.

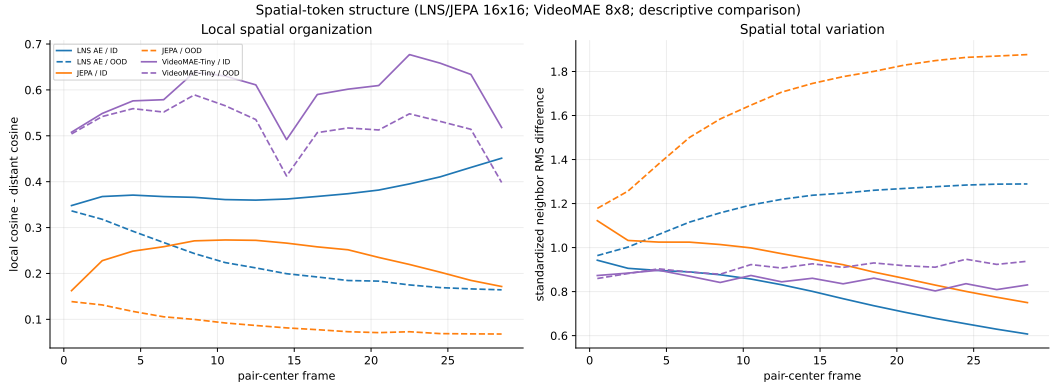
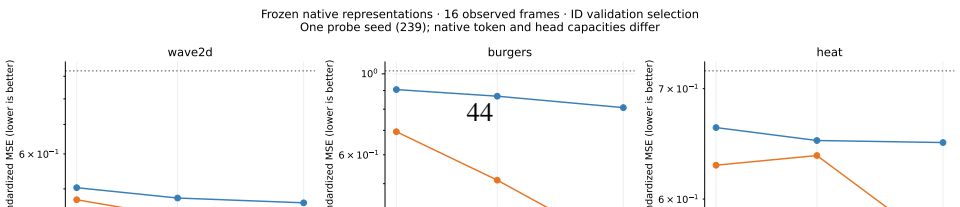


Figure 17: **Native-token spatial organization on paired ID and far-OOD trajectories.** *Left:* Local spatial organization. Local lift measures spatial locality; LNS exceeds JEPA, while VideoMAE is highest descriptively. *Right:* Spatial total variation. Spatial variation grows most under JEPA OOD, while ID becomes smoother.

Table 24: **Parameter readout across label budgets and PDEs.**

Dataset	Budget	LNS MSE ↓	JEPA MSE ↓	Decrease (%) ↑	LNS R^2 ↑	JEPA R^2 ↑
Wave2D	10%	0.503139	0.472475	6.1	0.4359	0.4684
Wave2D	50%	0.476702	0.431590	9.5	0.4651	0.5143
Wave2D	100%	0.464928	0.364986	21.5	0.4783	0.5893
Burgers	10%	0.905448	0.694140	23.3	0.1057	0.3144
Burgers	50%	0.868372	0.511263	41.1	0.1423	0.4950
Burgers	100%	0.807950	0.330690	59.1	0.2020	0.6734
Heat	10%	0.662985	0.629060	5.1	-0.1061	-0.0495
Heat	50%	0.651126	0.637604	2.1	-0.0863	-0.0638
Heat	100%	0.649273	0.559498	13.8	-0.0832	0.0665

To assess whether the full-budget improvements are consistent across specific test trajectories, we bootstrap the JEPA–LNS MSE difference, where negative values favor JEPA. The 95% intervals remain below zero on Wave2D ($[-0.1294, -0.0718]$) and Burgers ($[-1.0273, -0.0847]$), indicating consistent improvements across resampled environments. For Heat, the interval ($[-0.1867, 0.0102]$) crosses zero, so the improvement is less conclusive.



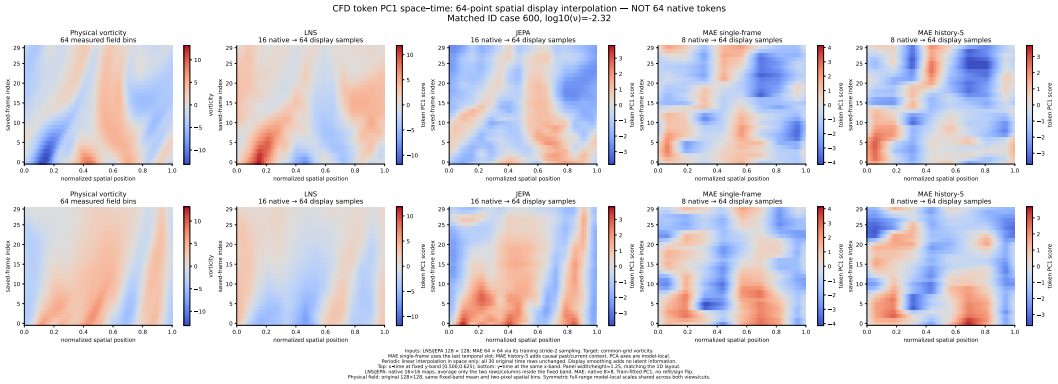


Figure 19: **CFD physical-field and token-PC1 space–time maps.**Matched CFD space–time maps along a fixed spatial line comparing physical vorticity with each model's token-PC1 representation.

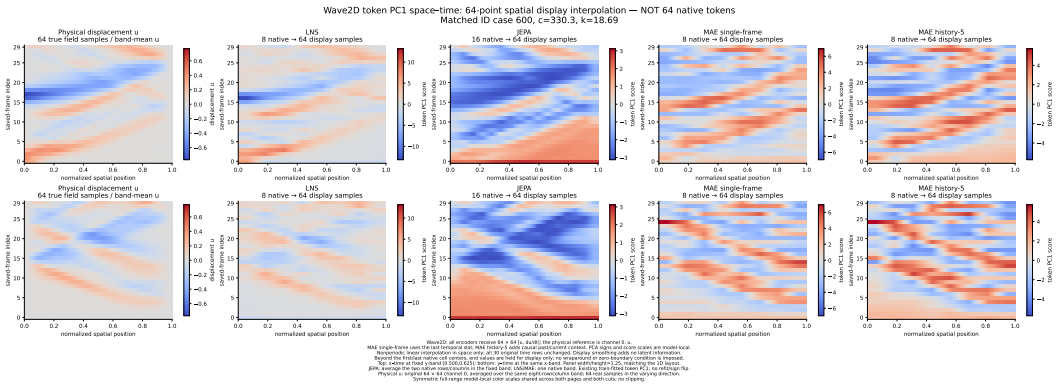


Figure 20: **Matched Wave2D displacement and token-PC1 space–time maps.**

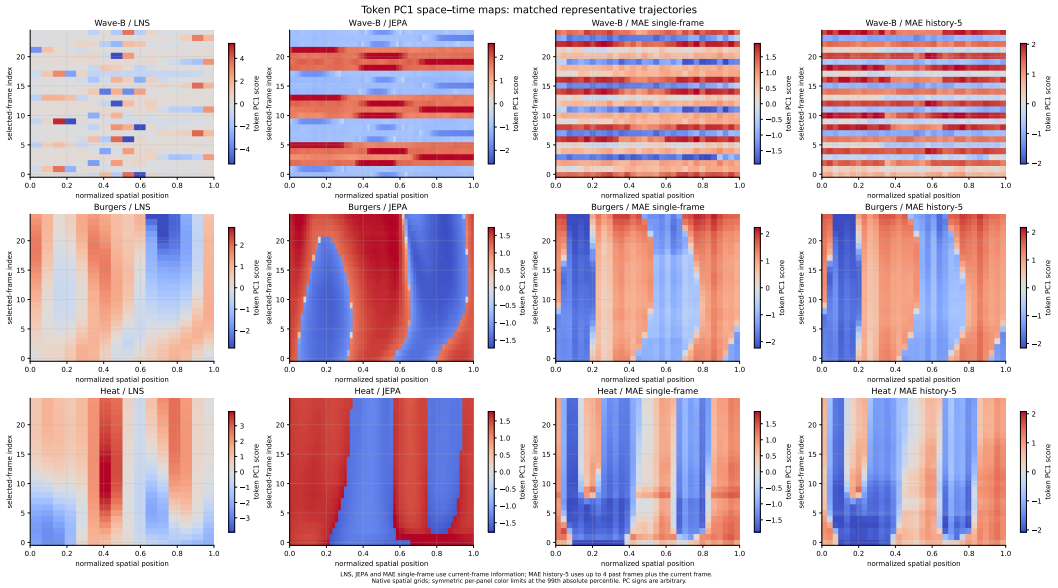


Figure 21: **Matched Wave-B, Burgers, and Heat token-PC1 space–time maps.**

Table 25: **Dimension-controlled local physical-state readout.**

Representation	CFD ID $R^2 \uparrow$	CFD OOD $R^2 \uparrow$	Wave2D ID $R^2 \uparrow$	Wave2D OOD $R^2 \uparrow$
LNS	0.9578	0.8818	0.8258	0.7254
JEPA	0.9921	0.9492	0.8773	0.8633
MAE-PDE, single	0.0962	0.0698	0.3658	0.3063
MAE-PDE, history-5	0.1034	0.0758	0.3696	0.2933

D.1.6 Direct Reconstruction versus Autoregressive Rollout

Accurate reconstruction measures how well a representation preserves a physical state; forecasting additionally requires predicting its evolution. We examine these properties separately. Direct reconstruction decodes the ground-truth representation at each time, $\widetilde{u}_t = D_{\mathrm{rec}}(z_t)$. Autoregressive rollout instead decodes predicted latent states, $\widehat{u}_t = D_{\mathrm{roll}}(\widehat{z}_t)$, generated from the observed prefix without access to future states. For the evaluated frame set $\mathcal{T}$, we average the physical relative L^2 error over trajectories and report

$$e_a = \frac{1}{N}\sum_{i=1}^{N} \frac{\|u^a_{i,\mathcal{T}} - u_{i,\mathcal{T}}\|_2}{\|u_{i,\mathcal{T}}\|_2}, \qquad R = \frac{e_{\mathrm{roll}}}{e_{\mathrm{rec}}}, \tag{37}$$

where $a \in \{\mathrm{rec}, \mathrm{roll}\}$ and the norm covers time, space, and physical channels. Thus, R is a ratio of mean errors.

Table 26: **Direct reconstruction and autoregressive rollout errors on the ID validation sets for JEPA and Zebra.** We report $R = e_{\mathrm{roll}}/e_{\mathrm{rec}}$, which measures the increase from reconstruction to rollout error; larger values indicate a larger reconstruction–prediction gap. Parentheses denote the decoder input. Zebra results are unavailable for Gray–Scott.

Dataset (input)	e_{rec}	e_{roll}	JEPA-R	e_{rec}	e_{roll}	Zebra-R
Advection (z)	0.0029	0.0235	8.12	0.0003	0.0079	26.33
Burgers (z)	0.0151	0.135	8.99	0.0016	0.1540	96.25
Heat (z)	0.0085	0.0808	9.51	0.0019	0.1150	60.53
Wave-B (z)	0.0125	0.1115	8.92	0.0011	0.2450	222.73
Wave-2D (z)	0.0415	0.363	8.74	0.0010	0.2070	207.00
Combined (z)	0.0027	0.0332	12.31	0.0022	0.0079	3.59
Vorticity (z)	0.0104	0.086	8.26	0.0170	0.1190	7.00
GS (z)	0.0079	0.0807	10.22	-	-	-

Table 26 shows a clear separation between reconstruction fidelity and temporal predictability. For JEPA, the rollout-to-reconstruction ratio remains relatively consistent across datasets, ranging from 8.12 to 12.31. In contrast, Zebra exhibits much larger and more variable gaps on several benchmarks, including Burgers ($96.25\times$), Wave-B ($222.73\times$), and Wave2D ($207.00\times$), despite its substantially smaller reconstruction errors. The ratio alone, however, does not rank rollout accuracy: for example, Zebra has a much larger ratio on Wave2D but a lower absolute rollout error than JEPA, whereas JEPA achieves lower rollout error on Burgers, Heat, Wave-B, and Vorticity. These results therefore show that reconstruction quality and temporal predictability are distinct properties, and that a representation with lower reconstruction error is not necessarily easier to evolve accurately over time.

D.2 Mechanism Analyses

We investigate how geometry alignment and physics-structured prediction improve latent dynamics and long-horizon forecasting. We organize our mechanism analysis around four questions. **Q1: Does geometry projection align latent evolution with the underlying physical trajectory?** We examine full-dimensional latent trajectory geometry in subsection D.2.3 and controlled 1D experiments in subsection D.2.6 to determine whether the projected representation better reflects physical evolution. **Q2: Does the physics-structured predictor capture parameter-dependent dynamics more faithfully?** We compare predicted and numerical parameter responses at matched initial

conditions in both latent and physical spaces in subsection D.2.2. **Q3: Do these improvements translate into better forecasting under parameter shift?** We evaluate field- and latent-space rollout accuracy across ID and OOD regimes in subsection D.2.4. **Q4: Why does physics supervision improve long-horizon extrapolation?** We separate one-step prediction defects from subsequent error amplification to identify the source of the long-horizon improvement in subsection D.2.5. Together, the analyses support all four questions: geometry projection improves physical alignment, the structured predictor better captures parameter-dependent dynamics, these gains improve field forecasting under parameter shift, and reduced error amplification provides a mechanism for the long-horizon extrapolation benefit.

D.2.1 Evaluation protocol and dynamics diagnostics

Experimental setup. We analyze three JEPA-based systems on Vorticity: *Vanilla*, which evolves standardized encoder features; *Geo*, which evolves features transformed by the frozen geometry projector; and *Physics*, which further introduces the structured parameter-dependent predictor. Wave2D provides compatible Geo and Physics systems. Within each dataset, Geo and Physics share the same frozen encoder, projector, decoder, numerical precision, and integration scheme, and all free rollouts start from the observed initial state. For Vorticity, we include a 120 Bridge data ranging viscosity between $[10^{-4}, 10^{-3}]$ for further analysis.

Parameter response. For parameter-response analysis, each of the 120 OOD trajectories is paired with a training-reference trajectory sharing the same initial condition but a different physical parameter, so the two rollouts start from the same latent state and differ only in the supplied parameter. We evaluate the resulting finite parameter response over 29 future steps, while the 1,200 ID-validation trajectories are used separately for standard rollout evaluation. Because the OOD split reuses training initial conditions, this protocol tests parameter generalization at fixed initial conditions. A separate perturbation analysis uses depth-eight Vorticity predictors, denoted *Free-D* and *Physics-D*, where Physics-D adds supervision on the viscosity-dependent branch.The 1D controls likewise follow their own dataset-specific protocols.

For two different parameters ξ_a and ξ_b, we compare the predicted response $\Delta\widehat{x}_t = \widehat{x}_t(\xi_b) - \widehat{x}_t(\xi_a)$ with the numerical response $\Delta x_t = x_t(\xi_b) - x_t(\xi_a)$, where x denotes either the decoded field or latent state. We measure response direction (Cos.), magnitude (Amp.), and relative error as

$$C_x(t) = \frac{\langle \Delta\widehat{x}_t, \Delta x_t \rangle}{\|\Delta\widehat{x}_t\|_2 \|\Delta x_t\|_2}, \qquad R_x(t) = \frac{\|\Delta\widehat{x}_t\|_2}{\|\Delta x_t\|_2}, \qquad E_x(t) = \frac{\|\Delta\widehat{x}_t - \Delta x_t\|_2}{\|\Delta x_t\|_2}.$$

The targets are $C_x = R_x = 1$ and $E_x = 0$. Metrics are averaged over time and then across the 120 matched initial conditions, with paired confidence intervals obtained by resampling complete initial conditions 1,000 times. Geo and Physics are compared against the same frozen target representation $q_t = G(E(u_t))$, so their latent-response comparison is independent of the physical decoder.

D.2.2 Parameter-dependent dynamics

On Vorticity, the main improvement comes from a more accurate direction of the parameter-induced dynamics while preserving the response magnitude (Table 27; Figure 22). The mean field-response cosine increases from 0.5161 for Vanilla to 0.6259 for Geo and 0.7079 for Physics, whereas only physics maintain field-response magnitude ratios close to one. In the shared projected latent space, Physics further increases the response cosine from 0.9128 to 0.9240 and reduces Geo's magnitude overestimation from 1.1423 to 1.0428. Thus, geometry alignment makes the parameter response substantially more directionally consistent, while the structured predictor further improves its direction and latent amplitude calibration.

Table 27: **Parameter-response accuracy at fixed initial conditions.**

Dataset	Model	$C_u \to 1$	$R_u \to 1$	$C_q \to 1$	$R_q \to 1$
Vorticity	Vanilla	0.5161	0.7923	0.7802	1.0138
Vorticity	Geo	0.6259	0.8462	0.9128	1.1423
Vorticity	Physics	0.7079	0.9870	0.9240	1.0428
Wave2D	Geo	0.9582	0.9786	0.9389	0.9903
Wave2D	Physics	0.9861	0.9959	0.9465	1.0023

Means over 29 steps and 120 matched initial conditions. Cosine has target 1; the magnitude ratio also has target 1. Geo and Physics use the same unquantized projected latent target. Vanilla uses its own standardized encoder coordinates.

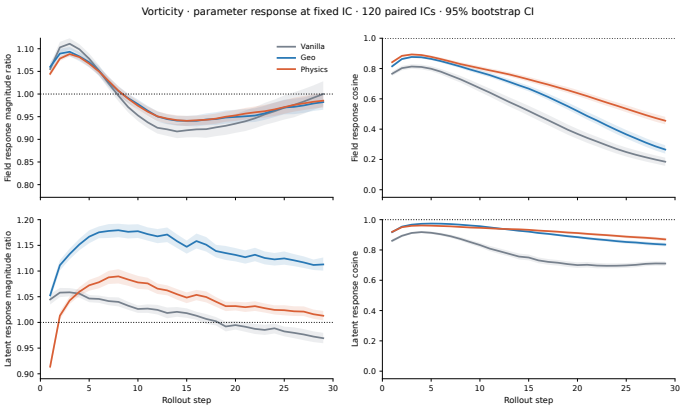


Figure 22: **Vorticity parameter-response magnitude and direction at fixed initial conditions.** Curves average all 120 matched pairs; shaded bands are pointwise 95% initial-condition bootstrap intervals. *Upper row* measures decoded fields. *Lower row* measures latent states. Both magnitude and cosine have target 1.

Wave2D shows an even cleaner improvement in both response direction and magnitude (Figure 23). Physics increases the field-response cosine from 0.9582 to 0.9861 and moves the magnitude ratio from 0.9786 to 0.9959, closely matching the numerical parameter response. In latent space, the cosine also improves from 0.9389 to 0.9465, while the magnitude ratio moves from 0.9903 to 1.0023. These results show that the structured predictor more faithfully captures the combined effect of changing (c, k) in both latent and physical space, with the largest gain appearing in the decoded field response.

The temporal decomposition exposes a meaningful difference between the equations (Table 28; Figure 24). Over steps 22–29, Vorticiy retains positive field and latent cosine gains of 0.1693 and 0.0346. Wave2D retains a field cosine gain of 0.0528, but its latent cosine decreases from 0.8900 to 0.8789. The paired change is -0.0111 [-0.0216, -0.0011], and latent relative-response error increases by 0.0234 [0.0050, 0.0431]. Average improvement therefore coexists with a measurable late-horizon latent limitation. A single whole-rollout average would conceal this reversal.

Table 28: **Paired changes in response accuracy: Physics minus Geo.**

Dataset	Steps	ΔC_u [95% CI]	ΔC_q [95% CI]	ΔE_q [95% CI]
NS	1–29	+0.0820 [0.0716, 0.0935]	+0.0112 [0.0084, 0.0144]	-0.0595 [-0.0695, -0.0509]
NS	22–29	+0.1693 [0.1472, 0.1937]	+0.0346 [0.0288, 0.0411]	-0.1067 [-0.1215, -0.0932]
Wave2D	1–29	+0.0279 [0.0236, 0.0321]	+0.0077 [0.0025, 0.0122]	-0.0368 [-0.0468, -0.0258]
Wave2D	22–29	+0.0528 [0.0435, 0.0625]	-0.0111 [-0.0216, -0.0011]	+0.0234 [0.0050, 0.0431]

Positive cosine changes and negative relative-error changes favor Physics. Intervals use 1,000 paired bootstrap resamples of complete initial conditions; steps within an initial condition remain together.

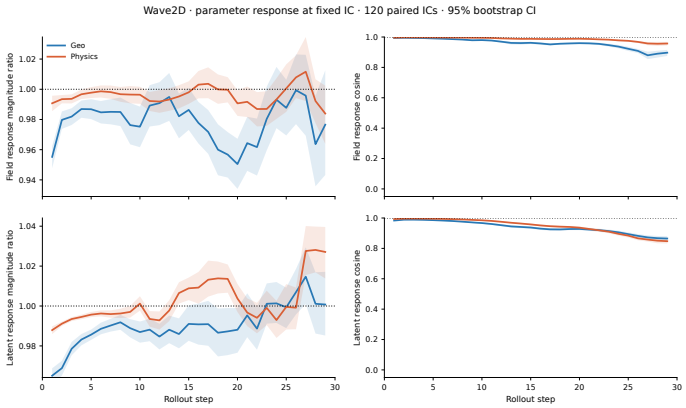


Figure 23: **Wave2D parameter-response magnitude and direction at fixed initial conditions.** Curves average all 120 matched pairs; shaded bands are pointwise 95% initial-condition bootstrap intervals. *Upper row* measures decoded fields. *Lower row* measures latent states. Both magnitude and cosine have target 1.

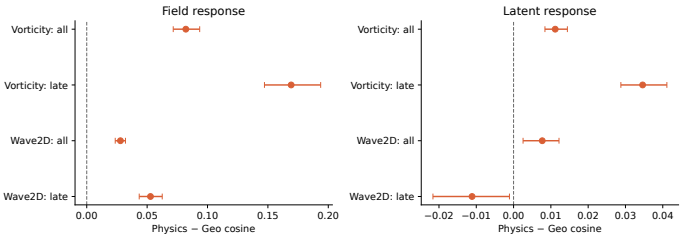


Figure 24: **Paired changes in parameter-response direction**. With 95% initial-condition bootstrap intervals. All denotes steps 1–29 and late denotes steps 22–29. The Wave2D late latent interval lies below zero, while its late field interval lies above zero.

D.2.3 Latent trajectory geometry

This section examines how parameter changes reshape latent trajectories and whether the predicted trajectory geometry matches the projected numerical dynamics. We consider two complementary diagnostics. First, for matched initial conditions under two parameter settings, we measure cross-parameter trajectory separation and velocity-direction similarity (Figure 25). Second, for each individual trajectory, we measure speed, curvature, and path efficiency in the full projected latent space (Table 29; Figure 28). The projected numerical trajectory, $q_t = G(E(u_t))$, serves as the reference, while PCA is used only for qualitative visualization.

Cross-parameter trajectory geometry. On Vorticity, Physics more closely reproduces the late geometry of the projected numerical trajectories (Figure 25). Its cross-parameter separation is 0.2743 versus 0.2694 for the projected truth, compared with 0.2995 for Geo, while its velocity-direction similarity is 0.2296 versus 0.2124 for the projected truth and 0.1804 for Geo. On Wave2D, the numerical trajectories under the two parameter settings are themselves nearly orthogonal at late times, with a velocity-direction similarity of -0.0010; Physics closely reproduces this behavior at -0.0014, compared with -0.0026 for Geo. Thus, cross-parameter direction similarity should be interpreted relative to the numerical dynamics rather than as a standalone measure of organization.

The corresponding PCA trajectory families provide qualitative examples of how the same initial state branches under different parameter settings (Figures 26 and 27). These visualizations are illustrative only; all quantitative geometry metrics are computed in the full latent space.

Single-trajectory temporal geometry. The aggregate trajectory statistics reveal different behaviors across equations (Table 29; Figure 28). On Wave2D, Physics more closely matches the projected

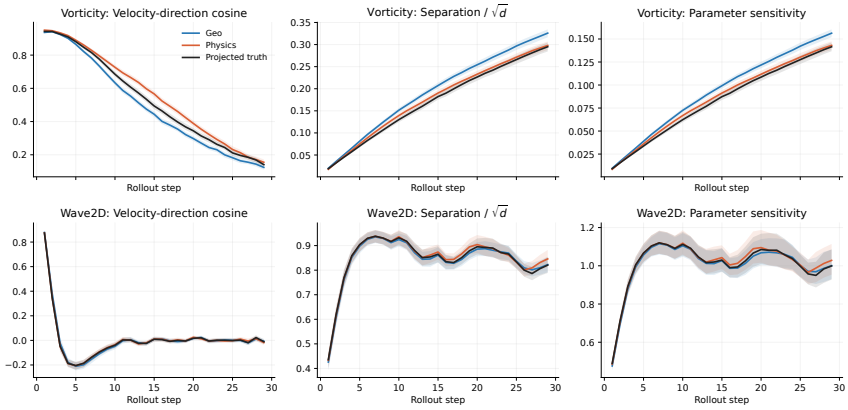


Figure 25: **Full-dimensional geometry of the two trajectories generated from each matched initial condition.** Velocity-direction cosine compares the two parameter settings within a model, whereas response cosine compares predicted and true parameter-induced differences. Lower cross-parameter direction similarity can therefore be correct, as the Wave2D projected-truth curve illustrates. Shading uses 1,000 initial-condition bootstrap resamples.

numerical speed and curvature than Geo: its mean speed is 0.5759 versus 0.5664 for the projected truth, while its curvature is 1.4768 versus 1.4755. On Vorticity, however, Physics produces a slower and smoother trajectory than the projected numerical reference, with speed 0.0624 versus 0.0718 and curvature 0.9768 versus 1.1934. Despite this departure, Physics achieves better parameter-response direction and lower field rollout error. These results show that smoother latent dynamics are not necessarily more physical; the relevant criterion is agreement with the projected numerical trajectory rather than smoothness alone.

Table 29: **Trajectory geometry in the shared projected coordinates on the original OOD split.**

Dataset	Source	Mean speed	Mean curvature	Path efficiency
Vorticity	Projected truth	0.0718	1.1934	0.1449
Vorticity	Geo	0.0737	1.1557	0.1452
Vorticity	Physics	0.0624	0.9768	0.1673
Wave2D	Projected truth	0.5664	1.4755	0.0775
Wave2D	Geo	0.5548	1.4539	0.0784
Wave2D	Physics	0.5759	1.4768	0.0759

Each row averages 120 trajectories. Speed is the step displacement norm divided by the square root of latent dimension; curvature is the change in normalized step direction. Projected truth is unquantized.

D.2.4 Field and latent rollout accuracy

To connect parameter-response accuracy with forecasting, we report the mean per-frame NRMSE over steps 1–29 (Table 30). On Vorticity, Physics reduces field error from 0.0429 to 0.0362 on ID validation and from 0.3917 to 0.2966 on OOD trajectories, corresponding to reductions of 15.6% and 24.3%. Wave2D shows a larger improvement in physical space: OOD field error decreases from 0.3039 to 0.1521 and far-OOD error from 0.2652 to 0.1405, while the corresponding latent-space gains are smaller and reverse under the far-OOD condition (0.2212→0.2366). The horizon curves show that these differences persist across rollout (Figure 29). Overall, the structured predictor consistently improves physical-field forecasting, although lower field error does not require uniformly lower latent error.

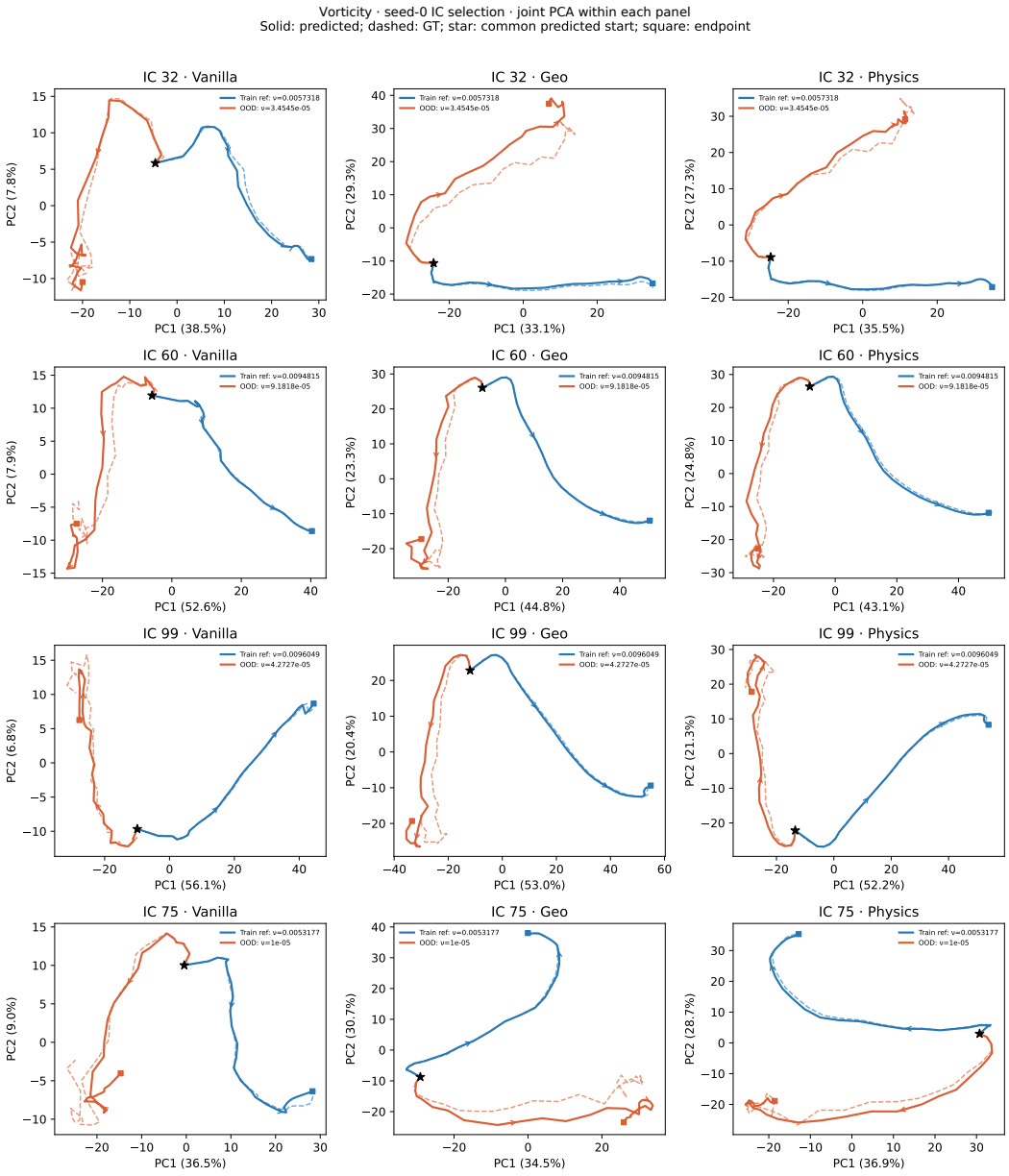


Figure 26: **Vorticity predicted and true latent trajectories for all four initial conditions selected. Each panel fits one joint PCA to the two predicted and two true trajectories.** Solid curves are predictions; dashed curves are truth. Stars mark the common predicted initial state.

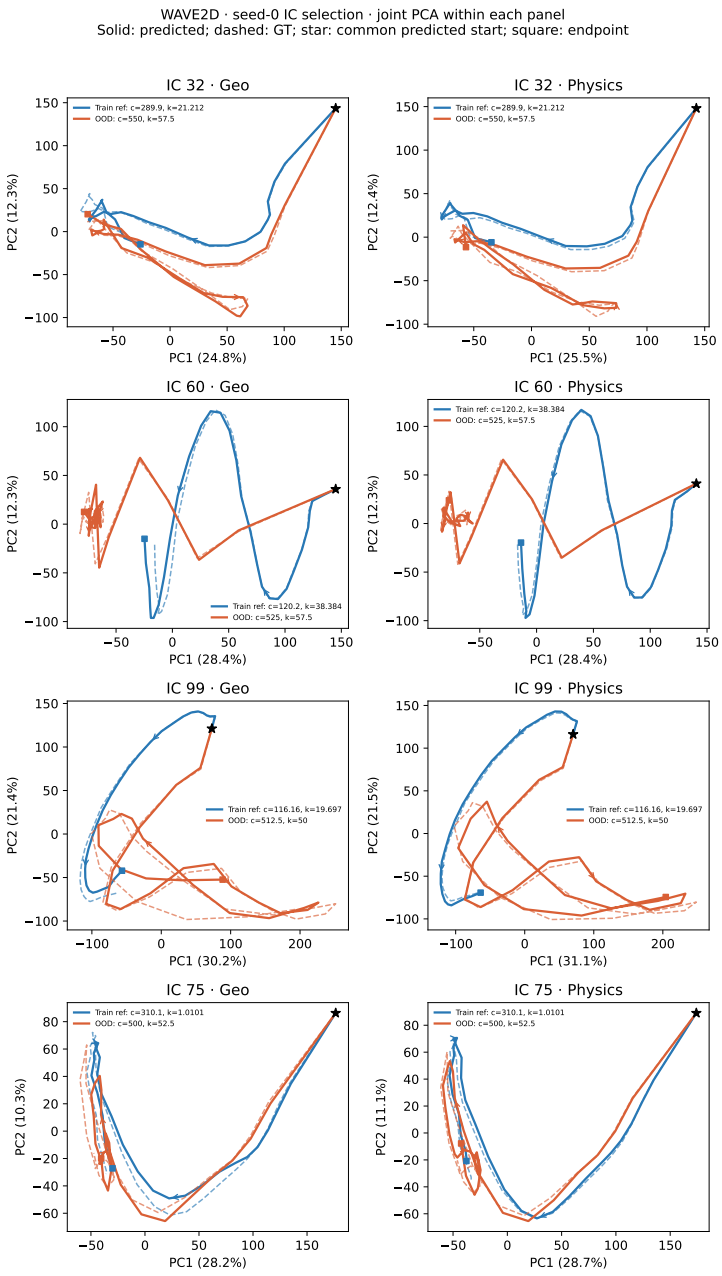


Figure 27: **Wave2D predicted and true latent trajectories for all four initial conditions selected.** Each panel fits one joint PCA to the two predicted and two true trajectories. Solid curves are predictions; dashed curves are truth. Stars mark the common predicted initial state.

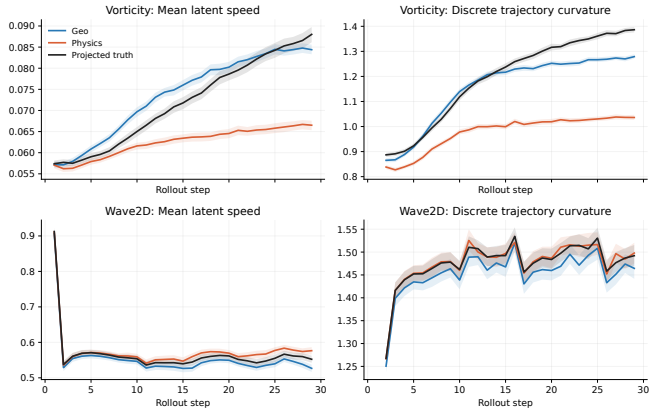


Figure 28: **Speed and direction-change curvature in the shared projected state space.** Shading is a trajectory bootstrap interval. The first curvature value is undefined and omitted. Physics is smoother than projected truth on Vorticity; the Wave2D speed and curvature are slightly larger than Geo. Smoothness alone does not rank predictive accuracy.

Table 30: **Average rollout error on the original evaluation splits.**

Dataset	Space	Split	Geo AUEC ↓	Physics AUEC ↓	Reduction (%) ↑
Vorticity	Field	ID-val	0.0429	0.0362	15.61
Vorticity	Field	OOD	0.3917	0.2966	24.27
Wave2D	Field	ID-val	0.1548	0.1083	30.01
Wave2D	Field	OOD	0.3039	0.1521	49.94
Wave2D	Field	Far-OOD	0.2652	0.1405	47.03
Wave2D	Latent	ID-val	0.1447	0.1357	6.18
Wave2D	Latent	OOD	0.2479	0.2168	12.54
Wave2D	Latent	Far-OOD	0.2212	0.2366	-6.97

AUEC averages per-frame NRMSE over steps 1–29. OOD uses all 120 original OOD trajectories; the Wave2D far-OOD group contains ten trajectories at $(c, k) = (512.5, 60)$.

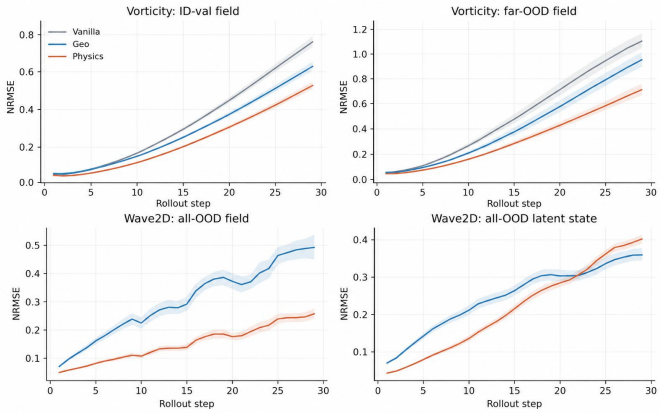


Figure 29: **Original-protocol rollout diagnostics.** Vorticity panels show ID-val and the farthest available OOD viscosity; Wave2D panels show all original OOD trajectories in field and latent space.

The final-step field visualizations put these averages alongside individual parameter changes (Figure 30; Figure 31). The Wave2D example shows smaller OOD field errors for Physics even though the preceding latent metrics identify a late-horizon limitation.

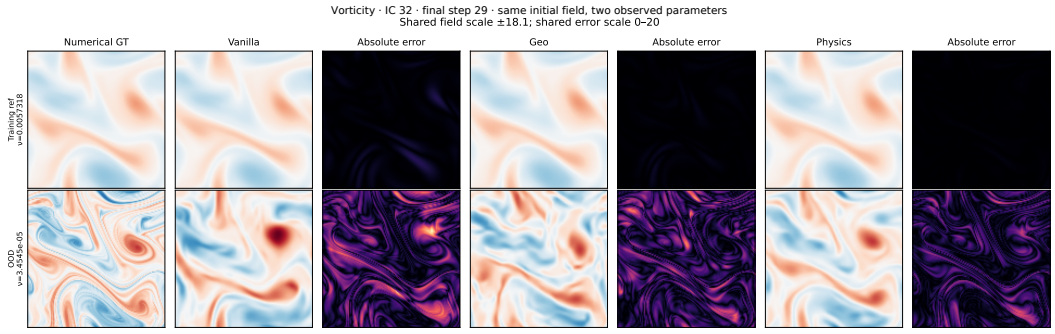


Figure 30: **Vorticity final-step fields for IC 32**, with the training-reference parameter above and the OOD parameter below. The two parameter settings start from exactly the same vorticity field.

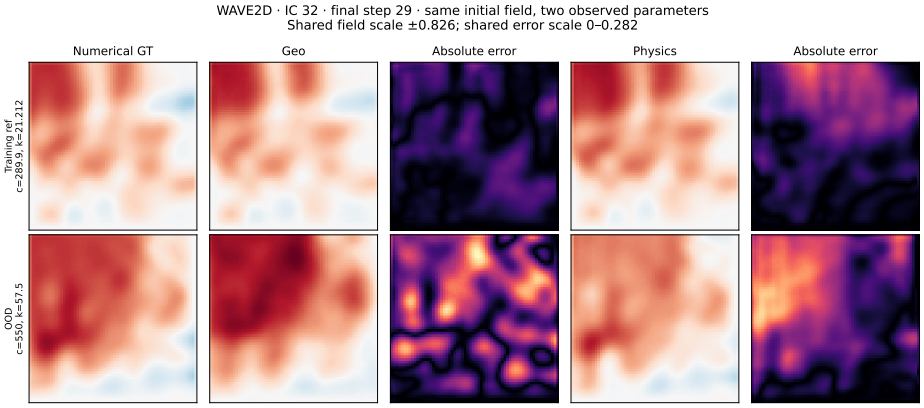


Figure 31: **Wave2D final-step fields for IC 32**, with the training-reference parameter above and the OOD parameter below.

D.2.5 Physics Supervision Reduces Error Propagation

To understand why physics supervision improves long-horizon extrapolation, we separate two sources of rollout error: the error introduced at each prediction step and the subsequent propagation of existing errors. Let $q_t = G(E(u_t))$ denote the projected numerical trajectory, $\widehat{q}_t$ the free rollout, and Ψ_θ one prediction step. Defining the rollout error $e_t = \widehat{q}_t - q_t$ and the one-step defect $r_t = \Psi_\theta(q_t) - q_{t+1}$ gives

$$e_{t+1} = r_t + \big[\Psi_\theta(q_t + e_t) - \Psi_\theta(q_t)\big] \approx r_t + A_t e_t, \qquad A_t = J\Psi_\theta(q_t).$$

Thus, long-horizon error depends not only on one-step prediction quality, but also on how strongly existing errors are amplified during rollout.

We test this mechanism using the Free-D and Physics-D Vorticity predictors described in Section D.2.1. In addition to ID and far-OOD evaluation, we use an intermediate Bridge regime with $\nu \in [10^{-4}, 10^{-3}]$, lying between the training range ($\sim [10^{-3}, 10^{-2}]$) and the far-OOD range ($[10^{-5}, 10^{-4}]$). To quantify error propagation, we measure the finite-time perturbation gain

$$A_K(q, \delta q) = \frac{\|F_\theta^{(K)}(q + \delta q) - F_\theta^{(K)}(q)\|_2}{\|\delta q\|_2}, \tag{38}$$

where $F_\theta^{(K)}$ denotes the K-step prediction map. Larger values indicate stronger amplification of an initial latent perturbation.

Physics-D substantially reduces long-horizon amplification under distribution shift (Figure 32; Table 31). At $K = 29$, the median gain changes only slightly on ID data, from 1.2860 to 1.2653, but decreases from 8.8056 to 4.9087 on Bridge and from 15.5361 to 6.4286 on far-OOD. On the far-OOD trajectories, Physics-D has lower gain in 98.3% of cases. The horizon curves further show that this difference grows with rollout length, indicating that physics supervision mainly changes the stability of long-horizon evolution rather than the local ID dynamics.

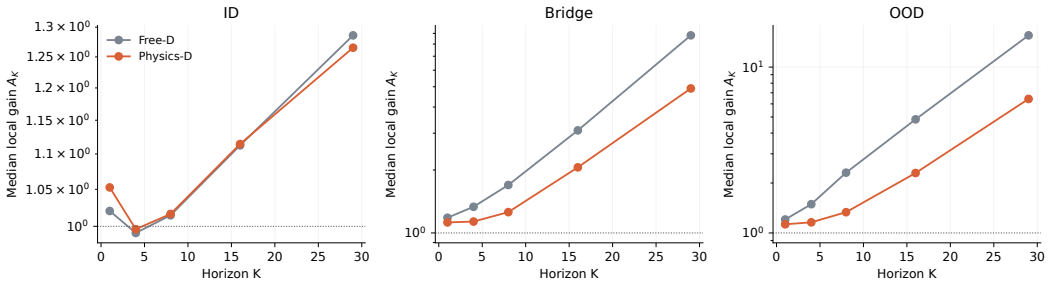


Figure 32: **Finite-time perturbation amplification on Vorticity.** Physics-D leaves ID amplification nearly unchanged but substantially suppresses long-horizon amplification on Bridge and far-OOD trajectories.

Importantly, the lower amplification is not explained by better one-step fitting. As shown in Table 31, Physics-D has slightly larger teacher-forced error on all three splits: 0.002188 versus 0.002086 on ID, 0.020687 versus 0.019979 on Bridge, and 0.027629 versus 0.026543 on far-OOD. Nevertheless, its long-horizon errors improve under distribution shift. On far-OOD trajectories, the final-step latent error decreases from 0.169647 to 0.159624, while the whole-rollout error decreases from 0.096307 to 0.090602. Bridge shows the same trend with a smaller margin, whereas ID rollout error slightly increases.

Table 31: **One-step prediction error, perturbation amplification, and latent rollout error on Vorticity.**

Split	Model	N	Teacher error	A_{29}	Final latent error	Whole latent error
ID	Free-D	120	0.002086	1.2860	0.036409	0.022268
ID	Physics-D	120	0.002188	1.2653	0.040790	0.025092
Bridge	Free-D	240	0.019979	8.8056	0.135650	0.077281
Bridge	Physics-D	240	0.020687	4.9087	0.133460	0.075893
OOD	Free-D	120	0.026543	15.5361	0.169647	0.096307
OOD	Physics-D	120	0.027629	6.4286	0.159624	0.090602

Perturbation amplification is also associated with long-horizon prediction error: across the 480 evaluated initial states, the Spearman correlation between log gain and final-step latent error is approximately 0.81 for both predictors. This association weakens after controlling for viscosity, indicating that amplification explains only part of the rollout behavior.

Overall, these results separate one-step prediction error from error propagation. Physics-D does not improve the former and slightly worsens it, but substantially reduces the latter under Bridge and far-OOD conditions. The resulting reduction in long-horizon rollout error therefore provides direct evidence that suppressed error amplification is an important contributor to the extrapolation benefit of physics supervision.

D.2.6 Extension to 1D PDEs

The one-dimensional experiments provide two complementary controls on the role of geometry. First, Burgers, Heat, and Wave-B compare the same held-out trajectories before and after the frozen geometry projection. Each test set contains 120 trajectories which is different from Table 3 with 25

observed frames, and all metrics use the full native state vectors. Projection reduces the turning-angle MAE from 46.0253 to 23.3122 degrees on Burgers, from 78.1210 to 23.0318 on Heat, and from 13.3181 to 5.7452 on Wave-B; the weighted multi-lag geometry error also decreases on all three datasets (Table 32).

Table 32: **Geometry projection on held-out one-dimensional trajectories.**

Dataset	N	Angle MAE: z	Angle MAE: q	Multi-lag: z	Multi-lag: q
Burgers	120	46.0253	23.3122	0.2387	0.0530
Heat	120	78.1210	23.0318	0.6688	0.0150
Wave-B	120	13.3181	5.7452	0.0746	0.0184

All metrics use the full 120-trajectory ID-test split and 25 observed frames. Angle errors are in degrees; multi-lag error compares latent and physical increment-direction cosines at lags 1, 2, and 4. Smaller is better.

Wave-B further shows that the objective is alignment rather than simply reducing curvature. Its mean physical turning angle is 118.7228 degrees, compared with 111.8439 for the encoder state and 118.4202 after projection. Thus, projection can either increase or decrease latent turning as needed to better match the physical trajectory geometry. These results isolate a coordinate-level effect of Geo, but do not imply that lower geometry error necessarily guarantees lower forecast error.

D.3 Extended Experiments and Ablation Studies

D.3.1 Full OOD results

Our method achieves the best OOD rollout performance across all evaluated PDE benchmarks 33. The largest gains are observed on Combined and Wave2D, where the error is reduced from the previous best 0.038 to 0.008 and from 0.610 to 0.157, respectively. On Vorticity, our method further improves over Zebra from 0.320 to 0.288, while substantial gains are also obtained on HeterNS and Gray–Scott. These results demonstrate consistently stronger parameter extrapolation across heterogeneous PDE dynamics.

Table 33: **Full OOD rollout performance across PDE benchmarks.** Relative L^2 error is reported, lower is better. Best results are **bold** and second-best results are underlined. - means the method diverges.

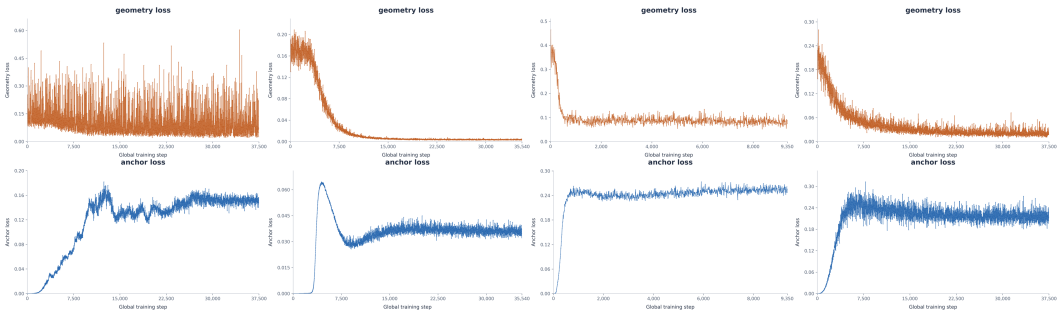
Method	Combined	Wave-2D	Vorticity	HeterNS Visc./Force	GS
FNO	-	-	0.447	-/-	0.192
GEPS	0.570	1.005	0.421	0.162/-	0.184
UniSolver	0.038	1.003	0.923	0.037/0.105	0.1636
MPP	-	-	0.845	-/-	-
DPOT	-	1.197	0.483	2.43/1.47	0.597
Poseidon-T	0.146	1.511	0.665	0.560/0.821	0.083
LE-PDE	0.169	0.924	0.715	0.334/0.923	0.239
LNS	0.1667	0.610	0.481	0.610/0.932	0.146
MAE-PDE	0.254	1.070	0.503	0.045/0.248	0.196
ENMA	0.243	1.151	0.467	1.501/2.271	0.134
Zebra	–	0.680	0.320	–	–
Ours	**0.008**	**0.157**	**0.288**	**0.011/0.103**	**0.033**
Rel. Impr.	77.9%	74.2%	9.7%	69.5%/1.5%	59.7%

D.3.2 Geometry Alignment Module

Optimization behavior of geometry alignment. We first examine the optimization behavior of the geometry projector. Figure 33 reports both the geometry loss and the anchor loss during training on four representative PDE systems. The two objectives exhibit a consistent complementary pattern. For Vorticity, Burgers, and Wave-2D, the geometry loss decreases rapidly during the early

stage of training and subsequently approaches a stable regime, indicating that the projector progressively aligns the latent trajectory with the geometry of the physical evolution. Gray–Scott shows substantially noisier geometry optimization, but the loss remains bounded throughout training.

In parallel, the anchor loss increases from its initially small value and gradually stabilizes at a dataset-dependent level. Burgers reaches this regime almost immediately, while Vorticity exhibits a transient overshoot before settling to a lower plateau. Wave-2D stabilizes after the initial rise with a mild downward drift, whereas Gray–Scott undergoes a longer adjustment before reaching a relatively steady regime. Together, these curves suggest that geometry alignment does not proceed by unconstrained deformation of the pretrained representation: as the geometry discrepancy is reduced, the anchor term becomes active and limits excessive deviation from the original latent structure. The absence of persistent growth in either objective further indicates that the two constraints can be jointly optimized across PDE systems with substantially different dynamics. This complementary behavior motivates examining whether the anchor constraint is necessary when lower geometric discrepancy can otherwise be achieved through more aggressive deformation of the latent space.



(a) GS (b) Vorticity (c) Burgers (d) Wave-2D

Figure 33: **Anchor and Geometry training loss for four benchmarks.**

Although the anchor loss stabilizes during training, its role is not to simply minimize the geometry discrepancy. We therefore remove the anchor while keeping the remaining alignment objective unchanged. As shown in Table 34, removing the anchor further reduces the turning-angle MAE from 6.3° to 4.8° on Vorticity and from 10.1° to 7.6° on Wave-2D. However, the resulting representations become substantially less informative: local-state probing deteriorates across all evaluated datasets, accompanied by considerably higher parameter-probing error. The ID/OOD rollout errors also increase from $0.040/0.397$ to $0.047/0.423$ on Vorticity and from $0.143/0.321$ to $0.166/0.354$ on Wave-2D. Thus, lower geometric error alone does not imply a better state space for dynamics prediction. The anchor constraint prevents over-deformation of the pretrained representation, balancing trajectory alignment with preservation of dynamics-relevant information.

Table 34: **Effect of the anchor constraint in geometry alignment.** Removing the anchor further decreases the turning-angle error, but substantially degrades the information retained in the representation and leads to worse ID and OOD rollout accuracy.

Dataset	Variant	Angle MAE (°) ↓	Local-state R^2 ↑	Param. MSE ↓	ID Rollout ↓	OOD Rollout ↓
Wave-2D	+ Anchor	10.1	**0.87**	**8.63**	**0.143**	**0.321**
	w/o Anchor	**7.6**	0.74	31.7	0.166	0.354
Vorticity	+ Anchor	6.3	**0.94**	**0.009**	**0.040**	**0.397**
	w/o Anchor	**4.8**	0.81	0.024	0.047	0.423

Geo modules behavior on AE. We further examine whether geometry alignment provides a representation-agnostic benefit or specifically complements predictive representations. For LNS, we freeze the pretrained encoder and apply the same geometry projector as in PDE-JEPA. We then train the dynamics predictors from scratch on the original and geometry-aligned latent spaces using identical architectures, capacities, and optimization budgets. As shown in Table 35, the original LNS representation already exhibits a substantially smaller trajectory-angle discrepancy than JEPA

(9.37° vs. 29.1°) and a lower ID rollout error (0.059 vs. 0.086). Applying geometry alignment to JEPA reduces the angle discrepancy to 6.30° and simultaneously improves rollout error to 0.040. In contrast, the same alignment only slightly reduces the LNS angle discrepancy from 9.37° to 8.21°, while degrading rollout accuracy from 0.059 to 0.062. More aggressive deformation without the anchor further reduces the angle error to 5.94°, but increases rollout error to 0.071. These results show that improved physical-geometry agreement does not universally translate into better latent dynamics. Instead, geometry alignment specifically addresses the substantial geometric mismatch of JEPA representations, whereas deforming the reconstruction-oriented LNS latent space can disturb its learned dynamical organization.

Table 35: **Effect of geometry alignment on JEPA and LNS representations.** Angle MAE measures the discrepancy between latent and physical trajectory geometry. Anchor deviation is computed as $\|q - z\|_2^2/(\|z\|_2^2 + \epsilon)$. Lower is better for all metrics.

Representation	Variant	Angle MAE ($^\circ$) ↓	Anchor Deviation ↓	ID Rollout Rel. L^2 ↓
JEPA	Original	29.1	0	0.086
JEPA	+ Geo	**6.30**	0.08	**0.040**
LNS	Original	9.37	0	**0.059**
LNS	+ Geo	8.21	**0.03**	0.062
LNS	+ Geo w/o Anchor	**5.94**	0.21	0.071

D.3.3 Predictor Module

Structured prediction and capacity on Gray–Scott. Gray–Scott supplies an additional matched comparison between a direct conditional transition and a structured continuous predictor,

$$\dot{\mathbf{q}} = \mathbf{C}_\theta(\mathbf{q}) + \frac{F}{s_F}\mathbf{D}_\theta^F(\mathbf{q}) + \frac{k}{s_k^{\mathrm{GS}}}\mathbf{D}_\theta^k(\mathbf{q})$$

where F and k are the reaction parameters and the scales are fixed from training. They share the frozen latent cache, sampled transitions, global batch of 64, optimizer schedule, 50 training epochs, and ID-validation selection rule. The continuous predictor uses RK4 with four substeps. The comparison tests the complete prediction rule, including its parameterization and numerical integration.

The structured predictor substantially improves multi-step accuracy despite nearly identical teacher errors (Table 36; Figure 34). Relative to the direct predictor, the 8-block structured model reduces the final-step pooled latent error from 0.630593 to 0.275269 and the whole-rollout error from 0.457266 to 0.171373, while maintaining essentially the same teacher error (0.041156 vs. 0.041170). The improvement is already visible at intermediate horizons, with h_8 decreasing from 0.399747 to 0.124119. These results show that the structured prediction rule provides a clear ID-validation benefit for Gray–Scott, complementing the coarse Advection result and indicating that the effect of predictor structure depends on the equation and its sampled dynamics.

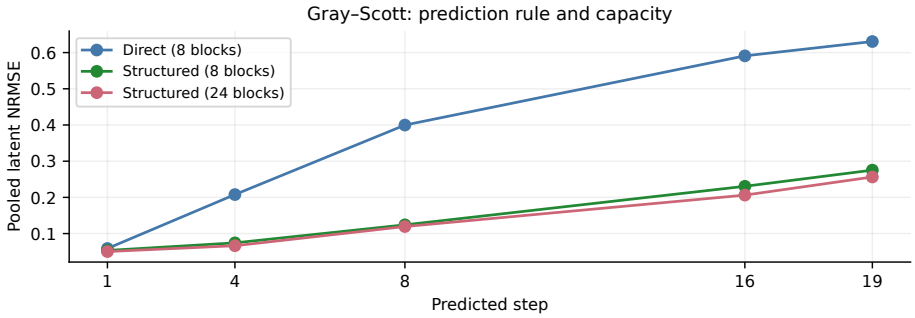


Figure 34: **Gray–Scott latent prediction on the complete ID-validation set.** The direct and structured eight-block predictors have equal parameter counts and training budgets. The twenty-four-block model tests additional capacity within the structured family.

Increasing the structured predictor from eight to twenty-four blocks provides a further, though more modest, improvement under the same optimizer-step budget. The final-step error decreases from 0.275269 to 0.253280, while the whole-rollout error decreases from 0.171373 to 0.162613; the earlier-horizon errors also improve from 0.053405 to 0.051337 at h_1 and from 0.124119 to 0.119414 at h_8. Thus, additional capacity consistently benefits the structured Gray–Scott predictor, in line with the trend observed on Vorticity.

Table 36: **Gray–Scott prediction rule and capacity controls on ID validation.**

Predictor	Params (M)	Teacher ↓	h_1 ↓	h_8 ↓	h_{19} ↓	Whole ↓
Direct (8 blocks)	14.42	0.041156	0.058798	0.399747	0.630593	0.457266
Structured (8 blocks)	14.42	0.041170	0.053405	0.124119	0.275269	0.171373
Structured (24 blocks)	42.81	0.039671	0.051337	0.119414	0.25328	0.162613

Physical-tangent supervision We next examine explicit factorization and physical-tangent supervision. Given a physical state u with instantaneous PDE tangent $\dot{u}$, we obtain its corresponding latent tangent through the Jacobian-vector product (JVP)

$$\dot{q}_{\text{phys}} = J_E(u)\dot{u} = D_u E(u)\,\dot{u},$$

where E is the frozen encoder. This computes the directional derivative of the representation along the physical evolution without forming the full encoder Jacobian. We compare a conditional generator M0, a strictly factorized generator M1, and the same factorized generator with JVP supervision M2,

$$\text{M0}: \quad \dot{q} = F_\theta(q, \beta/s_\beta), \qquad \text{M1, M2}: \quad \dot{q} = (\beta/s_\beta)A_\theta(q).$$

M2 additionally penalizes the discrepancy between the predicted latent tangent and $\dot{q}_{\text{phys}}$ with weight 0.1. All models share the frozen representation and decoder, use RK4 with four substeps, and are trained for 12,200 optimizer steps; M1 and M2 additionally share the same initialization.

As shown in Table 37, explicit factorization substantially improves prediction: from M0 to M1, the raw whole-rollout latent error decreases from 0.1100 to 0.0848, the selected latent error from 0.1091 to 0.0821, and the field RelL2 from 0.1009 to 0.0769, with corresponding reductions in phase and speed error. Adding JVP supervision further improves the aggregate trajectory metrics, yielding the lowest teacher error (0.0034), raw whole-rollout latent error (0.0200), selected latent error (0.0798), and field RelL2 (0.0737). M1 nevertheless retains lower phase and speed errors than M2, suggesting that the tangent constraint primarily improves aggregate rollout accuracy rather than every dynamical diagnostic. Overall, the matched comparison supports both explicit factorization and physical-tangent supervision at a temporal resolution that better resolves the local dynamics.

Integration Strategies. We further examine whether the performance of the structured predictor depends strongly on the numerical integrator. As shown in Table 38, low-order integration noticeably degrades prediction accuracy. Using Euler increases the final-step and whole-rollout errors

Table 37: **Ablation of factorization and physical-tangent supervision on raw-interval Advection.** All models are trained and evaluated under the same raw-interval protocol; M1 isolates the effect of explicit factorization over the conditional baseline M0, while M2 further adds physical-tangent JVP supervision.

Model	Teacher ↓	Raw latent whole ↓	Selected latent whole ↓	Field RelL2 ↓	Phase	Speed
M0: conditional	0.0039	0.1100	0.1091	0.1009	2.4208	0.2128
M1: factorized	0.0035	0.0848	0.0821	0.0769	1.1478	0.1018
M2: factorized + JVP	0.0034	0.0200	0.0798	0.0737	1.6361	0.1189

to 0.1914 and 0.0936, respectively, whereas RK2 reduces them to 0.1648 and 0.0796. Moving from RK2 to RK4 provides a further improvement, reaching 0.159669 final-step error and 0.077184 whole-rollout error.

Table 38: **Effect of numerical integration accuracy on Vorticity OOD prediction.** Higher-order integration substantially improves over Euler, while forecasting accuracy largely saturates by RK4.

Predictor	Integral	Late teacher	Final	Whole	Round trip
Physics-D	Euler	0.0518	0.1914	0.0936	1.1243
Physics-D	RK2	0.0462	0.1648	0.0796	0.8465
Physics-D	RK4	0.045295	0.159669	0.077184	0.781600
Physics-D	RK8	0.045275	0.159705	0.077226	0.670196

Beyond RK4, however, the forward-prediction metrics essentially saturate. RK8 yields nearly identical final and whole-rollout errors (0.159705 and 0.077226), despite further reducing the round-trip error from 0.781600 to 0.670196. This distinction suggests that higher-order integration continues to improve numerical reversibility, while RK4 already provides sufficient precision for forward forecasting. Therefore, the predictive advantage of the structured model is unlikely to arise simply from using an increasingly accurate numerical solver; rather, the learned continuous dynamics account for most of the forecasting gain once integration error is sufficiently controlled.

D.4 Trajectory Visualizations

D.4.1 Advection

Figure 35 visualizes Advection trajectories for two different values of the governing parameter β, showing the ground truth, forecast, absolute error, and representative spatial profiles over time.

D.4.2 Burgers

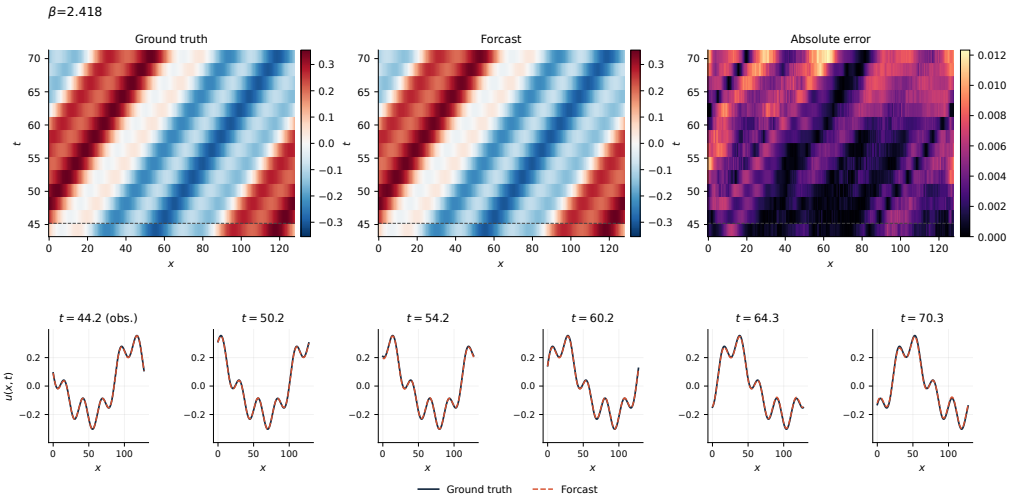
Figure 36 visualizes Burgers trajectories for two different values of the governing parameter β with fixed $\alpha = 1$ and $\gamma = 0$, showing the ground truth, forecast, absolute error, and representative spatial profiles over time.

D.4.3 Heat

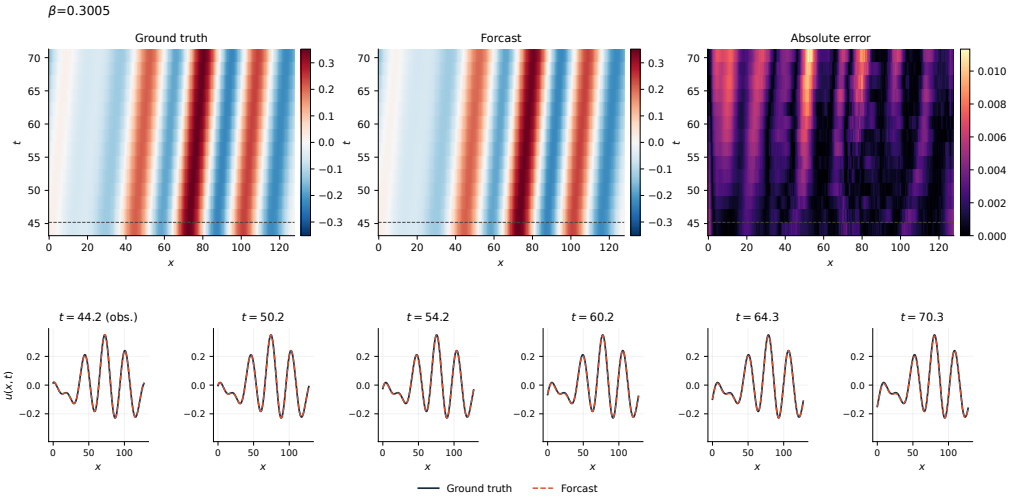
Figure 37 visualizes Heat trajectories for two different values of the governing parameter β with fixed $\alpha = 0$ and $\gamma = 0$, showing the ground truth, forecast, absolute error, and representative spatial profiles over time.

D.4.4 Wave-B

Figure 38 visualizes Wave-B trajectories under two different boundary conditions, showing the ground truth, forecast, absolute error, and representative spatial profiles over time.



(a) $\beta = 2.418$



(b) $\beta = 0.3005$

Figure 35: **Qualitative Advection forecasts under different governing parameters.**

D.4.5 Combined Equation

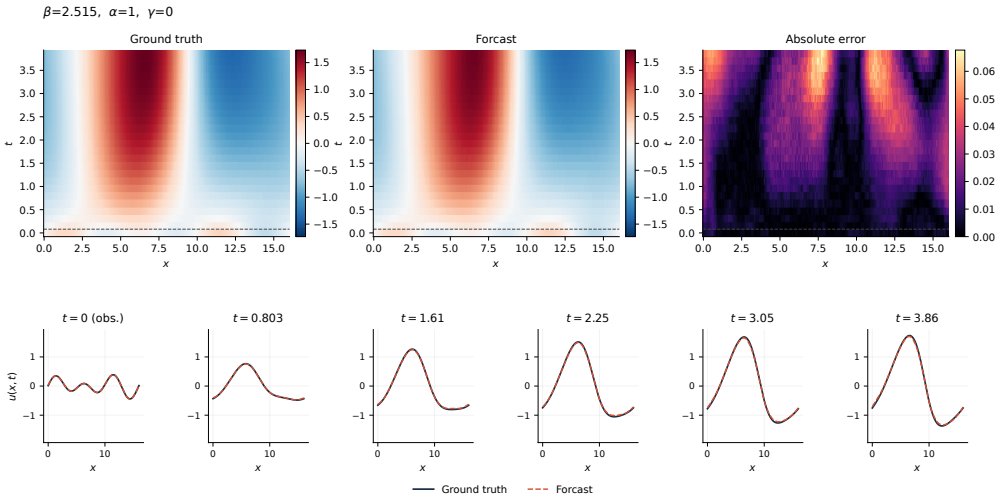
Figure 39 visualizes Combined Equation trajectories under representative ID and OOD governing parameters, showing the ground truth, forecast, absolute error, and representative spatial profiles over time.

D.4.6 Wave-2D

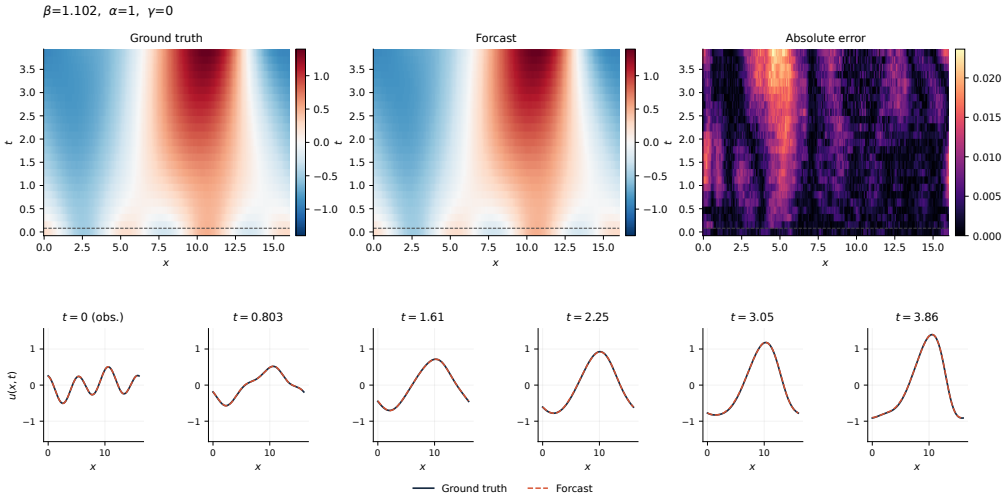
Figure 40 visualizes Wave-2D trajectories under representative ID and OOD governing parameters (c, k), showing the ground truth, forecast, and absolute error across representative rollout frames.

D.4.7 Vorticity

Figure 41 visualizes Vorticity trajectories under representative ID and OOD viscosity values ν, showing the ground truth, forecast, and absolute error across representative rollout frames.



(a) $\beta = 2.515,\ \alpha = 1,\ \gamma = 0$



(b) $\beta = 1.102,\ \alpha = 1,\ \gamma = 0$

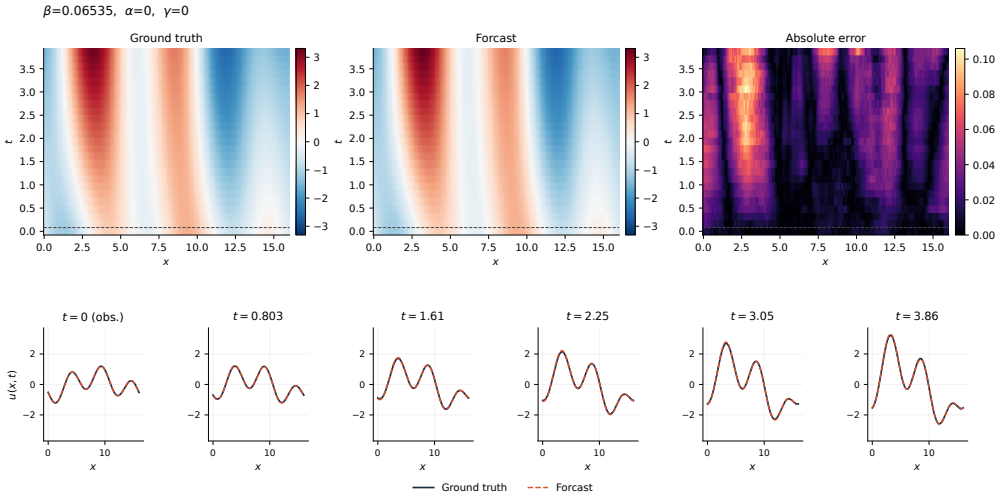
Figure 36: **Qualitative Burgers forecasts under different governing parameters.**

D.4.8 GRAY–SCOTT

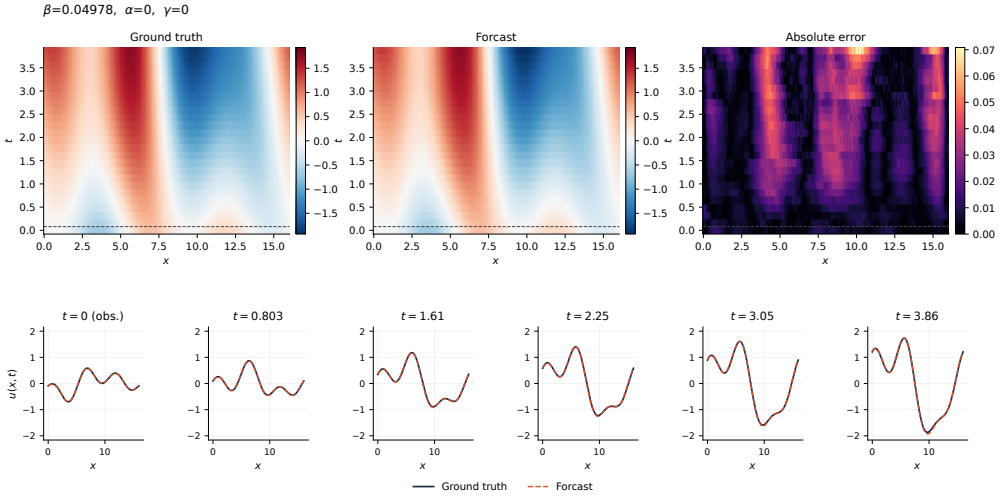
Figure 42 visualizes Gray–Scott trajectories under representative ID and OOD reaction parameters (F, k), showing the ground truth, forecast, and absolute error across representative rollout frames.

D.4.9 HETERNS

Figure 43 visualizes HeterNS trajectories under representative ID, OOD forcing, and OOD viscosity conditions, showing the ground truth, forecast, and absolute error across representative rollout frames.

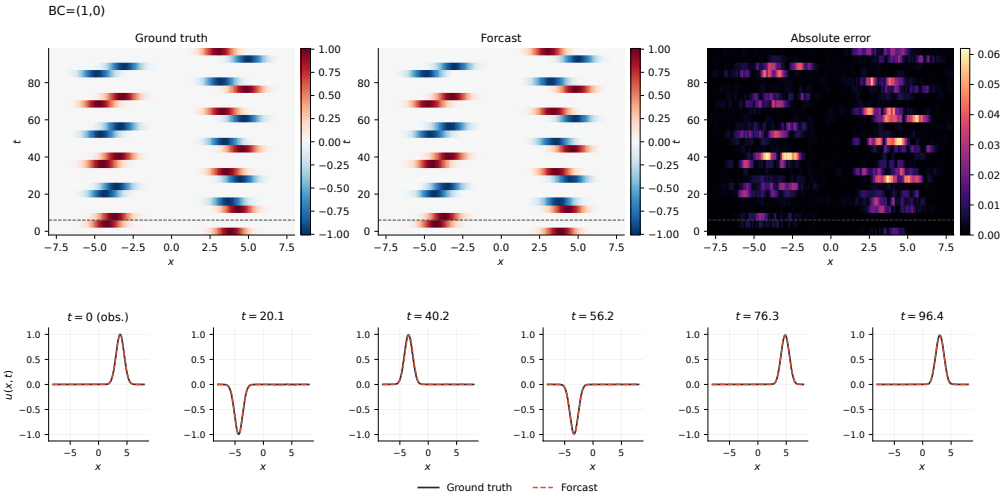


(a) $\beta = 0.06535,\ \alpha = 0,\ \gamma = 0$

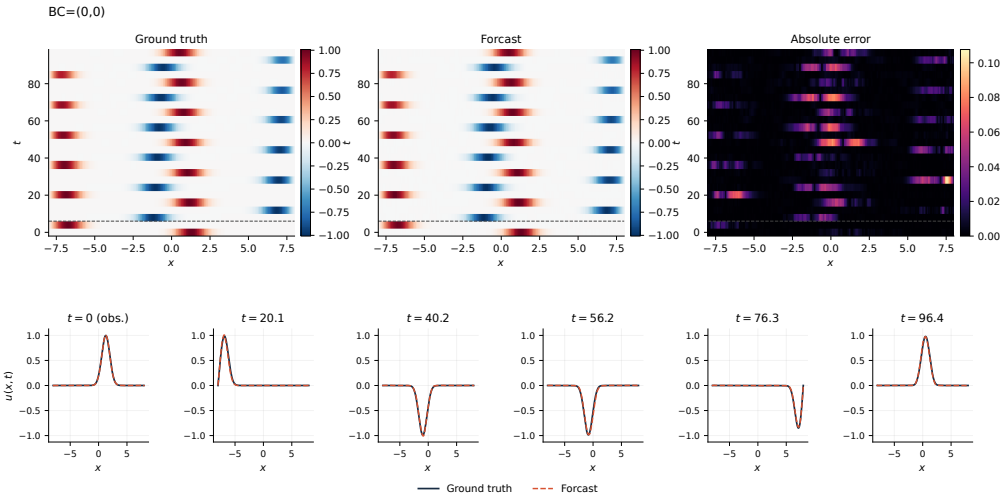


(b) $\beta = 0.04978,\ \alpha = 0,\ \gamma = 0$

Figure 37: **Qualitative Heat forecasts under different governing parameters.**

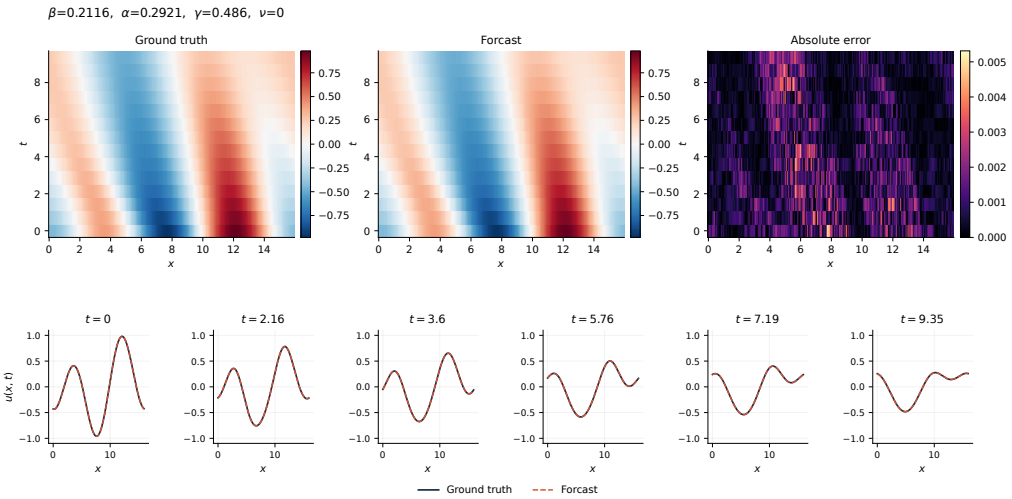


(a) BC $= (1, 0)$

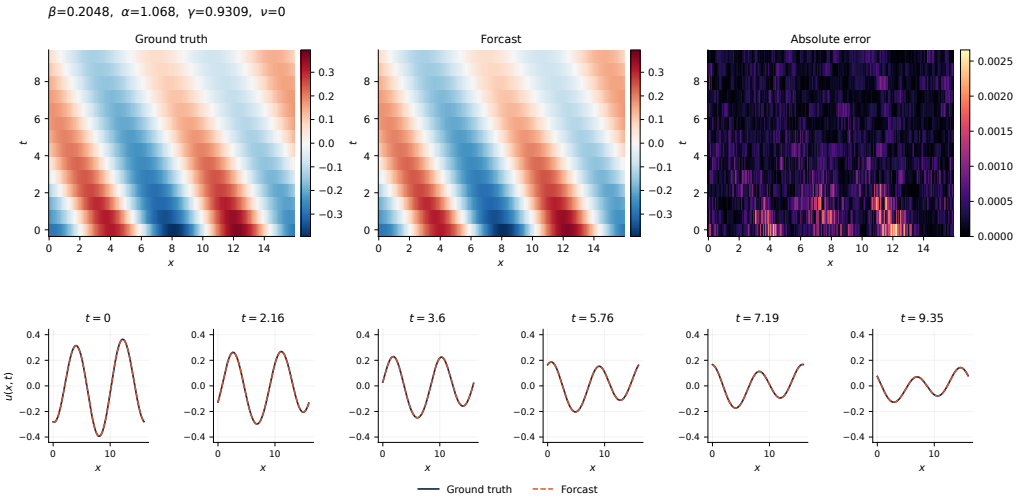


(b) BC $= (0, 0)$

Figure 38: **Qualitative Wave-B forecasts under different boundary conditions.**

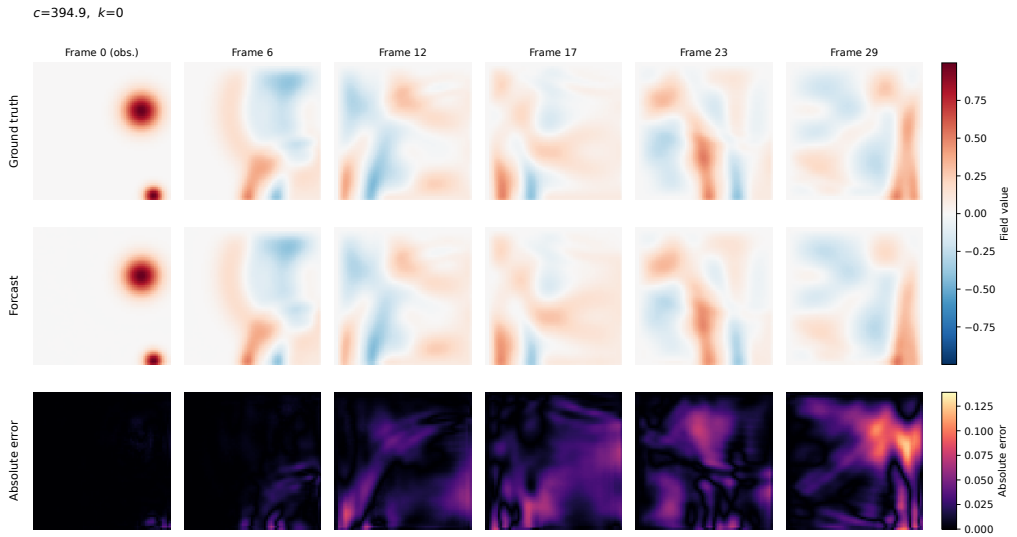


(a) ID: $\beta = 0.2116,\ \alpha = 0.2921,\ \gamma = 0.486,\ \nu = 0$

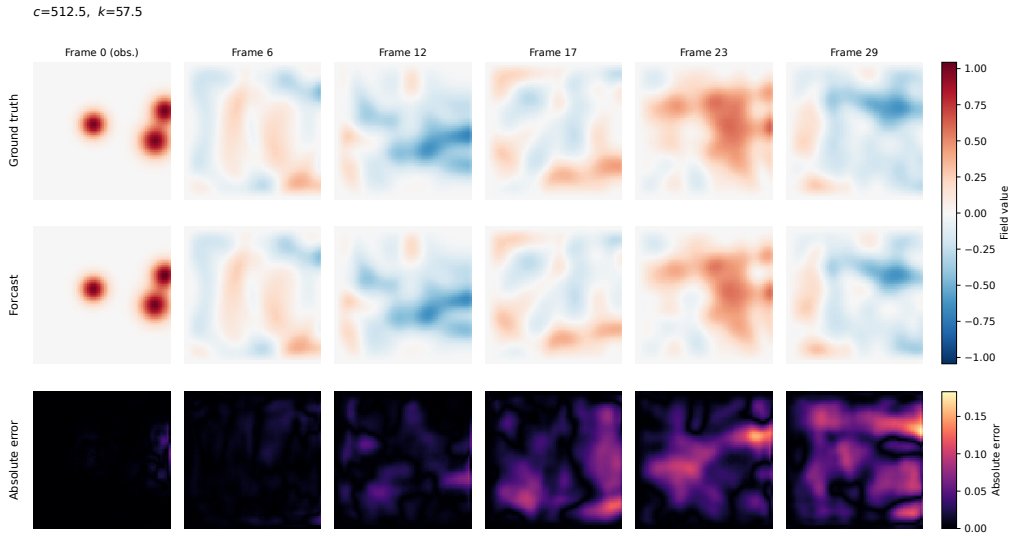


(b) OOD: $\beta = 0.2048,\ \alpha = 1.068,\ \gamma = 0.9309,\ \nu = 0$

Figure 39: **Qualitative Combined Equation forecasts under ID and OOD governing parameters.**

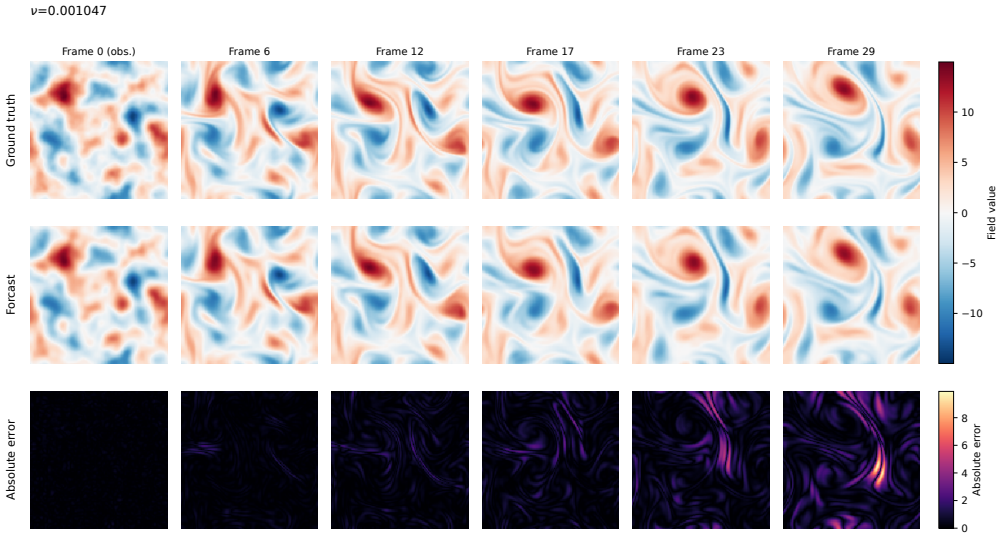


(a) ID: $c = 394.9,\ k = 0$

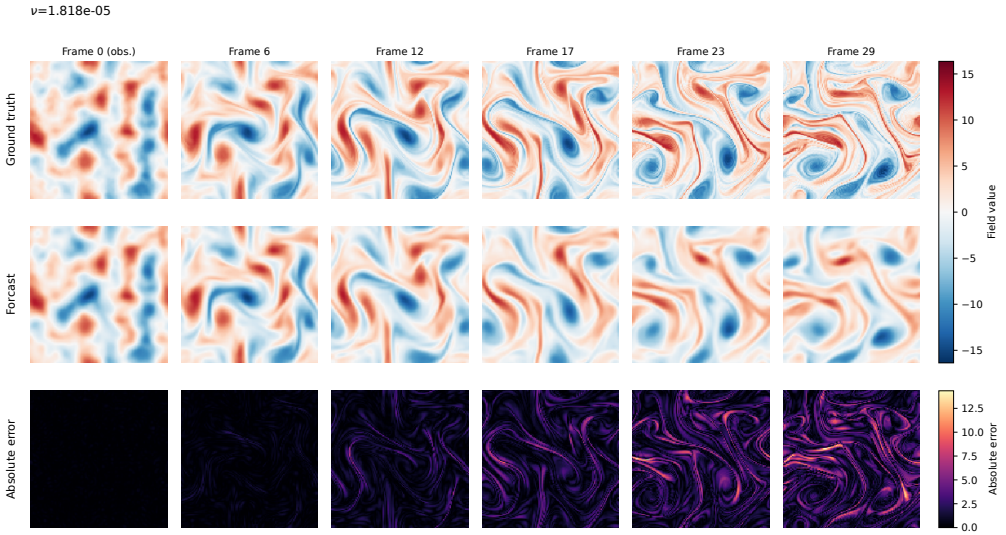


(b) OOD: $c = 512.5,\ k = 57.5$

Figure 40: **Qualitative Wave-2D forecasts under ID and OOD governing parameters.**

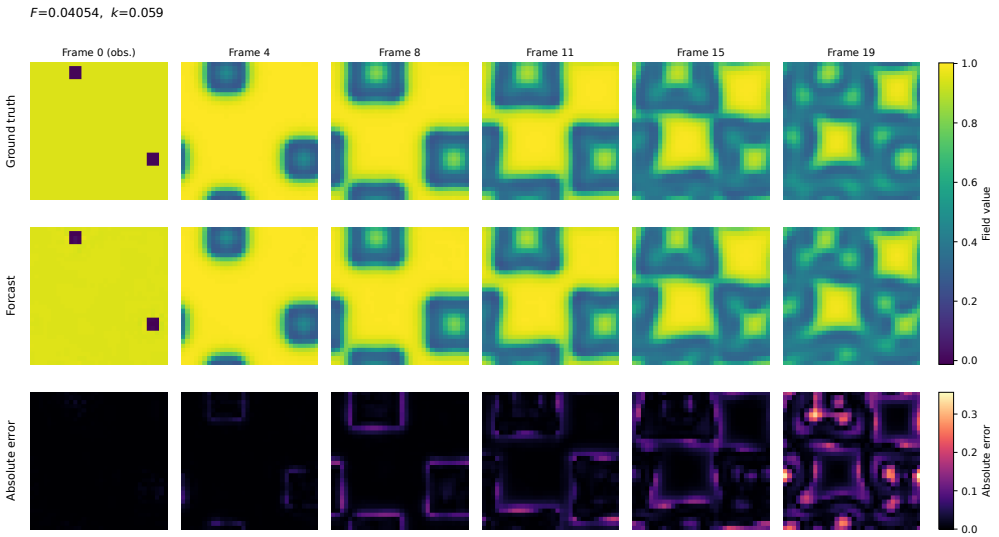


(a) ID: $\nu = 0.001047$

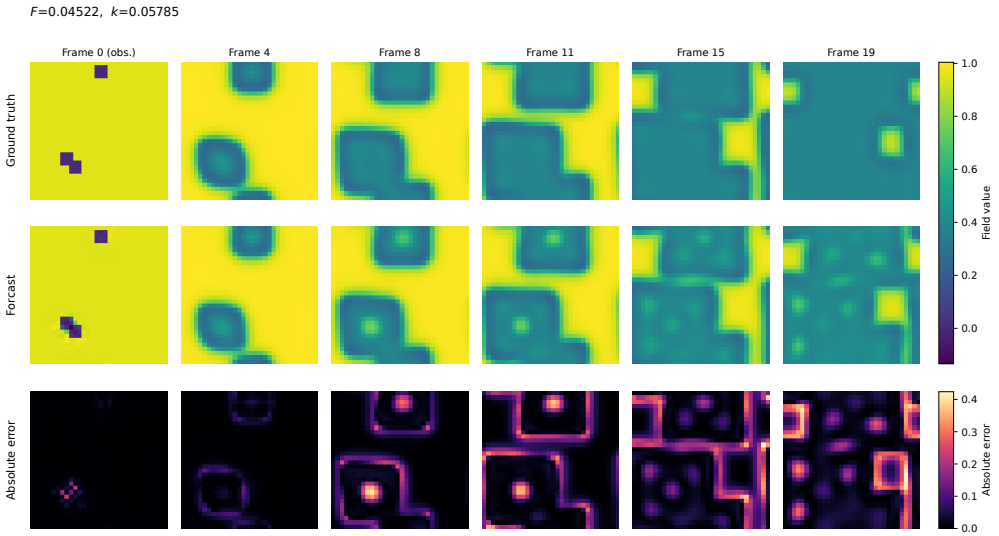


(b) OOD: $\nu = 1.818 \times 10^{-5}$

Figure 41: **Qualitative Vorticity forecasts under ID and OOD viscosity regimes.**

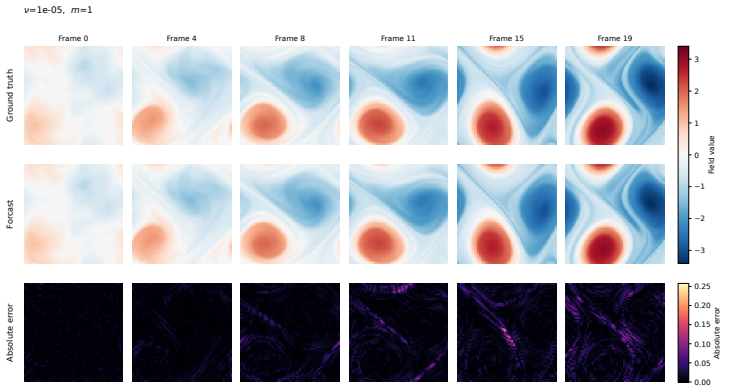


(a) ID: $F = 0.04054,\ k = 0.059$

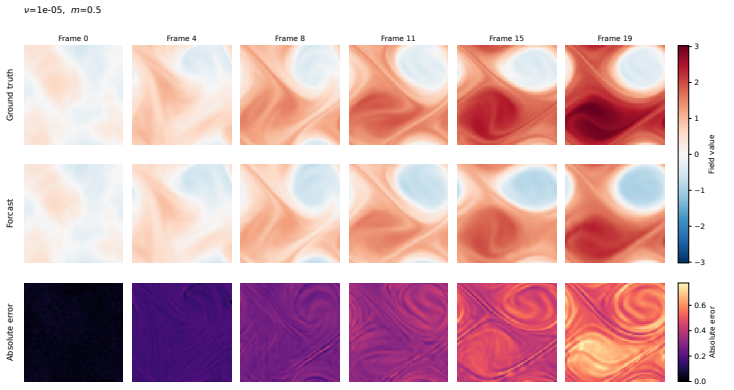


(b) OOD: $F = 0.04522,\ k = 0.05785$

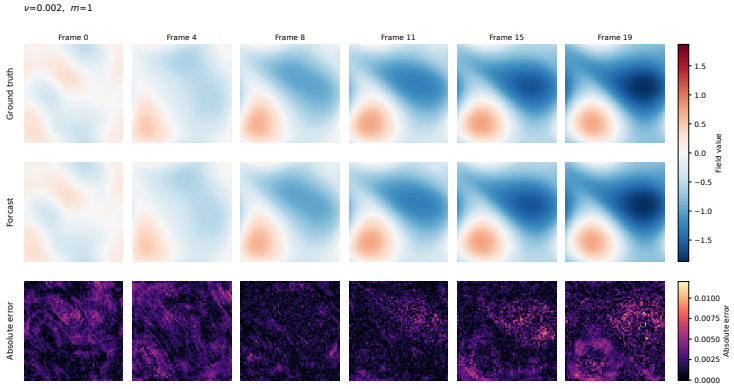
Figure 42: **Qualitative Gray–Scott forecasts under ID and OOD reaction parameters.**



(a) ID: $\nu = 10^{-5},\ m = 1$



(b) OOD forcing: $\nu = 10^{-5},\ m = 0.5$



(c) OOD viscosity: $\nu = 0.002,\ m = 1$

Figure 43: **Qualitative HeterNS forecasts under ID and OOD governing conditions.**